Virtual University	CS508-Modern Programming Language (Solved Macq's) LECTURE FROM (23 to 45)	VU LMS HELP vulmshelp.com
Junaidfazal08@gmail.com Bc190202640@vu.edu.pk	FOR MORE VISIT VULMSHELP.COME	**JUNAID MALIK** **0304-1659294**

AL-JUNAID TECH INSTITUT

1. Which of the following conversion has to be done explicitly by the programmer in C#?
   - Float to int
   - Into to Float
   - Int to long
   - Float of double
2. The dynamically generate a web page which of the following programming language is most suitable?
   - Ada
   - PHP
   - C++
   - Prolog
3. In C#, which of the following is used to give information to .Net compiler that a class is compliant with common language Specification (CLS)?
   - Class variables
   - Class attributes
   - Class definition
   - Class function
4. C# does not support the conversion of _____values to integers.
   - Boolean
   - Character
   - Floating point
   - Decimal
5. Classes and objects of PHP are similar to_____
   - Classes and objects of Basic
   - Classes and Objects of C
   - Classes and Objects of JAVA
   - Classes and Objects of C++
6. Which of the following language is used for server-side scripting?
   - C#
   - ADDA
   - PHP
   - C++
7. Which of the given operator is used to return to reference from a function in PHP?
   - &&
   - $
   - Dot
   - &
8. A variable in_____cannot be used unless and until it is not initialized.
   - ALGOL
   - C
   - C++
   - C#
9. In order to pass a value by reference to a method in C#, which of the following keyword is used with method parameter.
   - Refer
   - New

- Ref
- Explicit

10. The code written in PHP is returned to the web browser as:
    - CSS flexbox
    - Shell Scripts
    - Applets
    - Plain HTML
11. "With" statement in JavaScript is an alternative of_____
    - Multiple document.write statements
    - Single document.write statements
    - Multiple document.read statements
    - Single document.read statements
12. C# support only_____inheritance and it achieve_____inheritance through the concept of interfaces.
    - Multiple,multiple
    - Single,Multiple
    - None of the given
    - Single, Single
13. In the context of C#, which of the following is not true about delegates?
    - Delegates are value type
    - Delegates are event based
    - Delegates are like C/C++ function pointer
    - Delegates are reference type
14. FORTRAN, PL 1 and BASIC language provide -------- of variables.
    - Explicit declaration
    - Implicit declaration
    - Either implicit and explicit declaration
    - None of the given
15. Which of the given statement is true if "$a" is not equal to "$b" in PHP?
    - $a=! $
    - $a! ==$b
    - $a! =$
    - $a==! $
16. The loop control expression in any loop of C# must be of_____type.
    - Integer
    - Boolean
    - Decimal
    - Float
17. The_____operator in C# is used to test it the operator/expression is of a certain type.
    - 'IN'
    - 'IS'
    - 'AS'
    - 'OUT'

AL-JUNAID TECH INSTITUT

18. A script written in PHP are executed by:
    - Web browser
    - Client machine
    - Java Applets
    - Web Server
19. In PHP 'Switch' structure the_____starts or implements on implicit loop.
    - Return statement
    - Continue Statement
    - Break Statement
    - For Loop
20. PHP stand for
    - PHP, Home Page
    - PHP, Hypertext page
    - PHP, Hypertext Preprocessor
    - PHP, Home Preprocessor
21. Which of the Following type of number are not supported by PHP
    - Real Number
    - Complex Number
    - Single Integers
    - Unsigned Integer
22. The division of two integers (i.e 7/2) in PHP generates a/an.
    - Floating Point number
    - Integer
    - Exception
    - Runtime error
23. Arrays in JavaScript are_____based
    - Zero index
    - Length
    - Location
    - One index
24. The____model in JavaScript provides access to the contents of the browser windows.
    - COM
    - DOM
    - BOM
    - ROM
25. An_____in PHP is an ordered map.
    - Queue
    - Array
    - Stack
    - String
26. In the context of C#, a class that is defined on the body of another class is called.
    - Protected class
    - Previous Class
    - Public Class
    - Nested Class

AL-JUNAID TECH INSTITUT

27. If you want to override a method in C# then the override method must not be declared as.
    - Override
    - Static
    - Abstract
    - Virtual
28. In JavaScript, which of the given provides access to context of browser window.
    - DOM
    - ROM
    - COM
    - BOM
29. Which of the given is used before the declaration of variable in PHP?
    - # Sign
    - ? Sign
    - $ Sign
    - @ Sign
30. In how many Places a JavaScript code can be placed.
    - 3
    - 2
    - 4
    - 5
31. The "OUT" keyboard in C# used to pass_____variable by reference to a method.
    - Numeric
    - Static
    - String
    - Un-initialized
32. In_____string and Numeric values are inter convertible.
    - C#
    - JAVA
    - c/C++
    - PHP
33. A/An_____in a default mechanism for Specifying Types of variables.
    - Static Declaration
    - Run Time Declaration
    - Explicit Declaration
    - Implicit Declaration
34. In PHP, if a number is beyond the Bounds of the integer types, then It will be interpreted as a____instead.
    - Long
    - Float
    - Short
    - String
35. If X and Y are integer variables and Z is a Boolean variable then which of the Following Statement cause compilation error in C#?

- If(x=y)
{

    //Some Code

}
- If(x>y)
{

    //Some Code

    }
- If(y>x)
{

    //Some Code

    }
- If(z)
{

    //Some Code

    }

36. A/an_____ is a special word that cannot be used as a user defined name.
    - Reserved Word
    - Flag Word
    - Dummy Word
    - Abstract word
37. _____an object that describes the basic characteristics of the browser, notably its type and version.
    - Navigator
    - Location
    - Document
    - Window
38. Array in PHP are not like C/C++ arrays but they are_____
    - Templates
    - Structure
    - Maps
    - Record
39. Which of the Following Class in C# cannot be inherited from.
    - Inherit class
    - Abstract Class
    - Public class
    - Sealed Class
40. In the content of C#, a class that is defined the body of another class is called.
    - Protected Class
    - Nested Class
    - Private Class

- Public Class

41. In JavaScript navigator is a _____objects.
    - User defined
    - Built-in
    - Browser
    - Document
42. Object-Oriented programming in C# is supported through.
    - Classes
    - Functions
    - Control flow Statements
    - Packages
43. Which operator is used for Concatenation in JavaScript?
    - +
    - Dot(.)
    - |
    - $
44. When the Type is Fixed before run time then this is called as____
    - Dynamic Binding
    - Static Binding
    - Load Time Binding
    - Storage Binding
45. For event driven mechanism in C#, we use____just like____in C++.
    - Delegate,Function pointer
    - Templates,Delegate
    - Event handlers,packages
    - Events Buttons,Packages
    - EventnButtons, Graphics Code
46. In C#, the order of operator evaluation is:
    - From Left to right
    - From Right to Left
    - Arbitrary
    - Not specified

AL-JUNAID TECH INSTITUT

1) One difference LISP and PROLOG is
   - ➢ AI
   - ➢ Puzzle
   - ➢ Game
   - ➢ All of the given

2) Variable of_____is not an object of java
   - ➢ Primitive
   - ➢ Reference
   - ➢ Integer type
   - ➢ Both reference and Primitive

3) In_____every class in Java is part is overloaded
   - ➢ C++
   - ➢ Java
   - ➢ C#
   - ➢ C# and java

4) C# and C++ have_____ size.
   - ➢ Same
   - ➢ Different
   - ➢ Distinct
   - ➢ None of the given

5) _____compile into machine independent language, independent code which run in a managed execution environment.
   - ➢ C#
   - ➢ C++
   - ➢ Java
   - ➢ Ada

6) _____compile initially to an intermediate. Which can be run by interpretation or just in time compilation or an appropriate virtual machine?
   - ➢ C++ and C#
   - ➢ C# and Java
   - ➢ Java and C++
   - ➢ Ada and C++

7) How many modes for the source code are in C#?

- One
- Two
- Three
- Four

8) Which chaining type is used by Prolog?

- Backward
- Forward
- Horizontal
- Up

9) In C# Managed code is executed under the control of

- CRL
- CLR
- Both
- None

10) PHP syntax looks like_____

- ASP syntax
- C/C++
- Both
- None

11) In Java we can make pointer of_____.

- Any type
- Reference type only
- Both of the above options
- None of the given

12) Java has a String class which is not exactly an array of

- Char
- Elements
- Indices
- Long

13) How many string operators are in PHP?

- 2
- 3
- 4

➢ 10

14) C# code when compiled is converted into_____code.

➢ MISL
➢ MSIL
➢ MSLL
➢ None

15) in C#_____can be inherited from other class but no inheritance from it. On the other hand_____cannot be inherited not inheritance is possible from it is possible

➢ Private class , public class
➢ Sealed class ,struct in C#
➢ struct in C# , Sealed class
➢ public class , inherited class

16) The IS operator in C# is used for_____

➢ Run time type conversion
➢ Run time type checking
➢ Run time type casting
➢ None of the given

17) PHP is a_____typed language.

➢ Strongly
➢ Dynamic
➢ Static
➢ None of Given

18) Anything starting with Capital or underscore is a variable in_____

➢ Prolog,
➢ Lisp,
➢ COBOL
➢ None of the given

19) In C# the if statement condition is an/a_____

➢ Arithmetic expression
➢ Boolean expression
➢ Numeric expression
➢ Both Numeric expression and Boolean expression

20) In C# the value type and reference type variable are interconvert able through_____ concept.

- ➢ Tagged type
- ➢ Boxing
- ➢ interfaces
- ➢ None of the given

21) C# support only_____inheritance and it achieve_____ inheritance through the concept of interfaces.

- ➢ Multiple, multiple
- ➢ Single, multiple
- ➢ Single, single
- ➢ None of the given

22) Dynamic binding occurs at:

- ➢ Compile Time
- ➢ Design Time
- ➢ Link Time
- ➢ Run Time

23) Prolog language falls under the domain of:

- ➢ Scientific Applications
- ➢ Business Applications
- ➢ Special Purpose Languages
- ➢ None of these

24) First language that provided the concept of Pointer data type was_____. ➢ COBOL

- ➢ LISP
- ➢ PL/1
- ➢ JAVA

25) Variable use in VB without declaring decrease reliability and increase_____➢ Readably

- ➢ Writeable
- ➢ Cost
- ➢ Compile time

26) Dot operator in a_____ SNOBOL?

- ➢ Reference pointer
- ➢ Unary pointer
- ➢ Class pointer
- ➢ Binary pointer

27) The GOTO statement in SNOBOL is _____

- ➢ Explicit
- ➢ Implementation
- ➢ Punter method
- ➢ An Indirect Reference

28) Ada has _____do while loop just like C++.

- ➢ Also
- ➢ NO
- ➢ Defective
- ➢ None of the above

29) The last value execution in the_____ LISP is the return value

- ➢ Atom
- ➢ Object
- ➢ Class
- ➢ Function

30) Dotimes loop of LIPS similar working to ada:

- ➢ For
- ➢ Switch loop
- ➢ Do while
- ➢ While

31) Indexes of arrays in Ada do not start from

- ➢ 0
- ➢ 1
- ➢ 2
- ➢ 3

32) The keyword unsafe is used while dealing with_____

- ➢ Loop
- ➢ Arrays
- ➢ Pointers

➢ Classes

33) Java code when compiled is converted into_____code.

➢ Bit code
➢ Byte code
➢ Kbytes code
➢ Giga byte Code

34) Ada pointers are called Access types.

➢ True
➢ False

35) The dangling pointer problem is partially alleviated by Ada design

➢ True
➢ False

36) Following statement returns the union of the two list in LISP.

➢ >(L1 UNION L2)
➢ >(union L1 L2)
➢ >Union L1
➢ >(L1 union L2)

37) In LISP,_____is the main tool used for iteration

➢ Recursion
➢ For Loop
➢ While Loop
➢ Do-While Loop

38) In LISP, following statement returns the difference of the two lists.

➢ (Difference L1 L2)
➢ (set-difference L1 L2)
➢ (L1 Difference L2)
➢ (L1 difference L2)

39) Lists can be constructed with the help of three basic functions which are_____. ➢ scheme, common lisp and cons

➢ cons, append and scheme
➢ cons, list and append
➢ list, atoms and append

40) SNOBOL was designed for_____purpose.

- ➢ String manipulation
- ➢ AI
- ➢ Business
- ➢ Scientific

41) Computer architecture has a major influence on the design of programming language. ➢ True

- ➢ False

42) COBOL is mainly designed for_____.

- ➢ Scientific experiments
- ➢ Business application
- ➢ AI applications
- ➢ Publishing and writing algorithm

43) In SONOBAL binary operators have at least_____spaces

- ➢ 1
- ➢ 2
- ➢ 3
- ➢ 4

44) In SONOBOL can the size of the array be determined at run time.

- ➢ Yes
- ➢ No

45) _____is an important feature of Ada language used in embedded systems and operating systems. It is used in managing parallel threads of controls.

- ➢ Controls
- ➢ Tasking
- ➢ Classes
- ➢ Arrays

46) First electronic computer was_____.

- ➢ ENIAC
- ➢ Babbage analytical engine
- ➢ Itel 386
- ➢ Ibm x86

47) It is always better to find out the mistakes in the code at _____.

- ➢ Runtime
- ➢ Compile time
- ➢ Link time
- ➢ None of given option

48) Too much Orthogonality reduces Readability.

- ➢ Yes
- ➢ No

49) COBOL was the first language that brings the concept of_____

- ➢ Records
- ➢ Structure
- ➢ Object
- ➢ Variable

50) SONOBOL is case _____

- ➢ Sensitive
- ➢ In-sensitive
- ➢ Super sensitive
- ➢ Not sensitive

51) The space operator has _____ precedence than arithmetic operators.

- ➢ Higher
- ➢ Lower
- ➢ Higher and Lower
- ➢ None of the give

52) A language evaluation criteria includes following factors EXCEPT:

- ➢ Readability
- ➢ Writabilty
- ➢ Portability
- ➢ Modularity

53) The variable name has profound effect on_____.

- ➢ Readability
- ➢ Writability
- ➢ Orthogonality
- ➢ Portability

54) SIZE function in SONOBOL language is used to return the

size of _____ ➢ String

➢ Operator

➢ Variable

➢ Keyword

55) Indirect referencing in SONOBOL is same as of_____in C.

➢ Arrays

➢ Pointers

➢ Aliasing

➢ Stack

56) Following factors influences a portable language design EXCEPT:

➢ Computer architecture

➢ Readability

➢ Programmer's time

➢ Windows XP

57) + Sign is used for _____ in SONOBOL.

➢ Line Continuation

➢ Line Break

➢ Line Configuration

➢ Line Alignment

58) _____was the first object oriented language.

➢ COBOL

➢ LISP

➢ JAVA

➢ SIMULA

59) In C# struct are used as_____

➢ pointer

➢ array

➢ parameters

➢ class

60) JavaScript_____ from java language

➢ related

➢ un-related

➢ derived

- none of the given

61) In Prolog we specify_____and not_____.

- Solution, problem
- Solution, code
- Problem, solution
- None of the above

62) A_____ is a default mechanism for specify types of variable?

- implicit
- explicit
- Both of the above options
- None of the given

63) A_____ is a special word that cannot be used as a user-defined name

- Keyword
- Reserved word
- Both of the above options
- None of the given

64) We use_____in Ada.

- tagged type
- int type
- float type
- none of the given

65) Control structure is a_____ /jump..:)….. statement

- CONTINUE
- GOTO
- SWITCH
- None of the given

66) _____is place holder value that is not required.

- anonymous
- prolog
- both of the above options
- none of the given

67) LISP used in_____ and_____.

- functional paradigm, AI

- AI, functional paradigm
- Multi-paradigm, AI
- PROLOG Paradigm, AI

68) We use_____indirect referencing operator in SNOBOL

- Binary
- Binary $
- Unary
- Unary $

69) Elementary types are also called_____in ADA

- Static
- user defines
- built-in
- Discrete

70) ADA is a_____typed language

- partially strongly
- complex
- easy
- none of the given

71) _____has elaborated exception handling

- Ada
- C++
- JAVA
- COBOL

72) _____has a powerful set of operators but poor type checking

- C
- C++
- ADA
- LISP

73) _____and_____have declaration paradigm

- Prolog, SQL
- SQL, Prolog
- Lisp, SQL
- SQL, Lisp

74) 1st machine independent language is _____

- ➢ ALGOL
- ➢ COBOL
- ➢ ADA
- ➢ PASCAL

75) In C# Managed code is executed under the control of

- ➢ CLR
- ➢ CRL
- ➢ Both of the above options
- ➢ None of the given

76) The depth at which we can think is influenced by _____

- ➢ implementation details
- ➢ significance of implementation
- ➢ implemented recursion
- ➢ none of the above

77) Readability has no influence on the cost of _____

- ➢ testing
- ➢ deployment
- ➢ debugging
- ➢ coding

78) _____has distributed computing architecture

- ➢ PASCAL
- ➢ LISP
- ➢ C
- ➢ CORBA

79) Language with wide domain of application has higher_____

- ➢ Portability
- ➢ Generality
- ➢ Reliability
- ➢ Cost

80) In Java we can make pointer of_____.

- ➢ Any type
- ➢ Reference type only

81) The concept of C# jagged array is similar in function to

- C++ rectangular Array
- Java Array of Array

82) In_____Boolean expression is convertible into integer type

- C#
- C++
- JAVA
- Ada

83) _____compile initially to an intermediate. Which can be run by interpretation or just in time compilation or an appropriate virtual machine?

- C++ and C#
- C# and Java

84) For narrowing conversion which type conversion is appropriate?

- Implicit Conversion
- Explicit Conversion

85) For narrowing conversion which type conversion is appropriate?

- Implicit Conversion
- Explicit Conversion
- Both of the above options
- None of the given

86) C# and C++ have_____size.

- Same
- Different
- Distinct
- None of the given

87) C# code when compiled is converted into_____code.

- MSIL
- MISL

88) The concept of sealed class in C# is similar to _____

- Struct in C#,
- Struct in C++
- Abstract class in C#

➢ None of the given

89) Dynamic binding occurs at:

➢ Compile Time
➢ Design Time
➢ Link Time
➢ Run Time

90) Prolog language falls under the domain Of

➢ Scientific Applications
➢ Business Applications
➢ Special Purpose Languages
➢ None of these

91) For Recursion it is necessary that a language

➢ Dynamic
➢ Static
➢ Both dynamic and static
➢ Stack

92) Two important versions of LISP are _____

➢ Scheme and Atoms
➢ Scheme and Common Lisp
➢ Common Lisp and defacto
➢ List and Common Lisp

93) COBOL was the first language that brings the concept of_____

➢ Records
➢ Structure
➢ Object
➢ Variable

94) Ada pointers are called Access types.

➢ True
➢ False

95) SIZE function in SONOBOL language is used to return the size of _____

➢ String
➢ Operator

➢ Variable

96) A language is reliable if it does not create--------- during execution.

➢ Error

➢ Result

97) C++ support only -------- mode operator passing

➢ In/out

➢ union

98) Concept of package in java is similar to ---------

➢ C++

➢ C++ and c#

99)has also non-rectangular type of array

➢ C

➢ Java

C

C and C++

121. The syntax of PHP foreach loop is similar to------- loop.

- C++ foreach
- C# foreach
- Java foreach
- C++ for

122. With programming in---------we state what and not how.

- Prolog. P#92
- Algol
- C#
- Fortran

123. When the type is fixed before run time then this is called as-----

- Load time binding
- Static binding

- Dynamic binding
- None of the given

124. A programming language is ------ typed if type error are ----- detected.

- Weakly, Always
- Always, weakly
- Strongly, often
- Strongly, always. P#171

125. All objects in java are ------

- Explicit heap dynamic
- Dynamic storage binding
- Stack dynamic storage binding
- Static storage binding

126 ----------usually a logical grouping of heterogeneous elements.

- Array is
- Record is. P#176
- Structure is
- Both Structure and Record are

127. In java the arithmetic expression evaluation order is always from ------

- Center to right
- Right to Left
- Center to left
- Left to right

128. A control structure is a----------and the statement whose execution it controls.

Control algorithm

- Control statement. P#185
- Control keyword
- None of the given

129. In -------- control statements were based directly on IBM 704 hardware.

- Ada
- FORTRAN l p#185
- FORTRAN 77
- IBM Assembly language

130. We cannot pass array in---------- by value as it has no fix size. But it has fixed size in structure. C#

- Java
- C++
- C

131----------is a language for symbolic computation and its syntax is based on lambda calculus.

- ALGOL
- MATLAB
- C
- LISP p# 27

132. Ada is a -------- typed language.

- Weakly
- Random
- Strongly. P#48
- List

133. The integrity of types is checked from all together in -------

- SNOBOL
- ALGOL
- Plankal Koll
- Ada

134. A language is reliable if during execution it does not create ------

- Result
- Error
- Bytecode
- Malfunction

135 is not a valid Prolog atom.

- This _ is_ it
- Mary, _c1
- 'Hello'
- F

136 generates readable code which is easy to maintain.

- Clauses
- Backtracking. P#97
- Inference mechanism
- Facts

137. In C#, value type objects use memory of---------- data structure.

- List
- Tree
- Stack
- Heap

138 ---------- is an 8-byte integer.

- Short
- Long
- Byte
- Char

139. Simple data structure and large floating point operations are linked to --------

- Artificial intelligence language domain
- Parallel programming domain
- Scientific application programming domain
- System programming domain

140. Each source file in-------has a public class.

- C++
- C#
- Java p#98
- None of the given

141. There is no autonomic type conversion in -------

- C
- Both java and C++
- Java
- C++

142. Every class in java is part of some ----------

- Subclass
- Method
- Package p#107
- Object

143. Automatic garbage collection in java prevents ---------

- Multithreading
- Exception handling
- Indirect referencing
- Dangling pointer p#117

144. Value types in C# are inherited from ---------

- Java primitive type
- Object
- Reference
- None of the given

146. PHP supported all major Databases including ----------

- None of the given
- ODBC
- ORACLE
- SQL Server

147. In---------the Break statement may have the optional parameter or argument.

- C/C++
- Java
- PHP
- Java Script

148. Classes and objects of PHP are similar to --------

- Classes and objects of C++
- Classes and objects of java
- Classes and objects of C

- Classes and objects of Basic

149----------operator in Prolog is used for List construction and also for List dismantling.

- |(Vertical Slash). P#90
- !(Sign of exclamation)
- : (Colon)
- None of the given

150. In JavaScript --------- operator is used for concatenation.

- Dot(.)
- Plus(+). P#152
- |
- ^

151. The resultant value of variable 'x' in JavaScript statement var x = "5"-"2"; will be -------

- 5-2
- 52
- "5"-"2"
- 3

152. The-------- model in JavaScript provides access to the contents of the browser window.

- DOM. P#155
- BOM
- ROM

- COM

153. In ---------- we may specify the range and precision for floating point numbers. C#

- C/C++
- Ada
- Both Ada and C#

154 ---------- parameters are declared inside a subprogram.

- Default
- Actual
- Formal
- Copy

155. eq or equal operator in LISP is used to compare two ---------

- Atom symbols
- List symbols
- Non numeric symbols
- Numeric symbols

157. Object in JavaScript can be accessed through ---------

- Associated array
- Reference pointer method
- Reference type
- Both reference pointer method and reference type

158. Shortcut evaluation is not allowed for ----------

- Boolean operators
- Relational operators
- Bitwise operators

- Both boolean and relational operators

159. Which one of the following languages is designed for distributed computing architecture?

- CORBA
- FORTRAN
- ADA
- LISP

160. In Ada elementary types are also termed as --------

- User defined types
- Static types
- Built in types
- Public types

161. Study of programming languages helps in increasing the----------to express programming concepts.

- Capacity. P#5
- Level
- Volume
- Range

162 ----------is the designer of C++ language.

- Bjarne Stroustrup p#15
- James Goslings
- Mike Sheridan
- Patrick Naughton

163. Set membership in Prolog is different from---------- membership.

- List
- Symbol
- Join
- None of the given

164 ----------support array with built in bounds checking.

- Java
- C++
- C#
- Both java and C# p#118

165. FORTRAN, PL 1 and BASIC language provide --------

- Explicit declaration
- Implicit declaration
- Either implicit and explicit declaration
- None of the given

166. There is no arithmetic expression precedence rule in -------

- PASCAL
- Small talk
- SIMULA
- C/C++

167. C/C++ has both pre-test and post-test version of --------

- Counter controlled loop
- Logically controlled loop
- For loop
- While loop. P#192

168. C++ support only--------mode parameter passing.

- OUT
- IN
- Union
- IN/OUT

169.java has eliminated the concept of ----------

- Classes
- Variables
- Struck. P#31
- Arrays

170. In java Explicit conversions are done through --------

- Referencing
- Aliasing
- Casting. P#104
- Operator overloaded

171. Prolog database is based upon --------

- Facts
- Facts and queries
- Rules and queries
- Facts and rules. P#82

172. A predicate many have ------ or more arguments.

- Three
- Zero. P#88
- One
- Two

173 ----------is used as a place holder for a value that is not required.

- Variable
- Atom
- Constant
- Anonymous variable. P#89

174. In C#, if you don't use new keyword, you must---------- all the values in the struct by hand before you use it.

- Define
- Declare
- Create
- Initialize. P#135

175. Which statement best suited for C programming language?

- String manipulation language
- First language to provide exception handling
- With powerful set of operators but poor type checking p#30
- Introduce the notion of class

176. Which statement is true about SNOBOL?

- Its developers are computer expert
- ITE developers have limited introduction with computer science. P#46
- Is an example of aspect programming languages
- Have many data type

177. In SNOBOL, for entire assignment we use --------- operator in pattern matching statement. $(dollar)

- (.)Dot
- Unary $(Dollar)
- Binary space. P#36

178. Disjunction in Prolog is through -------- and conjunction is through ---------

- Coma (,), OR
- OR, coma(,)
- OR, Dot(.)
- Dot(.), OR

179. In ------- we can make a pointer variable of value type only.

- C++
- C#
- Java
- Both Java and C#

180 ---------- exception handling has a finally ( ) block.

- Java
- C++
- C
- Plankal kul

181. Which of the following statement is True if $a is not identical to $b in php?

- $a!==$b. P#143
- $a!=$
- $a=!$
- $a=!$

182 ---------- is the magical thing in Prolog.

- Program partition
- Inference engine

- List processing
- None of the given

183. In JavaScript navigator is---------- object.

- Built-in
- Browser. P#155
- Document
- User defined

184. The concept of package in java is similar to ---------

- Global class in C++
- Assembly in C#
- Global class in C#
- Global Class in both C++ and C# p#130

185. Array in PHP are not like C/C++ array but they are ---------

- Templates
- Records
- Maps
- Both Templates and Records

186. “with” statement in JavaScript is an alternative of ---------

- While statement
- Union statement
- Single document.write statements
- Multiple document.write statements. P#161

187. Sub ranges of integer types are not compatible with --------

- Integers types

188. Arrays in JavaScript are --------- based.

- 0

189. Unlike C/C++, arrays in PHP are_____.

- Maps
- Templates
- Records
- Both Templates and Records

190. Enumeration type in C# may take any type of_____in contrast to C++ where it take only _____.

- Numeric value, integer value
- Value type value, numeric value
- Primitive type value, reference type value
- Value type value, reference type value

191. PHP syntax looks like_____.

- ASP syntax
- C/C++ syntax
- Ada syntax
- Both ASP and Ada

192. How many <script> tags can be used in the <head> element?

- Zero
- One
- Two
- Many

193. C# code when compiled is converted into_____code.

- MSIL
- MISL

- Bytecode
- machine native code

194. In C#, strings are _____.

- Non-Immutable
- Immutable
- Neither Immutable nor Non-Immutable
- Aliasable

196. In_____the relationship between a get and set method is inherited, while in___ it has to be maintained.

- Java, C++
- C++, C#
- Ada, Java
- C#, Java or C++

197. In_____we can mke a pointer variable of value typr only.

- C++
- C#
- Java
- Both java and C++

198.In JavaScript, which of the given provides access to the contents of browser window?

- DOM
- BOM
- ROM
- COM

199. A variable in_____cannot be used unless and until it is not initialized.

- C++
- C#
- ALGOL
- C

200. In how many places a javascript code can be placed?

- 1
- 2
- 3
- 4

201. Which operator is used for concatenation in JavaScript?

- +
- |
- $
- dot(.)

202. Classes and objects of PHP are similar to _____.

- Classes and objects of Java
- Classes and objects of C++
- Classes and objects of Basic
- Classes and objects of C

203. Which of the following is not true about delegates?

- Delegates are value type
- Delegates are event based
- Delegates are like C/C++ function pointer
- Delegates are reference type

204. Which of the given statement is True if "$a" is not identical to "$b" in php?

- $a != $
- $a =! $
- $a !== $b
- $a ==! $

205. Which of the given operator is used to return a reference from a function in PHP?

- &
- &&
- $
- Dot

206. Managed or safe code in_____is executed under the control of common language runtime (CLR) with automatic garbage collection, no explicit memory allocation and de allocation and no explicit destructor.

- C++
- Java
- Ada and C++
- C#

207. In C# the value type and reference type variable are inter convertible through _____ concept.

- Tagged type
- Interfaces
- Boxing
- None of the given

209. Java code when compiled is converted into_____code.

- Bit code
- Byte code

- Kbytes code
- Giga byte Code

211. In_____Boolean expression is convertible into integer type.

- C#
- C++
- JAVA
- Ada

212. _____, an object that describes the basic characteristics of the browser, notably its type and version.

- Window
- Document
- Navigator
- Location

213. In C# _____can be inherited from other class but no inheritance from it. On the other hand_____cannot be inherited and not inheritance from it possible.

- Private class, public class
- Sealed class, struct in C#
- struct in C#, Sealed class
- public class, inheritance class

214. To make necessary arguments at run time error is called-----

- Type checking
- Exception handling
- Error handling
- None of the given

215. In----------we have to address the client side compatibility issue.

- JavaScript
- VB Script
- PHP
- Perl

216. How to compare and examine language?

- Readability
- Writability
- Reliability
- All of the given options

217. In SNOBOL 2 spaces can be used, the purpose of 1st space is for ---------- and 2nd for -------

- Pattern matching, String Concatenation
- String concatenation, Pattern matching
- String concatenation, String concatenation
- Pattern matching, Pattern matching

218. Which of the following is an incorrect option from the following statements regarding 'objectives of learning new languages'?

- Help to compare different languages.
- Help in transition from one language to other language
- Help in understanding the language piracy policy
- Help to choose a language for development of a certain application

219. The more is the simplicity of a language, the more it will always be readable.

- True
- False

220. Value upon successful completion of pattern matching, the substring matched

by the pattern is assigned to the variable as -------

- Value
- String
- Integer
- Real numbers

221. The main design goals of Ada were EXCEPT

- Program reliability and maintenance
- Design language for DOD
- Programming is a human activity therefore avoid unintentional mistakes
- Designing a portable language

222. One of the major design goals of Ada was

- To help developers to code the software in lesser time
- To make it difficult for developers to make mistakes
- To help develop an IDE for Ada
- To make testers to test Ada application easily

223. We can have nested procedures in -------- language.

- C++
- Ada
- C
- PROLOG

224. Learning the implementation details of a language help in the efficient use of that language.

- True
- False

225. Too much simplicity in language design can also cause problems.

- True
- False

226. In ----------, Enumeration type can also be used as indexes in arrays.

- Ada
- C
- C++
- LISP

227. In following is the correct syntax of 'for' statement in Ada.

- for variable in low_value..high_value loop-Loop body goes here end loop;
- for variable in low_value..high_value loop-Loop body goes here end;
- begin; for variable in low_value..high_value loop-Loop body goes here end loop;
- begin loop; for variable in low_value..high_value loop-Loop body goes here end loop;

228. Block statements in Ada is very different to a block in C.

- True
- False

229----------represents the class of languages from functional paradigm.

- LISP
- PROLOG
- Ada
- C++

230. Which statement is wrong about static variables?

- They are bound to memory cells before the execution of program
- They use to direct access the memory

- There is run time overhead of allocation and de-allocation of memory
- Storage can't be shared among variables

231. If you want an argument to a function to always be passed by reference, you can prepend

- Percentage sign (%) to the argument name in the function definition
- Dollar sign ($) to the argument name in the function definition
- An ampersand (&) to the argument name in the function definition
- Tilled sign (~) to the argument name in the function definition

232. 146. PHP supported all major Databases including ----------

- ODBC
- ORACLE
- SQL Server
- All of the given options

233. Managed or safe code in----------is executed under the control of common language runtime (CLR) with automatic garbage collection, no explicit memory allocation and de allocation and no explicit destructor.

- C++
- Java
- Ada and C++
- C#

234. ---------- is more strongly typed language then ----------

- C++, C#
- C#, C++
- C++, Java
- None of the given

235. Enumeration type in C# may take any type of---------- in contrast to C++

where it take only----------

- Numeric value, integer value
- Value type value, numeric value
- Primitive type value, reference type value

236. -------- exception inherit from exception class and --------- exception is anywhere in the program.

- Java, C#
- C++, C#
- C#, Java
- Java, C++

237. In ------ the relationship between a get and set method is inherited, while in ------ it has to be maintained.

- Java, C++
- C++, C#
- Ada, Java
- C#, Java or C++

238. Which of the following is not true about delegates?

- Delegates are value type
- Delegates are event based
- Delegates are like C/C++ function pointer
- Delegates are reference type

239. There are------ "categories" of types in java.

- Two NOTE: primitive type and reference type
- Three
- Four

- Five

240. A variable in---------cannot be used unless and until it is not initialized.

- C++
- C#
- ALGOL
- C

241. Prolog is considered to be as ---------

- Case and type insensitive
- Case and type sensitive
- Case sensitive and type insensitive
- Case insensitive and type sensitive

242. In java, out-of-bounds array index is always causes a ------- error.

- Syntax
- Compile time
- Runtime
- Logical

244. In prolog, a “member(X, T)” predicate is used to check the membership of X in ___

- Class
- Symbol
- List
- Both class and symbol

245.cannot be converted to other types in java.

- Booleans
- Char

- Double
- Int

246. The---------- operator in C# is used to test if the operator is of a certain type.

- "IN"
- "AS"
- "IS"
- "OUT"

247. In C# the "If" statement condition is an/a ------

- Arithmetic Expression
- Variable Expression
- Numeric Expression
- Boolean Expression

248. In JavaScript, which of the given provides access to the contents of browser windows?

- ROM
- DOM
- COM
- BOM

249. Which of the given is used before the declaration of variable in PHP?

- # Sign
- ? Sign
- $ Sign
- @ Sign

250. The "IS" operator in C# is used for runtime ----------

- Type conversion

- Type casting
- Type tagging
- Type checking

256. PHP stands for ----------

- PHP, Home Page
- PHP, Hypertext page
- PHP, Hypertext Preprocessor
- PHP, Home Preprocessor

257. In PHP, if a number is beyond the bounds of the integer type then it will be interpreted as a----------instead.

- String
- Short
- Long
- Float

259. Which of the following is very effective is solving complex puzzles?

- FORTRAN
- PROLOG
- ALGOL
- C/C++

260. The program is written in C# compiles to:

- Machine code
- An intermediate language (IL)
- Object code
- Source code

260. Concurrency in java is supported through:

- Resource allocation
- Batch execution
- Threads
- Operator Overloading

262. Which of the following are lightweight processes in java?

- Threads
- Packages
- Applets
- JVM

263. The methods that allow two threads to cooperate (or synchronize) with each other in a java program are:

- Wait() and run()
- run() and notify()
- Wait() and notify()
- sleep() and yield()

264. In how many ways threads can be created in java?

- Two
- Many
- Three
- Four

265. C# language is very similar to:

- Prolog
- C-language
- Delphi
- Java

AL-JUNAID TECH INSTITUT

266. Which of the following method is used to implement runable interface in java?

- Stop Thread()
- Run Thread()
- Stop()
- Run()

267. Which of the following method in a java class cannot be changed (overridden) in a subclass?

- Public void display (){-}
- Public static void area (){-}
- Private void radius (){-}
- Public final void result (){-}

268. In C#, the name of class and the name of the file in which it is saved.

- Must start with lowercase letter
- Must start with uppercase letter
- Must match up
- Do not need to match up

269. PHP supports various data types excepts ------

- Integer
- Array
- Object
- Char

270. Which of the given operator is used to return a reference from a function in PHP?

- &
- &&

- $
- Dot

271. An----------in PHP is an ordered map.

- Array
- Queue
- Stack
- String

272. In----------the “break” statement may have an optional parameter or argument.

- C/C++
- Java
- PHP
- C#

273. In PHP “switch” structure, the ---------- starts or implements an implicit loop.

- Break statement
- For loop
- Continue statement
- Return statement

274. C# provides the feature of built-in bounds checking of:

- Functions and classes
- Objects and methods
- Arrays and strings
- Interfaces and packages

275. Which of the following keywords is used to make a class non-extendable in java?

- Static
- Private
- Final
- Public

276. In a PROLOG program, the definitions of predicate sentences or phrase are known as:

- Facts
- Clauses
- Declaration
- Inferences

NOTE: **Clauses**

Clauses are definition(s) of Predicate sentence or phrase.

277. What happens when you don't initialize the fields in the constructor of a java class?

- A compile-time error occurs
- Logical error occurs
- Default values are assigned WICKIPEDIA
- Runtime error occurs

278. Expert system ---------- provide a structure to design expert system.

- Techniques
- Shells
- Lists
- Mechanism

279. The inference mechanism of PROLOG is based upon ----------

- Forward chaining
- Back tracking

- Propositional logic
- Recursion

280. What happens when you try to use an out-of bounds array index in JAVA?

- Array size is automatically incremented
- A runtime error occurs
- A garbage value is printed
- Nothing happens

281. Which of the following is treated as an atom and not as a List data type in PROLOG?

- [a, t,o,m]
- [a,b,c]
- [a]
- [ ]

282. How many types of arrays are available in C#?

- Four
- Two
- Three
- One

283. All java classes (built-in or user-defined) are sub-classes of---------- class.

- Thread
- System
- Object
- Throwable

NOTE:

In Java, all classes (built-in or user-defined) are (implicitly) subclasses of the class **Object**.

AL-JUNAID TECH INSTITUT

284. The scripts written in PHP are executed by:

- Client machine
- Web server
- Web browser
- Java Applet

285. The head and tail of a list in PROLOG is separated by:

- Semicolon (;)
- Comma (,)
- Vertical bar(|)
- Colon (:)

Final-Term Papers Solved MCQS with Reference

1. The syntax of PHP foreach loop is similar to _____ loop.
   - C++ foreach
   - **C# foreach**
   - Java foreach
   - C++ for

2. With programming in _____ we state what and not how.
   - **Prolog** **PG # 92**
   - Algol
   - C#
   - Fortran

3. When the type is fixed before run time then this is called as _____.
   - Load time binding
   - **Static binding** **Click Here For Reference**
   - Dynamic binding
   - None of the given
4. A programming language is _____ typed if type errors are _____ detected.
   - Weakly, Always
   - Always, weakly
   - Strongly, often
   - **Strongly, always** **PG # 171**
5. All objects in Java are _____.
   - Explicit heap dynamic
   - **Dynamic storage binding**
   - Stack dynamic storage binding
   - Static storage binding
6. _____ usually a logical grouping of heterogeneous elements.
   - Array is
   - **Record is** **PG # 176**
   - Structure is
   - Both Structure and Record are

7. In Java the arithmetic expression evaluation order is always from _____
   - Centre to right
   - Right to Left
   - Centre to left
   - **Left to right** **Click Here For Reference**

8. A control structure is a _____ and the statement whose execution it controls.
   - Control algorithm
   - **Control statement** **PG # 185**
   - Control keyword
   - None of the given

9. In _____ control statements were based directly on IBM 704 hardware.
   - Ada
   - **FORTAN I** **PG # 185**
   - FORTRAN 77
   - IBM Assembly language

10. We cannot pass array in _____ by value as it has no fix size. But it has fixed size in structure.
    - C#
    - Java
    - C++
    - **C**

11. _____ is a language for symbolic computation and its syntax is based on lambda calculus.

- ALGOL
- MATLAB
- C
- **LISP** **PG # 27**

12. Ada is a_____ typed language.

- Weakly
- Random
- **Strongly** **PG # 48**
- List

13. The integrity of types is checked from all together in _____.

- SNOBOL
- ALGOL
- Plankal Koll
- **Ada**

14. A language is reliable if during execution it does not create _____.

- Result
- **Error**
- Bytecode
- Malfunction

15. _____ is not a valid Prolog atom.

- this_is_it
- Mary, _c1
- **‘Hello’**
- f

16. _____ generates readable code which is easy to maintain.

- Clauses
- **Backtracking** **PG # 97**
- Inference mechanism
- Facts

17. In C#, value type objects use memory of _____ data structure.

- List
- Tree
- Stack
- **Heap**

18. _____ is an 8-byte Integer.

- Short
- **Long**
- Byte
- Char

19. Simple data structure and large floating point operations are linked to _____.

- Artificial intelligence language domain
- Parallel programming domain
- **Scientific application programming domain**
- System programming domain

20. Each source file in _____ has a public class

- C++
- C#
- **Java** **PG # 98**
- None of the given

21. There is no automatic type conversion in _____.

- **C**
- Both Java and C++
- Java
- C++

22. Every class in Java is part of some _____.

- Subclass
- Method
- **Package** **PG # 107**
- Object

23. Automatic garbage collection in Java prevents _____.

- Multithreading
- Exception handling
- Indirect referencing
- **Dangling pointers** **PG # 117**

24. Value types in C# are inherited from _____.

- Java Primitive type
- **Object**
- Reference
- None of the given

25. In C# the if statement condition is an/a _____.

- Arithmetic expression
- Boolean expression
- Numeric expression
- **Both Numeric expression and Boolean expression**

26. PHP supported all major Databases including _____.

- None of the given
- ODBC
- **ORACLE**
- SQL Server

27. In _____ the Break statement may have the optional parameter or argument.

- C/C++
- Java
- PHP
- **Java Script**

28. Classes and objects of PHP are similar to _____.

- Classes and objects of C++
- **Classes and objects of Java**
- Classes and objects of C
- Classes and objects of Basic

29. _____ operator in Prolog is used for List construction and also for List dismantling.

- **|(Vertical Slash)** **PG # 90**
- !(Sign of exclamation)
- :(Colon)
- None of the given

The | has dual purpose in Prolog; it is used for list construction as well as list dismantling.

30. In JavaScript _____ operator is used for concatenation.

- Dot (.)
- **Plus (+)** **PG # 152**
- |
- ^

31. The resultant value of variable ‘ x ‘ in JavaScript statement var x = "5” - “2”; will be _____.

- 5-2
- 52
- “5” - “2”
- **3**

32. The _____ model in JavaScript provides access to the contents of the browser window.

- **DOM** **PG # 155**
- BOM
- ROM
- COM

33. In _____ we may specify the range and precision for floating point numbers.

- C#
- C/C++
- **Ada**
- Both Ada and C#

34. _____ parameters are declared inside a subprogram.

- **Default**
- Actual
- Formal
- Copy

35. eq or equal operator in LISP is used to compare two _____.

- Atom symbols
- List symbols
- Non numeric Symbols
- **Numeric symbols**

36. In C#, the value type and reference type variables are interconvertable through _____ concept.

- Tagged type
- **Boxing**
- Interfaces
- Both Interfaces and Tagged type

37. Object in JavaScript can be accessed through _____.

- Associated array
- **Reference pointer method**
- Reference type
- Both Reference pointer method and Reference type

38. Shortcut evaluation is not allowed for _____.

- Boolean operators
- Relational operators
- **Bitwise operators**
- Both boolean and relational operators

39. Which one of the following languages is designed for distributed computing architecture?

- **CORBA**
- FORTRAN
- ADA
- LISP

40. In Ada elementary types are also termed as _____.

- **user defined types**
- static types
- built in types
- public types

41. Study of programming languages helps in increasing the _____to express programming concepts.

- **Capacity** **PG # 5**
- Level
- Volume
- Range

42. _____is the designer of C++ language.

- **Bjarne Stroustrup** **PG # 15**
- James Gosling
- Mike Sheridan
- Patrick Naughton

43. Set membership in Prolog is different from_____membership.

- **List**
- Symbol
- Join
- None of the given

44. _____support array with built in bounds checking.

- Java
- C++
- C#
- **Both Java and C#** **PG # 118**

45. FORTRAN, PL 1 and BASIC language provide_____.

- Explicit declaration
- **Implicit declaration** **PG # 169**
- Either implicit or explicit declaration
- None of the given

46. There is no arithmetic expression precedence rule in_____

- PASCAL
- **Small talk**
- SIMULA
- C/C++

47. C/C++ has both pre-test and post-test version of_____.

- Counter controlled loop
- Logically controlled loops.
- For loop
- **While loop** **PG # 192**

48. C++ support only_____mode parameter passing.

- OUT
- IN
- Union
- **IN/OUT**

49. Java has eliminated the concept of_____

- Classes
- Variables
- **Struct** **PG # 31**
- Arrays

50. In Java Explicit conversions are done through_____.

- Referencing
- Aliasing
- **Casting** **PG # 104**
- Operator overloading

51. Prolog database is based upon_____.

- Facts
- Facts and queries
- Rules and queries
- **Facts and rules** **PG # 82**

52. A predicate may have_____or more arguments.

- Three
- **Zero** **PG # 88**
- One
- Two

53. _____is used as a place holder for a value that is not required.

- Variable
- Atom
- Constant
- **Anonymous variable** **PG # 89**

54. In C#, if you don't use new keyword, you must_____all the values in the struct by hand before you use it.

- Define
- Declare
- Create
- **Initialize** **PG # 135**

55. Which statement best suited for C programming language?

- String manipulation language
- First language to provide exception handling
- **With powerful set of operators but poor type checking** **PG # 30**
- Introduce the notion of class

56. Which statement is true about SNOBOL?

- Its developers are computer experts
- **Its developers have limited introduction with computer science** **PG # 46**
- Is an example of aspect programming languages.
- Have many data types

57. In SNOBOL, for entire assignment we use_____operator in pattern matching statement.

- $(dollar)
- (.) Dot
- Unary $(Dollar)
- **Binary Space** **PG # 36**

58. Disjunction in Prolog is through_____and conjunction is through_____.

- Coma (,), OR
- **OR, coma (,)**
- OR, Dot(.)
- Dot(.) , OR

59. _____has also non rectangular type of array.

- C++
- **JAVA**
- C
- C and C++

60. _____exception handling has a finally ( ) block.

- **JAVA**
- C++
- C
- Plankal Kul

61. In_____we can make a pointer variable of value type only.

- **C++**
- C#
- Java
- Both Java and C#

62. Which of the following statement is True if $a is not identical to $b in php?

- **$a !==$b** **PG # 143**
- $a != $
- $a =! $
- $a ==! $

63. _____is the magical thing in Prolog.

- Program partition
- **Inference engine** **Click Here For Reference**
- List Processing
- None of the given

64. In JavaScript navigator is _____object.

- Built-in
- **Browser** **PG # 155**
- Document
- User defined

65. The concept of package in Java is similar to_____.

- Global Class in C++
- Assembly in C#
- Global class in C#
- **Global Class in both C++ and C#** **PG # 130**

66. Arrays in PHP are not like C/C++ arrays but they are_____.

- Templates
- Records
- **Maps**
- Both Templates and Records

An **array in PHP** is actually an ordered map. A map is a type that associates values to keys.

67. "with" statement in JavaScript is an alternative of_____.

- While statement
- Union statement
- Single document.write statement
- **Multiple document.write statements** **PG # 161**

68. Sub ranges of integer types are not compatible with_____.

- **integer types** **PG # 172**

69. Arrays in JavaScript are _____ based.

- **0** **PG # 152**

Note: Give me a feedback and your Suggestion also If you find any mistake in mcqz plz inform me Via Contact us Page on our Site. And tell me your answer with references.

☺😐☹ MUHAMMAD FAISAL ☺😐☹

MIT 4th Semester

Al-Barq Campus (VGJW01) Gujranwala

faisalgrw123@gmail.com

MEGA File Solved MCQ's For Final TERM EXAMS

CS508- Modern Programming Languages

Question No: 1 (Marks: 1) - Please choose one

One difference LISP and PROLOG is

- AI
- **Puzzle**
- Game
- All f the given

Question No: 2 (Marks: 1) - Please choose one

Variable of _____ is not an object of java

- Primitive
- Reference
- **Integer type**
- Both reference and Primitive

Question No: 3 (Marks: 1) - Please choose one

In _____ every class in Java is part is overloaded

- C++
- **Java**
- C#
- C# and java

Question No: 4 (Marks: 1) - Please choose one

C# and C++ have _____ size.

- Same
- Different
- **Distinct**
- None of the given

Question No: 5 (Marks: 1) - Please choose one

_____ compile into machine independent language, independent code which run in a managed execution environment.

- **C#**
- Java
- C++
- Ada

Question No: 6 (Marks: 1) - Please choose one

_____ compile initially to an intermediate. Which can be run by interpretation or just in time compilation or an appropriate virtual machine?

- C++ and C#
- **C# and Java**
- Java and C++
- Ada , C++

Question No: 7 (Marks: 1) - Please choose one

In _____ the relationship between a get and set method is inherited, while in _____ it has to be maintained.

- Java , C++
- C++ , C#
- Ada , Java
- **C# , Java or C++**

Question No: 8 (Marks: 1) - Please choose one

How many modes for the source code are in C#?

- One
- **Two**
- Three
- Four

Question No: 9 (Marks: 1) - Please choose one

Which chaining type is used by Prolog?

- **Backward**
- Forward
- Up
- Horizontal

Question No: 10 (Marks: 1) - Please choose one

_____exception inherits from exception class and _____ exception is anywhere in the program.

- Java , C#
- C++ , C# c
- C# , Java
- **Java ,C++**

Question No: 11 (Marks: 1) - Please choose one

In C# Managed code is executed under the control of

- CLR
- **CRL**
- Both of the above options
- None of the given

Question No: 12 (Marks: 1) - Please choose one

In_____ Boolean expression is convertible into integer type.

- C#
- **C++**
- JAVA
- Ada

Question No: 13 (Marks: 1) - Please choose one

PHP syntax looks like _____

- ASP syntax
- **C/C++ syntax**
- Both of the above options
- None of the given

Question No: 14 (Marks: 1) - Please choose one

For narrowing conversion which type conversion is appropriate?

- Implicit Conversion
- **Explicit Conversion**
- Both of the above options
- None of the given

Question No: 15 (Marks: 1) - Please choose one

In Java we can make pointer of_____.

- Any type
- **Reference type only**
- Both of the above options
- None of the given

Question No: 16 (Marks: 1) - Please choose one

The concept of package in Java is similar to _____

- **Assembly in C#**
- Global class in C#
- Function in C#
- None of the given

Question No: 17 (Marks: 1) - Please choose one

The concept of C# jagged array is similar in function to _____

- C++ rectangular Array
- **Java Array of Array**
- Java pointer Array
- C# Array

Question No: 18 (Marks: 1) - Please choose one

Java has a String class which is not exactly an array of

- **Char**
- Elements
- Indices
- Long

Question No: 19 (Marks: 1) - Please choose one

How many string operators are in PHP?

- **2**
- 3
- 4
- 10

Question No: 20 (Marks: 1) - Please choose one

C# code when compiled is converted into _____ code.

- **MSIL**
- MISL
- MSLI
- None of the given

Question No: 21 (Marks: 1) - Please choose one

In C# _____can be inherited from other class but no inheritance from it. On the other hand _____ cannot be inherited not inheritance is possible from it is possible.

- Private class , public class
- Sealed class ,struct in C#
- **struct in C# , Sealed class**
- public class , inherited class

Question No: 22 (Marks: 1) - Please choose one

The concept of sealed class in C# is similar to _____

- **Struct in C#,**
- Struct in C++
- Abstract class in C#
- None of the given

Question No: 23 (Marks: 1) - Please choose one

Enumeration type in C# may take any type of _____ in contrast to C++ where it take only _____

1) **Numeric value, integer value**
2) Value type value, numeric value
3) Primitive type value, reference type value
4) Value type value, reference type value

Question No: 24 (Marks: 1) - Please choose one

The IS operator in C# is used for_____

1) Run time type conversion
2) Run time type checking
3) Run time type casting
4) **None of the given (The 'is' operator supports run time type information)**

Question No: 25 (Marks: 1) - Please choose one

PHP is a _____typed language.

1) Strongly
2) Dynamic
3) Static
4) **None of Given**

Question No: 26 (Marks: 1) - Please choose one

_____is more strongly typed language then _____

1) C++, C#
2) C#, C++
3) C++, Java
4) **None of the given**

Question No: 27 (Marks: 1) - Please choose one

Anything starting with Capital or underscore is a variable in _____

1) **Prolog,**
2) Lisp,
3) COBOL
4) None of the given

Question No: 28 (Marks: 1) - Please choose one

In C# the if statement condition is an/a _____

1) Arithmetic expression
2) **Boolean expression**
3) Numeric expression
4) Both Numeric expression and Boolean expression

Question No: 29 (Marks: 1) - Please choose one

In C# the value type and reference type variable are interconvert able through _____ concept.

1) Tagged type
2) **Boxing**
3) Interfaces
4) None of the given

Question No: 30 (Marks: 1) - Please choose one

C# support only_____ inheritance and it achieve_____ inheritance through the concept of interfaces.

1) Multiple, multiple
2) **Single, multiple**
3) Single, single
4) None of the given

Question No: 31 (Marks: 1) - Please choose one

Managed or safe code in_____ is executed under the control of common language runtime (CLR) with automatic garbage collection, no explicit memory allocation and de allocation and no explicit destructor.

1) C++
2) Java
3) Ada and C++
4) **C#**

Question No: 32 (Marks: 1) - Please choose one

PHP supported all major Databases including_____.

1) ODBC
2) Oracle
3) **SQL Server**
4) None Of given

Question No: 33 (Marks: 1) - Please choose one

Dynamic binding occurs at:

1) Compile Time
2) Design Time
3) Link Time
4) **Run Time**

Question No: 34 (Marks: 1) - Please choose one

Prolog language falls under the domain of:

1) **Scientific Applications**
2) Business Applications
3) Special Purpose Languages
4) None of these

Question No: 35 (Marks: 1) - Please choose one

Object in java script can be accessed through ……

1) Reference pointer method
2) None of the above
3) Capital letter or underscore in PROLOG
4) Variable in PROLOG

Question No: 36 (Marks: 1) - Please choose one

For Recursion it is necessary that a language:

1) **Dynamic**
2) Static
3) Both dynamic and static
4) Stack

Question No: 37 (Marks: 1) - Please choose one

First language that provided the concept of Pointer data type was _____.

1) COBOL
2) LISP
3) **PL/1**
4) JAVA

Question No: 38 (Marks: 1) - Please choose one

Variable use in VB without declaring decrease reliability and increase _____.

1) Readably
2) **Writeable**
3) Cost
4) Compile time

Question No: 39 (Marks: 1) - Please choose one

Dot operator in a _____ SNOBOL?

1) **Reference pointer**
2) Unary pointer
3) Class pointer
4) Binary pointer

Question No: 40 (Marks: 1) - Please choose one

The GOTO statement in SNOBOL is _____

1) **Explicit**
2) Punter method
3) Implementation
4) An Indirect Reference

Question No: 41 (Marks: 1) - Please choose one

Ada has _____ do while loop just like C++.

1) Also
2) **No**
3) Defective
4) None of the above

Question No: 42 (Marks: 1) - Please choose one

The last value execution in the _____ LISP is the return value

1) Atom
2) **Object**
3) Class
4) Function

Question No: 43 (Marks: 1) - Please choose one

Dotimes loop of LIPS similar working to ada:

1) **For**
2) Switch loop
3) Do while
4) While

Question No: 44 (Marks: 1) - Please choose one

Indexes of arrays in Ada do not start from,

- **0**
- 1
- 2
- 3

Question No: 45 (Marks: 1) - Please choose one

The keyword unsafe is used while dealing with_____

1) Loop
2) Arrays
3) **Pointers**
4) Classes

Question No: 46 (Marks: 1) - Please choose one

Tasks in ____are processes and cannot share data but thread in _____ do it.

1) C# , C++
2) C++ , Ada
3) Java , C#
4) **Ada , Java**

Question No: 47 (Marks: 1) - Please choose one

Java code when compiled is converted into _____ code.

- Bit code
- **Byte code**
- Kbytes code
- Giga byte Code

Question No: 48 (Marks: 1) - Please choose one

If you want an argument to a function to always be passed by reference, you can prepend

- Percentage sign (%) to the argument name in the function definition
- Dollar sign ($)to the argument name in the function definition
- **An ampersand (&) to the argument name in the function definition**
- Tilled sign (~)to the argument name in the function definition

Question No: 49 (Marks: 1) - Please choose one

Ada pointers are called Access types.

- **TRUE**
- FALSE

Question No: 50 (Marks: 1) - Please choose one

The dangling pointer problem is partially alleviated by Ada design.

- TRUE
- **FALSE**

Question No: 51 (Marks: 1) - Please choose one

Which statement is wrong about Static variables?

- They are bound to memory cells before the execution of program
- They use to direct access the memory.
- **There is run time overhead of allocation and de-allocation of memory.**
- Storage can’t be shared among variables

Question No: 52 (Marks: 1) - Please choose one

Following is the correct structure of a "block" in Ada.

```
declare          -- declare section optional
  statements
begin
  declarations

exception          -- exception section optional
  handlers
end;
```

```
declare     -- declare section optional
   declarations
begin
   statements
exception     -- exception section optional
   handlers
end;
```

```
declare          -- declare section optional
  statements
begin
  declarations
  handlers
exception          -- exception section optional
end;
```

```
declare          -- declare section optional
  statements
  declarations
begin
  statements
exception          -- exception section optional
  handlers
end;
```

Question No: 53 (Marks: 1) - Please choose one

Following statement returns the union of the two list in LISP.

- >(L1 UNION L2)
- **>(union L1 L2)**
- >Union L1
- >(L1 union L2)

Question No: 54 (Marks: 1) - Please choose one

In LISP, _____ is the main tool used for iteration.

- **Recursion**
- For Loop
- While Loop
- Do-While Loop

Question No: 55 (Marks: 1) - Please choose one

_____ represents the class of languages from functional paradigm.

- **LISP**
- PROLOG
- Ada
- C++

Question No: 56 (Marks: 1) - Please choose one

In LISP, following statement returns the difference of the two lists.

- (Difference L1 L2)
- **(set-difference L1 L2)**
- (L1 Difference L2)
- (L1 difference L2)

Question No: 57 (Marks: 1) - Please choose one

Lists can be constructed with the help of three basic functions which are _____.

- scheme, common lisp and cons
- cons, append and scheme
- **cons, list and append**
- list, atoms and append

Question No: 58 (Marks: 1) - Please choose one

Block statement in Ada is very different to a block in C.

- True
- **False**

Question No: 59 (Marks: 1) - Please choose one

Two important versions of LISP are _____.

- Scheme and Atoms
- **Scheme and Common Lisp**
- Common Lisp and defacto
- List and Common Lisp

Question No: 60 (Marks: 1) - Please choose one

In _____, Enumeration type can also be used as indexes in arrays.

- **Ada**
- C
- C++
- LISP

Question No: 61 (Marks: 1) - Please choose one

Following is the correct syntax of 'for' statement in Ada.

```
for variable in low_value .. high_value loop
    -- Loop body goes here
end loop;
```

```
for variable in low_value .. high_value loop
    -- Loop body goes here
end;
```

```
begin;
for variable in low_value .. high_value loop
    -- Loop body goes here
end loop;
```

```
begin;
for variable in low_value .. high_value loop
    -- Loop body goes here
end loop;
```

Question No: 62 (Marks: 1) - Please choose one

SNOBOL was designed for _____ purpose.

- **String manipulation**
- AI
- Business
- Scientific

Question No: 63 (Marks: 1) - Please choose one

Computer architecture has a major influence on the design of programming language.

- **True**
- False

Question No: 64 (Marks: 1) - Please choose one

Too much simplicity in language design can also cause problems.

- **True**
- False

Question No: 65 (Marks: 1) - Please choose one

COBOL is mainly designed for _____.

- Scientific experiments
- **Business application**
- AI applications
- Publishing and writing algorithm

Question No: 66 (Marks: 1) - Please choose one

In SONOBAL binary operators have atleast____ spaces

- 1
- **2**
- 3
- 4

Question No: 67 (Marks: 1) - Please choose one

In SONOBOL can the size of the array be determined at run time.

- **Yes**
- No

Question No: 68 (Marks: 1) - Please choose one

_____ is an important feature of Ada language used in embedded systems and operating systems. It is used in managing parallel threads of controls.

- Controls
- **Tasking**
- Classes
- Arrays

Question No: 69 (Marks: 1) - Please choose one

First electronic computer was _____.

- **ENIAC**
- Babbage analytical engine
- Itel 386
- Ibm x86

Question No: 70 (Marks: 1) - Please choose one

It is always better to find out the mistakes in the code at _____.

- Runtime
- **Compile time**
- Link time
- None of given option

Question No: 71 (Marks: 1) - Please choose one

Too much Orthogonality reduces Readability.

- **True**
- False

Question No: 72 (Marks: 1) - Please choose one

COBOL was the first language that brings the concept of _____.

- **Records**
- Structure
- Object
- Variable

Question No: 73 (Marks: 1) - Please choose one

Learning the implementation details of a language helps in the efficient use of that language.

- **True**
- False

Question No: 74 (Marks: 1) - Please choose one

We can have nested procedures in _____ language.

- C++
- **Ada**
- C
- PROLOG

Question No: 75 (Marks: 1) - Please choose one

One of the major design goals of Ada was:

- **to help developers to code the software in lesser time**
- to make it difficult for developers to make mistakes
- to help develop an IDE for Ada
- to make testers to test Ada application easily

Question No: 76 (Marks: 1) - Please choose one

SONOBOL is case _____

- Sensitive
- **In-sensitive**
- Super sensitive
- Not sensitive

Question No: 77 (Marks: 1) - Please choose one

The main design goals of Ada were EXCEPT

- Program reliability and maintenance
- Design language for DoD
- Programming is a human activity therefore avoid unintentional mistakes
- **Designing a portable language**

Question No: 78 (Marks: 1) - Please choose one

The space operator has _____ precedence than arithmetic operators.

- Higher
- **Lower**
- Higher and Lower
- None of the given

Question No: 79 (Marks: 1) - Please choose one

Pattern. Variable Upon successful completion of pattern matching, the substring matched by the pattern is assigned to the variable as _____

- **Value**
- String
- Integer
- Real numbers

Question No: 80 (Marks: 1) - Please choose one

A language evaluation criteria includes following factors EXCEPT:

- Readability
- Writabilty
- Portability
- **Modularity**

Question No: 81 (Marks: 1) - Please choose one

The variable name has profound effect on _____.

- Readability
- Writability
- Orthogonality
- **Portability**

Question No: 82 (Marks: 1) - Please choose one

SIZE function in SONOBOL language is used to return the size of _____

- **String**
- Operator
- Variable
- Keyword

Question No: 83 (Marks: 1) - Please choose one

Indirect referencing in SONOBOL is same as of _____in C.

- Arrays
- **Pointers**
- Aliasing
- Stack

Question No: 84 (Marks: 1) - Please choose one

Following factors influences a portable language design EXCEPT:

- Computer architecture
- Readability
- Programmer's time
- **Windows XP**

Question No: 85 (Marks: 1) - Please choose one

The more is the Simplicity of a language, the more it will always be readable.

- **True**
- False

Question No: 86 (Marks: 1) - Please choose one

+ Sign is used for _____ in SONOBOL.

- **Line Continuation**
- Line Break
- Line Configuration
- Line Alignment

Question No: 87 (Marks: 1) - Please choose one

Which of the following is an incorrect option from the following statements regarding 'objectives of learning new languages '?

- Help to compare different languages.
- Help in transition from one language to other language.
- **Help in understanding the language piracy policy**
- Help to choose a language for development of a certain application.

Question No: 88 (Marks: 1) - Please choose one

_____ was the first object oriented language.

- COBOL
- LISP
- JAVA
- **SIMULA**

Question No: 89 (Marks: 1) - Please choose one

In SONOBOL 2 spaces can be used, the purpose of 1st space is for _____ and 2nd for:

- **Pattern matching, String Concatenation**
- String concatenation, Pattern matching
- Both of the above options
- None of the given

Question No: 90 (Marks: 1) - Please choose one

Maps in SNOBOL are also available in _____

- C
- C++
- MATLAB
- Prolog

Question No: 91 (Marks: 1) - Please choose one

In C# struct are used as_____.

- pointer
- array
- **parameters**
- class

Question No: 92 (Marks: 1) - Please choose one

JavaScript_____ from java language.

- **related**
- un-related
- derived
- none of the given

Question No: 93 (Marks: 1) - Please choose one

In_____ we have to address the client side compatibility issue.

- **JavaScript**
- VB Script
- PHP
- Perl

Question No: 94 (Marks: 1) - Please choose one

In Prolog we specify_____ and not_____ .

- **Solution, problem**
- Solution, code
- Problem, solution
- None of the above

Question No: 95 (Marks: 1) - Please choose one

A_____ is a default mechanism for specify types of variable?

- **implicit**
- explicit
- Both of the above options
- None of the given

Question No: 96 (Marks: 1) - Please choose one

A_____ is a special word that cannot be used as a user-defined name

- Keyword
- **Reserved word**
- Both of the above options
- None of the given

Question No: 97 (Marks: 1) - Please choose one

We use _____ in Ada.

- **tagged type**
- int type
- float type
- none of the given

Question No: 98 (Marks: 1) - Please choose one

Control structure is a _____ /jump..:)….. statement

- ✂ CONTINUE
- ✂ **GOTO**
- ✂ SWITCH
- ✂ None of the given

Question No: 99 (Marks: 1) - Please choose one

_____ is place holder value that is not required.

- ✂ **anonymous**
- ✂ prolog
- ✂ both of the above options
- ✂ none of the given

Question No: 100 (Marks: 1) - Please choose one

LISP used in _____ and _____.

- ✂ **functional paradigm, AI**
- ✂ AI, functional paradigm
- ✂ Multi-paradigm, AI
- ✂ PROLOG Paradigm, AI

Question No: 101 (Marks: 1) - Please choose one

First arguments in LISP is _____ symbol.

- ✂ Atom
- ✂ argument
- ✂ integer
- ✂ LIST

Question No: 102 (Marks: 1) - Please choose one

We use_____ indirect referencing operator in SNOBOL

- Binary
- Binary $
- Unary
- **Unary $**

Question No: 103 (Marks: 1) - Please choose one

Elementary types are also called _____ in ADA.

- Static
- **user defines**
- built-in
- Discrete

Question No: 104 (Marks: 1) - Please choose one

ADA is a _____ typed language

- **partially strongly**
- complex
- easy
- none of the given

Question No: 105 (Marks: 1) - Please choose one

_____ has elaborated exception handling

- **ADA**
- C++
- JAVA
- COBOL

Question No: 106 (Marks: 1) - Please choose one

_____ has a powerful set of operators but poor type checking

- **C**
- C++
- ADA
- LISP

Question No: 107 (Marks: 1) - Please choose one

_____and _____ have declaration paradigm

- **Prolog, SQL**
- SQL, Prolog
- Lisp, SQL
- SQL, Lisp

Question No: 108 (Marks: 1) - Please choose one

1st machine independent language is _____

- **ALGOL**
- COBOL
- ADA
- PASCAL

Question No: 109 (Marks: 1) - Please choose one

To make necessary arguments at run time error is called _____

- Type Checking
- **Exception handling**
- Error Handling
- None of the given

Question No: 110 (Marks: 1) - Please choose one

Language with wide domain of application has higher_____

- Portability
- **Generality**
- Reliability
- Cost

Question No: 111 (Marks: 1) - Please choose one

_____ has distributed computing architecture

- PASCAL
- LISP
- C
- **CORBA**

Question No: 112 (Marks: 1) - Please choose one

Readability has no influence on the cost of _____

- testing
- **deployment**
- debugging
- coding

Question No: 113 (Marks: 1) - Please choose one

The depth at which we can think is influenced by _____

- **implementation details**
- significance of implementation
- implemented recursion
- none of the above

Question No: 114 (Marks: 1) - Please choose one

There are no struct, union, enum, unsigned, typedef, or pointers types in _____.

- ➔ **Java**
- ➔ C
- ➔ C++
- ➔ COBOL

Question No: 115 (Marks: 1) - Please choose one

In C# Managed code is executed under the control of

- CLR
- **CRL**
- Both of the above options
- None of the given

----------Wish U Best of L|U|C|K for EXAMS ----------

Rana Abubakar Khan

If you want to learn computer programming then contact with me

truefriendlion@gmail.com

CS508 FINAL TERM PAPER SHARED BY STUDENT

ON FEBRUARY 20, 2018 AT 2:06PM

Mostly MCQs from this file.

1. enlist 3 Design issues arithmetic operators. (3)

2. what the elephant is?? a mammal . Write this in prolog(3).

3. SNOBOL concatenation and replacement example (3).

4. JAVA code with errors for inheritance given we have to find error and rewrite code. (5)

5. Enlist php opening and closing tags. (3)

6. Prolog example given to eliminate redundant elements from list(5).

7. Duality in operators of php in controlled statement. with example (5).

CS508 FINAL TERM PAPER SHARED BY STUDENT

ON FEBRUARY 23, 2018 AT 10:18PM

-overloading sideeffects.

-php odbc connection

-foreach c#

-LIST elements [head|tail] concept

-remprop syntax

Difficulty level was high atleast for me.

Best of luck everyone.

CS508 FINAL TERM PAPER SHARED BY STUDENT

ON FEBRUARY 26, 2018 AT 3:50PM

90% mcqs were from hadi and arsalan file.

Subjective:

1)Java is a object oriented programming language. suppose two pcs one is pentium 1 and the other 1 is intel core 2 duo. Discuss the Longvariable storage? And it is possible that storage of variables are same in any pc???

2)Aik program the java ka us ka output btana tha

3)concatenation ka question tha

4)SNOBOL and SNOBOL4 ka question tha

5)What is static binding and Storage binding? Also discuss advantages and disadvantages

Bus itnay hi yaad hain.

Good luck guyz!!!

CS508 FINAL TERM PAPER SHARED BY STUDENT

CS508 FINAL SUBJECTIVE (12-08-2017)

Q)The list is one of the original LISP data types, based on the concept of List in LISP, answer the following parts.

a) Write a function in LISP that creates a list by taking following three arguments as its elements.

Foo, Bar, Jack

b) Create a list that inserts symbol ‘z’ at the front of list L. (5marks)

Q)Suppose there is a function append() in prolog. This function can take 3 arguments. The first two arguments are list and the third argument is a variable which is assigned a value after appending second argument at the end of first argument. Consider the following queries and write down the outputs of them. (5marks)

1. ?- combine([a, b,c],[1,2,3],X).
2. ?- combine([foo,bar,yip,yup], [glub,glab,glib,glob], Result).
3. ?- combine([], [1, 2, 3], Z).
4. ?- combine(combine([ali, is],[a, good], X), [boy, *], Y).

Q)There is usually no need to convert between numbers and strings in PHP. Numbers and strings are interchangeable.

a)Write the resultant output after the execution of following php statements.

$foo = 1 + "11 Rabbits";

$foo = "11.0 rabbits " + 1;

b) Mention the advantages and disadvantages of using strings in numeric expressions.

(3 marks)

Q)How many ways can be used to write foreach statement using php.Write syntax in each case. (3 marks)

Q)What will be the output of the following C# code?

```
using System;
public class HelloWorld
{
   public static void Main()
   {
         string s1 = "hello";
         string s2 = s1;
         s1 = “world”;
          Console . WriteLine(s1);
    }
}
```

Q)We have a java program in which a variable named “LongVar” of type long is declared and used. Suppose this program runs on two machines; one is having Intel **Pentium I** processor and the other is having Intel **Core 2 Duo**. How many bytes of memory “LongVar” will consume on each machine? Give reasons to support your answer.

Q)From the given code snippet in C#, find out whether aliasing problem will arise or not? Justify your answer with valid reasons. (5marks)

```
object studentcs = new object();
studentcs.id = 123;
object  studentbba =  studentcs;
studentbba.id = 204;
```

CS508 FINAL TERM PAPER SHARED BY STUDENT

CS508 Current Final Term Papers In ONE thread 25-feb-2017 to 8 march 2017

Share by Student.

Q41:
Should the name of the class and the name of the file in which it is saved need to match up in case of c#. Justify your answer.
Q42:
What was the primary role behind the development of programming language like php?
Q43:
Write a prolog fact for the following statement
"The empty set is the subset of every set"
Q44:
Identify the approach used in the following record definition system.
"Employee.EmployeeName.First"
Q45:
How is dynamic binding in java different from C++?
Q46:
Explain stack dynamic type of array with two examples.
Q47:
Describe a character array of size 5 in Java also write down the default initial value that will be assigned to each element of an array when no initial value is assigned.
Q48:
How can we define many variables at once in Java Script variables with two examples?
Q49:
Point out the design issues of counted controlled loop with three examples.
Q50:

Suppose you are writing a program for some gaming application in java you need to use arrays data structure to store some values. What basic steps will be taken for array manipulation in Java with two examples?

Q51:

Write a code snipped in C# to print character of vowel array using foreach loop.

CS508 FINAL TERM PAPER SHARED BY STUDENT

ON FEBRUARY 26, 2017 AT 5:26PM

Today was my paper.

Only 15% MCQ'S from past papers 85% conceptual

3 long question From coding

How many typing method in Java ?

One was from Prolog languge , Relation and facts

How many types of constants ?

Write special atoms ?

Read your handout from 22 to 39. Mostly questions are from there.

Best of luck for your paper.

CS508 FINAL TERM PAPER SHARED BY STUDENT

CS508 CURRENT PAPER (20-08-2016)

2 marks questions:

1. Which type of inheritance is achieved through the use of interface in JAVA?
2. Does C supports Boolean expression, justify?
3. Find the length of list L' (D E F G I) by using list manipulation function?
4. What is difference b/w implicit and explicit conversion?

3 marks questions:

1. Separate predicate and clauses:

Man (symbol)

Brother (ahmad, chand)

Snowy(X), rainy(X), cold(X)

1. What is role of monitor in JAVA
2. Name any three control statement in PHP?

5 marks questions:

1. JAVA and C# commonalities?
2. Explain function side effect with occurs in arithmetic expression?
3. Write the resultant output of following function statement in LISP:

1) Intersection'(a b c) ' (b a b c)

2) Union'(c) ' (a b c)

1. Being a programmer , you are required to identify different kind of issues and tradeoffs

Face by a programmer while doing string and number inter-conversion in PHP?

CS508 FINAL TERM PAPER SHARED BY STUDENT

ON AUGUST 21, 2016 AT 10:16AM

9 mcqs from past paper
subjective
1-in c c++/java what type of operator are used for short-circuut evalution?
2-which operator are used for inequality in prolog?
3- why one have to be aware of left recursion while writing program in prolog?
4-in c# when does you absolutely have to declare a class as abstract?
5- coercion belong to which type of conversion write disadvantage of using it?
6-name file extension used in php?
7-difference b/w forward and backward chaining which is prefer?
8-what is run-time environment in case of .net similar to java in case of java explain the run time environment ? marks 5
9- annalist the merits and demerits pointers? marks 5
10 why does compiler genrate error in java code segment ?
public interface MYInterface
{
void sumint(inta,intb)
{ intc:
c=a+b;
}
void sumint(inta,intb);
es terha ka kch code tha

CS508 FINAL TERM PAPER SHARED BY STUDENT

ON AUGUST 22, 2016 AT 9:51AM

Cs508 todays Paper 8:00Am
All BC >.< MCQs were from All over the book mostly from the topic Ada ,LISP , Java , C# , Fortran ,PHP , PROLOG etc. :p
1 Question of 5 marks was from past papers
1) What code in LISP language used to compute power of x to y? 5 marks
2) what was main reason of creating php language (3) marks
3) Java code was given we have to tell the error . 5 marks
4) 1 Question was From LISP cdr and other points .3 marks
5) 1 was from php abot $ift1/2 is it valid or not. 2 marks

6) 1 question was from java script. 3 marks
7) Problems caused by Controlled loops in programing languages 5marks
3 question or thay bhol gye
Overall paper was average tha . Plzz pray for my next paper. Allah pak sab ko kamyab karay . Ameen

CS508 FINAL TERM PAPER SHARED BY STUDENT

CS508 Final Exam (20-08-2016) SUBJECTIVE ONLY

Question: 41 (Marks: 2)

Interpret the following axioms.

c:- a.
c:- b.

Question: 42 (Marks: 2)

Define value type variable with context to C#.

Modern Programming Languages (CS508)

Question: 43 (Marks: 2)

In C#, which type of variables can be used to create pointers?

Modern Programming Languages (CS508)

Question: 44 (Marks: 2)

Write a single HTML statement that creates a Text Field named "Country".

Modern Programming Languages (CS508)

Question: 45 (Marks: 3)

Write a function in LISP that return the difference of the two lists L3 and L4.

Modern Programming Languages (CS508)

Question: 46 (Marks: 3)

JavaScript is an Object Based Programming language. Identify three types of objects in java script.

Modern Programming Languages (CS508)

Question: 47 (Marks: 3)

How C# rectangular arrays syntax is different from C++ rectangular arrays syntax?

Modern Programming Languages (CS508)

Question: 48 (Marks: 3)

What is meant by compound terms (structures) in PROLOG? Give any example.

Modern Programming Languages (CS508)

Question: 49 (Marks: 5)

What does meant by limited access to variables? Also point out the design considerations for parameter passing.

Modern Programming Languages (CS508)

Question: 50 (Marks: 5)

Given the database below and study the Prolog queries below it. Indicate whether you think the goal will succeed or not by answering yes or no as prompted.

Data:

```
blue_box.
red_box.
green_circle.
blue_circle.
orange_triangle.
```

Queries:

```
?- green_circle.
?- circle_green.
?- red_triangle.
?- red_box.
?- orange_Triangle.
```

Note:- You will give your answer in "yes" or "no" only.

Modern Programming Languages (CS508)

Question: 51 (Marks: 5)

Suppose you are developing a small application using java. The application uses "arraycopy" method for copying array contents. Identify the error(s) in the code given below and also write the correct code.

```
int [ ] A, B;

A = new int[10];
// copies first 5 values from A to B
arraycopy(A, 0, B, 0, 5);
// copies last value from A into last element of B
arraycopy(A, 9, B, 4, 1);
```

Modern Programming Languages (CS508)

Question: 52 (Marks: 5)

```
var students=["Ali", "Aakash", "Taha", "Faiq"];
```

Based on the above Javascript array, what will be the output of following JavaScript statements:

1. document.write(students[2]);
2. document.write("My name is " + students[0]);
3. document.write(students[1]+students[3]);
4. students[2]="Furqan";
   document.write(students[0]+" "+students[1]+" "+students[2]+" "+students[3]);

CS508 FINAL TERM PAPER SHARED BY STUDENT

CS508 MY TODAYS PAPER: 22/08/2015

1) Which type of variable is used in Dynamic storage management and also inefficient and unreliable?

Ans:- (page # 168)

Explicit Heap Dynamic Variables

2) In how many types we can include JavaScript code in page?

Ans:- page # 141

JavaScript may be placed at three places

In the <head> element

In the <body> element

External to the HTML file

3) How many types of object in Java Script?

Ans:- Pg # 147

There are many types of objects in JavaScript

- Built-in objects (primarily type related)
- Browser objects (navigator, window etc.)
- Document objects (forms, images etc.)
- User defined objects

4) **Describe how to define an array in JavaScript? Also make an array naming** "myarray" and give 3 values red , green , blue.

Var myarray=["red", "green", "blue"];

Ans:- Pg # 144

• Arrays in JavaScript are 0 based

• We access arrays by index values

var firstArray = ["Mars", "Jupiter", "Saturn"];

5) Different types of starting tags recommended in PHP ?? give syntax

Answer:-

1. <?php

echo "this is the recommended style";

?>

2. <script language="php">

echo "this style is also available";

</script>

3. <?

echo "this is short tags style; it needs to be configured";

?>

4. <%

echo 'this is ASP-style tags; it may not be available';

%>

6) Different control statements of PHP and give examples

Answer:-

Control Statements

PHP supports the following control statements:

• if

• while

• do while

• for

• switch

• foreach

• break

• continue

7) What is the inheritance mechanism in PHP.

1) Single?
2) Multiple?

Answer:-It supports Single inheritance only and uses the keyword extends to inherit from a super class.

8) Find error and if error then correct syntax

```
Int a , b ;

Someobject.f00(ref a, b) ;
```

Answer:-

```
Int a , b ;
```


```
b= 0 ;

Someobject.f00(ref a, b) ;
```

9) What is correct to make pointer in C#

   1) **Int *a , b , c;**
   2) Int *a, *b, *c

10) What are methods used in java for synchronizing threads?

Ans:- wait() , notify()

Listed more threads

Start()

Notifyall()

Sleep()

11) System.arraycoppy(A, 1 , B , 0 , 7)

   1) What is source array?
   2) What is destination array?
   3) What is sorce position?
   4) What is destination position?
   5) Number of elements to be copied.

Answer:- PG # 125

   1) What is source array = A
   2) What is destination array = B
   3) What is sorce position = 1
   4) What is destination position = 0
   5) Number of elements to be copied = 7

12) Make an array of 5 values, with type character in java.

String [] afridi={“shalober, malikden” so on” }

;

Char [] Pakistan={ Islamabad, Karachi, peshwar, queta, Lahore }

string[] Pakistan={p,I,q};

string[] Pakistan=new string[3];

pakistan

What is Aliasing ?

Aliasing is used to assign by reference. To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable) as shown below:

<?php

$foo = ‘Fakhar';

$bar = &$foo; // Reference $foo via $bar.


```
$bar = "My name is $bar"; // Alter $bar...

echo $bar;

echo $foo; // $foo is altered too.

?>
```

Variable variable

A variable variable is like pointers that is, it maintains the address of a variable in it. This

is shown in the example below:

```
<?php

$a = ‘hello’;

$$a = 'world';

echo "$a ${$a}“; // ‘hello world”
```

PHP logical operator

AND, OR, XOR, &&, ||, !

There are two types of special words

1. Keyword
2. Reserved

Keywod is a special only in certain context.

Reserved word is a special word which can't be used as user defined words

What is binding ?

Binding is a association between an entity,

What is predictes ?

Predicates are decleartion of relation or rule. Predicte may have one or more arugument.

What is clause ?

Clauses are definition of predicate statement.

Prolog has two constants

1. Atoms
2. Number

Numbers only including integer and floationg point.

Defferent types of atoms

1. Alphanumeric atoms
2. Symbolic atoms (<, >, +, -. *)
3. Special atoms (; [], {},!)
4. Quoted atoms ('amin' mateen')

Two types of clauses

1. Rule
2. Facts

What is Propositional Logic ?

Propositional Logic is a set of Boolean statements. It involves logic connectives such as

¬ ∧ ∨ ⇔ ⇒.

What is Anonymous variable ?

The _ is a special variable. It is called the anonymous variable. It is used as a place holder for a value that is not required. For example _ is used as a variable in the following clause:

What is vertical bar in prologo ?

Vertical bar is dual purpose in prolog , It is used for list dismantling and list construction.

How many types in java ?

There is two types in java

1. Primitive types
2. Reference types

There is five function of System.arraycopy

1. Src
2. Srcpos
3. Dst
4. Dstpos
5. Count

Int [] [] A;

A= new int[5][];

A[0]=new int [2];

A[1]= new int [3];

Type converstion in java

1. Implicit convertion/205-+2
2. Explicit convertion

Access level in java

Public

Privite

Protected

Puckage

Abstract clases of java is same of Pure virtual.

Throable classes ?

There is two predefined subclasses

Erro

Exception.

C#

Two modes in c#

1. Managed code.
2. Unmanged code.

What is deference between java and c++ ?

The main deference between java and c++ is memory safe. Java is a memeory safe languge and c++ not.

C++ have single and multlple inhertencie

Java have only signle ingertain.c

Two example of predefined reference type in C#?

Answer:

1) object

2) string

CS508 FINAL TERM PAPER SHARED BY STUDENT

CS508 2016 final subjectives.

Q1:Which type of inheritance is achieved through the use of interface in JAVA?

Ans: Java directly supports single inheritance. To support concept of a interface class is used. Inheritance is achieved as shown below:

class SuperClass

{

…

}

class SubClass extends SuperClass

{ …

}

1. Q2: Does C supports Boolean expression, justify?

Ans: Traditionally, C did not provide a standard Boolean type, partly and partly to allow the programmer to make the appropriate space/time tradeoff. (Using an `int` may be faster, while using `char` may save data space. Smaller types may make the generated code bigger or slower, though, if they require lots of conversions to and from `int`.

Q3: Find the length of list L' (D E F G I) by using list manipulation function?

Ans: >(reverse L) ; reverses a list

(D E F G I)

and

>(length L) ; returns the length of list L

5

Q4: What is difference b/w implicit and explicit conversion?

Ans: An implicit conversion means that a value of one type is changed to a value of another type without any special directive from the programmer. A char can be implicitly converted to an int, a long, a float, or a double. For example, the following will compile without error:

```
char c = 'a';
```


```
int k = c;

long x = c;

float y = c;

double d = c;
```

Explicit conversions are done via casting: the name of the type to which you want a value converted is given, in parentheses, in front of the value. For example, the following code uses casts to convert a value of type double to a value of type int, and to convert a value of type double to a value of type short:

```
double d = 5.6; int k = (int)d; short s = (short)(d * 2.0);
```

1. Q5: Separate predicate and clauses:
   Man (symbol)
   Brother (ahmad, chand)
   Snowy(X), rainy(X), cold(X)

Answer: Clauses are definition(s) of Predicate sentence or phrase. It has two types: rules and facts. A rule is a function definition. It may have 0 or more conditions. A fact has all parameters as constants and it cannot contain any condition. Example

– Fact:

```
brother(ahmed, chand).
```

Saying that ahmed and chand are brothers.

Example – Rule:

```
brother(X,Y):-
man(X), man(Y),
```

father(Z,Y),

father(Z,X).

It says that two symbols, X and Y, are brothers if X is a man and Y is a man and their father is same.

Q6: What is role of monitor in JAVA?

Answer: A monitor is mechanism to control concurrent access to an object.

Thread 1:

```
public void a()
{
    Synchronized (someObject) {
        // do something (1)
    }
}
```

Thread 2:

```
public void b()
{
    synchronized(someObject) {
        // do something else (2)
    }
}
```

Q7: Name any three control statement in PHP?

Ans: PHP supports the following control statements:

- if

- while
- do while
- for
- switch
- foreach
- break
- continue

Q8: JAVA and C# commonalities?

Ans: Java and C# are very similar and have a number of commonalities.

a. Both of these languages compile into machine-independent language-independent code which runs in a managed execution environment.

b. Both C# and Java compile initially to an intermediate language: C# to Microsoft Intermediate Language (MSIL), and Java to Java bytecode. In each case the intermediate language can be run - by interpretation or just-in-time compilation - on an appropriate 'virtual machine'. In C#, however, more support is given for the further compilation of the intermediate language code into native code.

c. Both of these have garbage Collection coupled with the elimination of pointers (in C# restricted use is permitted within code marked unsafe).

d. Both have powerful reflection capabilities.

e. In case of both these languages, there is no header files, all code scoped to packages or assemblies, no problems declaring one class before another with circular dependencies.

Q9: Explain function side effect with occurs in arithmetic expression?

Ans:
Functional side effects: when a function changes a two-way parameter or a non-local variable

- Problem with functional side effects:

à When a function referenced in an expression alters another operand of the expression; e.g., for a parameter change:

```
a = 10;
/* assume that fun changes its parameter */
b = a + fun(a);
```

Q10: in c c++/java what type of operator are used for short-circuut evalution?

Ans: C, C++, and Java: use short-circuit evaluation for
the usual Boolean operators (&& and ||), but
also provide bitwise Boolean operators that are
not short circuit (& and |)

Q11: which operator are used for inequality in prolog?

Ans: `inequality \==`

`Q12`: which operator are used for inequality in prolog?

`Ans`: While writing programs one has to be aware of the left recursion, failing to do that may result in infinite loops. It has a steep learning curve and suffers from lack of standardization.

Q13: in c# when does you absolutely have to declare a class as abstract?

Ans: When at least one of the methods in the class is abstract.

When the class itself is inherited from an abstract class, but not all base abstract methods have been over-ridden.

Q14: coercion belong to which type of conversion write disadvantage of using it?

Ans: Coercion is an implicit type conversion

- Disadvantage of coercions:

They decrease in the type error detection ability of the compiler

- In most languages, all numeric types are coerced in expressions, using

Widening conversions

- In Ada, there are virtually no coercions in expressions

Q14: name file extension used in php?

Ans: .php

Q15: difference b/w forward and backward chaining which is prefer?

Ans: Backward chaining (from goal to fact) finds that D can be proven if sub-goal C is true because D :- C. The system then deduces that the sub-goal is C is true because C :- A, B. Since the system could prove C it has proven D.

To deduce a goal (theorem), the logic programming system searches axioms and combines sub-goals. For this purpose we may apply forward (from fact to goal) or backward (from goal to fact) chaining. For example, given the axioms:

C :- A, B.

D :- C.

Given that A and B are true, forward chaining (from fact to goal) deduces that C is true

because C :- A, B and then D is true because D :- C.

Q16: what is run-time environment in case of .net similar to java in case of java explain the run time environment ? (5marks)

Ans: C# is a strongly-typed object-oriented language. It is Similar to Java

and C++ in many respects. The .NET platform is centered on a Common Language

Runtime (CLR - which is similar to a JVM) and a set of libraries which can be exploited

by a wide variety of languages which are able to work together by all compiling to an intermediate language (IL).

Q17: annalist the merits and demerits pointers?

Ans: Dangling pointers

A pointer points to a heap-dynamic variable that has been deallocated

- Lost Heap-Dynamic Variables

à A heap-dynamic variable that is no longer referenced by any

program pointer

The process of losing heap-dynamic variables is called memory leakage

- Pointers are like goto's - they widen the range of cells that can be accessed

by a variable

- Pointers are necessary - so we can't design a language without them.

With programming in _____ we state what and not how.

_____ support array with built in bounds checking.

When the type is fixed before run time then this is called as_____.

FORTRAN , PL 1 and BASIC language provide_____

If type binding is dynamic then _____must be dynamic.

If type binding is dynamic then _____must be dynamic.

When the type is fixed before run time then this is called as_____.

Dangling pointer errors in Java reduce the language _____ .

There is no arithmetic expression precedence rule in _____

In Java the arithmetic expression evaluation order is always from _____

We cannot pass array in _____ by value as it has no fix size. But it has fixed size in structure.

In C programming language array is passed by_____.

Java supports _____ programming paradigm.

In Java Explicit conversions are done through _____.

The earliest form of a computer language was _____ motion.

Prolog database is based upon _____.

52

From the given code snippet in C#, find out whether aliasing problem will arise or not? Justify your answer with valid reasons.

```
object studentcs = new object();
studentcs.id = 123;
object  studentbba =  studentcs;
studentbba.id = 204;
```

51

Find out whether there is an error or not in the given C# program. If there is an error give reason.

Using System;


```
class Addition
{
   int a, b;
   public  Addition (int a, int b)
   {
      this.a = a;
      this.b = b;
   }
   public void Add()
   {
      a = 6;
      b = 7;
      result=a+b;
      Console.WriteLine(result);
   }
  private readonly int a, b;
  public static void Main()
  {
     Addition addi=new Addition(4,6);
     addi.Add();
  }
}
```

50

Explain problem with Functional Side Effects which occurs in arithmetic expressions.

Note: Use of same example given in handouts is not allowed.

49

Suppose there is a function append() in prolog. This function can take 3 arguments. The first two arguments are list and the third argument is a variable which is assigned a value after appending second argument at the end of first argument. Consider the following queries and write down the outputs of them.

1. ?- combine([a, b,c],[1,2,3],X).
2. ?- combine([foo,bar,yip,yup], [glub,glab,glib,glob], Result).
3. ?- combine([], [1, 2, 3], Z).
4. ?- combine(combine([ali, is],[a, good], X), [boy, *], Y).

48

What will be the output of the following C# code?

```
using System;
public class HelloWorld
{
  public static void Main()
  {
        string s1 = "hello";
        string s2 = s1;
        s1 = “world”;
        Console.WriteLine(s1);
    }
}
```

47

In how many ways the static type binding is performed? Briefly define each.

46

Write the resultant output of the following Php code.

```
<?php
$a = ‘hello’;
$$a = 'world';
echo "$a ${$a}“;
?>
```

45

Suppose you have to select heads from the following lists of members. Use prolog’s list head and tail concept to do this.

1. [Ali, | [Saad, Sana, Haadi]]
2. [Suleman, Arshad, | [Akbar, Aliya, Sara]]
3. [Aneela, Ainie, Fahad, Jibran | []]

44

Identify error in the following JavaScript statement and also write corrected statement.

```
x=x+1    alert(x)
```

43

Consider the following

```
short int num= 50000;
System.Console.Println(num);
```

Which type of run-time error will occur in this case? Justify your answer.

42

How C# char is different from C++ char? Give two reasons.

41

Write vowel predicate in prolog which has vowel letters list as an argument.

Arrays in PHP are not like C/C++ arrays but they are _____.

Symbolic expression in LISP is a form of _____

In JavaScript navigator is a _____object.

The document object model in Java Script include the features and issues of _____.

In JavaScript _____operator is used for concatenation.

Expert system_____ provide a structure to design expert system.

_____ operator in Prolog is used for List construction and also for List dismantling.

_____operator is used to return a reference from a function in PHP.

In php *foreach* statement works only on _____.

PHP syntax look like _____.

_____is more strongly typed language then _____.

Each Thread has its own set of _____and _____.

Every class in Java is part of some _____.

In LISP there is no concept of _____ .

List in LISP is itself a _____.

_____ may also change global variable as a side effect.

Random number generation with strict requirements is available in _____.

Which statement is true about SNOBOL?

In Ada all pointers are initialized to _____.

_____ is the correct size of a Decimal datatype.

In C#, if you don't use new keyword, you must _____ all the values in the struct by hand before you use it.

In C#, reference type objects use memory of _____ data structure.

The inference mechanism is based upon _____.
Prolog database is based upon _____.

The earliest form of a computer language was _____ motion.

CS508 FINAL TERM PAPER SHARED BY STUDENT

My today's Paper of CS508 (11am)

Total Questions: 52

MCQ'S: 40

2 Marks Q: 4

3 Marks Q: 4

5 Marks Q: 4

Some of the MCQ'S are:

1. Threads in Java are based upon the concept of Monitor
2. A class declared as 'sealed' cannot be inherited from
3. In C# the value type and reference type variable are interconvert able through _____ concept.

Tagged type

Boxing

Interfaces

None of the given

4. In _____, Boolean values can convert to integers.

C#

C++

Java

Ada

5. In _____ control statements were based directly on IBM704 hardware.

Ada

FORTRAN I

FORTRAN 77

IBM Assembly language

6. _____ is a unit level control iterative statement.

Recursion

Iteration

Loop

Switch statement

7. In ENIAC physical motion was eventually replaces by _____ signals.

Analog

Digital

Electrical

Logical

8. Function must have at least _____ return statement.

Three

Two

One

Four

9. _____ operations must have one of its parameters of tagged type.

Concurrent

Synchronized

Generic

Primitive

10. In C# value type objects use memory of _____ data structure

List

Tree

Stack

Heap

11. _____ has efficient use of processor and memory.

C++

Plankul kool

LISP

CORBA

12. _____ programming language is known as “Write Only” language.

FORTRAN

BASIC

APL

PL/1

13. Everything in LISP is considered as _____

Class

Object

Function

Symbol

14. The Era of 1970's is known as :_____

Analysis and elaboration

Effective software technology

Discovery and description of programming language concepts

Aspect-Oriented Programming

15. All primitive types in _____ have fixed size.

FORTRAN

C++

JAVA

C#

16. An array in PHP is an ordered _____

Data structure

MAP

Tree

Execution type

17. There may be _____ *elseif* within the same php *if* statements.

Several

Few

Single

Two

2 Marks Q: 4

Which operator is used to access element of a struct type?
Just like C++, the dereference operator '->' is used to access elements of a struct type.

Write the functionality of "=" operator in prolog.

Which type of binding time it is when you want to bind a non static local variable to a memory cell.
Runtime - e.g. bind a non-static local variable to a memory cell

Write the difference between public class and non public class in java.

A public class can be used in any package and a non-public class can only be used in its own package.

3 Marks Q: 4

What are the three conditions in which a string is evaluated as a float in php.

The string will evaluate as a float if it contains any of the characters '.', 'e', or 'E'. Otherwise, it will evaluate as an integer.

Coercion belongs to which type of conversion? Also write the Disadvantage of using it?

Coercion is an implicit type conversion, and it's disadvantage is that they decrease in the type error detection ability of the compiler. In most languages, all numeric types are coerced in expressions, using widening conversions. In Ada, there are virtually no coercions in expressions

Write a property named HOURS in C# which enables a client to specify a time in hours. The getter/setter for the Hours property performs the conversion between hours and seconds.

Convert the following into prolog rules:

1. Socrates is a mortal if he is a human.
2. X is an animal if X eats grass.
3. X is a sportsperson if X plays football.

5 Marks Q: 4

Write the difference between Managed and unmanaged code and also mention at least 3 scenarios where unmanaged code is useful.

MANAGED CODE

1. Managed code is executed under the control of Common Language Runtime(CRL).

2. It has automatic garbage collection. That is, the dynamically allocated memory area which is no longer is in use is not destroyed by the programmer explicitly. It is rather automatically returned back to heap by the built-in garbage collector.

3. There is no explicit memory's allocation and deallocation and there is no explicit calls to the garbage collector i.e. call to destructor.

UNMANAGED CODE (Java does not have this concept**)**

The unmanaged code provides access to memory through pointers just like C++. It is useful in many scenarios. For example:

• Pointers may be used to enhance performance in real time applications.

• **External Functions:** In non-.net DLLs some external functions requires a pointer as a parameter, such as Windows APIs that were written in C.

• **Debugging:** Sometimes we need to inspect the memory contents for debugging purposes, or you might need to write an application that analyzes another application process and memory.

In a program we need two dimensional int type array, the array has 2 rows where first row has 5 columns and second row has 3 columns. Write down the code of declaring this array in java.

int [ ][ ] A; // A is a two-dimensional array

A = new int[2][]; // A now has 2 rows, but no columns yet

A[0] = new int [5]; // A's first row has 5 column

A[1] = new int [3]; // A's second row has 3 columns

What does meant by limited access to variables? Also point out design considerations for parameter passing.

Limited access to variables, which means one-way whenever possible.

Design Considerations for Parameter Passing

1. Efficiency

2. One-way or two-way

- These two are in conflict with one another!

Good programming =>limited access to variables, which means one-way whenever possible

Efficiency =>pass by reference is fastest way to pass structures of significant size.

You are required to identify order of execution of HTML tags:


```
<body>
<h1>Hello World</h1>
<script type=“text/javascript”>
//script statements
</script>

<body>
<h3>Welcome to this website</h3>
<script type=“text/javascript”>
//script statements
</script>

<body>
<h2>Mobile Applicaions</h2>
<script type=“text/javascript”>
//script statements
</script>

</body>
```

- You can use as many **<script>** tags as you like in both the **<head>** and **<body>** and they are executed sequentially.

- ```
  <h1>Ready start</h1>
  <script language="Javascript" type="text/javascript">
      alert("First Script Ran");
  </script>
  <h2>Running…</h2>
  <script language="Javascript" type="text/javascript">
      alert("Second Script Ran");
  </script>
  <h2>Keep running</h2>
  <script language="Javascript" type="text/javascript">
      alert("Third Script Ran");
  </script>
  </h1>Stop!</h1>
  </body>
  ```

CS508 FINAL TERM PAPER SHARED BY STUDENT

on March 11, 2015 at 8:21pm

My Today Paper
Sir say MCQs saray lectures say mix diye thay.
Short question
1) Write one line html code to define text field containing "Paksitan".
2) Wrtie the use of "!" in PROLOG.
3) Static binding
4) Why variable in prolog are loosely held?
5) Which language you'll use to develop intelligent business system. Case ma medical system tha...
6) JavaScript k program ka output btana tha array ka question tha concatenation k sth.
bs ye he questions yaad hain...
  ```

CS508 FINAL TERM PAPER SHARED BY STUDENT

ON FEBRUARY 27, 2016 AT 12:09PM

1. from JS **2Marks**
2. from JS **2Marks**
3. package name inclusion (not remembered) **2Marks**
4. Algol if then statement **2Marks**
5. separate Prolog atom and variable from following **3marks**
6. why pointers are included in C#. give solid arguments **3Marks**
7. reason behind the php evolution **3Marks**
8. when string is evaluated as float **3Marks**
9. Enumeration and Immutability implementation **5marks**
10. **.**Design issue for arithmetic expressions **5Marks**
11. Two object were given, second one was initialized by first, have to prove either Aliasing exist or not **5Marks**
12. A company want to design an expert system for real time medical problem. which programming language is most appropriate? support with arguments. **5marks**

CS508 FINAL TERM PAPER SHARED BY STUDENT

ON FEBRUARY 27, 2016 AT 10:54PM

My today cs508 paper. Almost 40% mcqs were from past papers.
1. Backward chaining problem. (2)
2. Int x and x=23 identify the error and also correct the errors. (2)
3. Coercien is related which type of casting also write disadvantages of that casting. (3)
4. Give syntax of java and c+ multi dimensional arrays. (3)
5. Ek tha k we have varable named myvary of type long and if we run this in pentium 1 and core 2 duo computer to ye dono me kitni kitni memory le ga. Give reasons(3)
6. Ek boxing ka code diya to or kaha tha k output maloom kry.(5)
7. Backward chaining and foreward chaining ka question tha solve krna tha. X: Y , Z and W:X kuch is trh ka tha (5)
C# and java arrays ek dosry se related he how? And type if multi dimensional arrays in c#. (5)
8. Which languages support type arguments? Matlb k aisi konsi languages hai jin k arguments dety waqt unki tatype b dena zruri he. Name btaiye. (5)
9. Php opening and ending tags likhy (3)
Baqi yad ni aa rhy. Remember ne in ur prayers..

CS508 FINAL TERM PAPER SHARED BY STUDENT

ON FEBRUARY 29, 2016 AT 11:35AM

Mcq's were easy and simple … 20 to 25 Mcq's were from Past papers and 7 to 8 Mcq's from Midterm Specially form ADA .. and remaining were new ……
Read ADA , LISP , PROLOG, JAVA, C# , PHP , Java Script ….. No need to study last lectures 39 to 45 … hardly 2 mcqs were there from these lectures and 1 short qs ….

Subjective :

Name the Package in which built in Class for string creation is used in Java. **[2 marks]**

Name any two Special predicates used in PROLOG language. **[2 marks]**

Consider the following

double a;

float b;

If we convert "a" to "b" Does the variable "a" will lose some data and also mention the conversion type. **[2 marks]**

How JavaScript statements are terminated? **[2 marks]**

Name any three Control statements supported by PHP? **[3 marks]**

Enlist any 3 design issues related with arithmetic expression? **[3 marks]**

Which of the following sequences are PROLOG atoms and variables? **[3 marks]**

explain, X, b , < ----- > , MyVariable

A company use visual studio 2005 .. If they use numeric type with range -32,768 to 32,767 , which types of data types you will use ? **[3marks]**

Categories the following Java variables and function with primitives and Reference types ?

Long, byte, Struct, array , class **[5 marks]**

Code was given in C# and write the output .. Qs was very easy and simple **[5 marks]**

Write the name of languages that which have pretest logical loop statements **[5 marks]**

Declare a jagged array in C# with named "studentName" and there should b 3 elements in array. **[5 marks]**

CS508 FINAL TERM PAPER SHARED BY STUDENT

Today's My CS508 paper started at 29/02/2016 10:30 AM

MCQS from mostly from past papers 20 to 25.

1. Which type of variable is used in Dynamic storage management and also inefficient and unreliable?(**2Marks)**

1. why prolog used steep learning curve?**(2Marks).**
2. Write package name in java working with arrays. (**2Marks)**
3.
4. Identify the error in given statement. Also write the corrected code.

x=x+1 alert(x) **(3Marks).**

1. Write extensions of php file.(**3Marks**)
2. Write c# program using arrays which prints vowel using foreach loop.(**5Marks)**
3. Suppose you are required to write a program for gaming in java you need to used array data structure what are the basic steps working with java arrays.(**5Marks)**
4. **9.** Write a semantic model of parameter passing. **(5Marks)**
5. **10.** Write output for given code in php. And also write advantages and disadvantages.

$foo = 1 + "10 Rabbits";

$foo = "10.0 rabbits" + 1; **(5Marks)**

LISP Programming Language: An Introduction (Lecture 18-21)

Functional Programming Paradigm and LISP

Functional programming is a style of programming that emphasizes the evaluation of expressions, rather than execution of commands. The expressions in these languages are formed by using functions to combine basic values. A functional language is a language that supports and encourages programming in a functional style.

LISP is a representative language of this style of programming.

Lisp stands for "LISt Process". It was invented by John McCarthy in 1958 at MIT. It has simple data structure (atoms and lists) and makes heavy use of recursion. It is an interpretive language and has many variations. The two most popular versions are Scheme and Common Lisp (de facto industrial standard). It is the most widely used AI programming language.

Valid Objects

A LISP program has two types of elements: atoms and lists.

Atoms:

Atoms include numbers, symbols, and strings. It supports both real numbers and integers.

symbols: a symbol in LISP is a consecutive sequence of characters (no space). For example **a, x,** and **price-of-beef** are symbols. There are two special symbols: **T** and **NIL** for logical true and false.

strings: a string is a sequence of characters bounded by double quotes. For example **"this is red"** is a string.

S-expression
An **S-expression** (S stands for symbolic) is a convention for representing data or an expression in a LISP program in a text form. It is characterized by the extensive use of prefix notation with explicit use of brackets (affectionately known as Cambridge Polish notation).

S-expressions are used for both code and data in Lisp. S-expressions were originally intended only as machine representations of human-readable representation of symbols, but Lisp programmers soon started using S-expressions as the default notation.

S-expressions can either be single objects or atoms such as numbers, or lists.

Lists
List is the most important concept in LISP. A list is a group of atoms and/or lists, bounded by **(** and **)**. For example **(a b c)** and **(a (b c))** are two lists. In these examples the list **(a b c)** is a list of atoms **a, b,** and **c,** whereas **(a (b c))** is a list of two elements, the first one an atom **a**, and the second one a list **(b c)**.

Top elements of a list

The first-level elements in LISP are called top-level elements. For example top elements of list (a b c) are a, b, and c. Similarly, top elements of list (a (b c)) are a and (b c).

An empty list is represented by **nil.** It is the same as **()**.

Function calls

In LISP, a function and a function call is also a list. It uses prefix notation as shown below:

(function-name arg1 ... argn)

A function call returns function value for the given list of arguments. Functions are either provided by LISP function library or defined by the user.

Following are some examples of function calls. Note that the first symbol is the name of the function and the rest are its arguments.

```
>(+ 1 3 5)
9
```

Note that the arithmetic functions may take more than two arguments.

```
>(/ 3 5)
3/5
```

When we use two integers in division, the answer is converted into fraction.

```
>(/ 3.0 5)
0.59999999999999998
```

Note the difference between integer and real division.

```
>(sqrt 4)
2
```

Evaluation of S-expression

S-expressions are evaluated as follows:

1) Evaluate an atom.
   - numerical and string atoms evaluate to themselves;
   - symbols evaluate to their values if they are assigned values, return Error, otherwise;
   - the values of T and NIL are themselves.

2) Evaluate a list
   - evaluate every top element of the list as follows, unless explicitly forbidden:

- the first element is always a function name; evaluating it means to call the function body;
- each of the rest elements will then be evaluated, and their values returned as the arguments for the function.

Examples:

1.
>(+ (/ 3 5) 4)
23/5

The first element is + so we make a call to + function with (/ 3 5) and 4 as arguments. Now (/ 3 5) is evaluated and the answer is 3/5. After that 4 is evaluated to itself and thus returns 4. So, 3/5 and 4 are added to return 23/5.

2.
>(+ (sqrt 4) 4.0)
6.0

This example is similar to the first example.

3.
>(sqrt x)
Error: The variable
X is unbound.

This code generates an error as the symbol x is not associated with any value and thus cannot be evaluated.
setq, set, and **setf** are used to assign a value to a symbol.

For example, the following code assigns a value 3 to symbol x.

>(setq x 3.0)
3.0

setq is a special form of function (with two arguments). The first argument is a symbol which will not be evaluated and the second argument is a S-expression, which will be evaluated. The value of the second argument is assigned to be the value of the first argument.

To forbid evaluation of a symbol a **quote** or **'** is used.

For example

>(quote x)
x

Similarly if we have
>(setq x 3.0)
3.0

and then we do

>(setq y x)
3.0

then y will also have a value 3.0

But if we use a quote as shown below

<(setq y ‘x)
x

then x will not be evaluated and y will get the value x instead of 3.0.

Functions

There are many built-in function is LISP. This includes math functions as well as functions for manipulating lists. The math functions include:

- +, -, *, /, exp, expt, log, sqrt, sin, cos, tan, max, min

with the usual meanings.

List Constructors
Lists can be constructed (created and extended) with the help of three basic functions. These are **cons, list** and **append.**

cons takes two arguments. The first argument is a symbol and the second one is a list. It inserts the first argument in front of the list provided by the second argument. It is shown in the following example:

>(**cons** 'x L) ; insert symbol x at the front of list L, which is
(X A B C) ; (A B C)

list takes arbitrary number of arguments and makes a list of these arguments as shown below:

>(**list** 'a 'b 'c) ; making a list with the arguments as its elements
(A B C) ; if a, b, c have values A, B, C, then (list a b c) returns list (A B C)

append takes two lists as arguments and appends one list in front of the other as shown below:

>(**append** '(a b) '(c d))
(A B C D) ; appends one list in front of another

List Selectors
In order to select elements from a list, selectors functions are used. There are two basic selector functions known as **first** (or **car**) and **rest** (or **cdr**). The rest can be build with the help of these functions.

first (or **car**) takes a list as an argument and returns the first element of that list as shown in the following examples:

```
>(first '(a s d f))
 a

>(first '((a s) d f))
 (a s)
>(setq L '(A B C))
(A B C)

>(car L)
A
```

rest (or **cdr**) takes a list as its argument and returns a new list after removing the first element from the list. This is demonstrated in the following examples:

```
>(rest '(a s d f))
(s d f).

>(rest '((a s) d f))
 (d f)

>(rest '((a s) (d f))
 ((d f))

>(setq L '(A B C))
(A B C)

>(cdr L)
(B C)
```

Some of the other useful list manipulation functions are:

```
>(reverse L)     ; reverses a list
  (C B A)
```

and

```
>(length L)      ; returns the length of list L
 3
```

Predicates

A predicate is a special function which returns NIL if the predicate is false, T or anything other than NIL, otherwise. Predicates are used to build Boolean expressions in the logical statements.

The following comparative operators are used as functions for numerical values and return a T or NIL. =, >, <, >=, <=;

For example:

- (= (+ 2 4) (* 2 3))
- T

- (> (- 5 2) (+ 3 1))
- NIL

For non numeric values you can only check for equality using **equal** or **eq.**

Some other useful predicates are listed below:
atom: test if x is an atom
listp: test if x is a list

Also **number**, **symbolp**, **null** can be used to test whether the operand is a number, symbol, or a null value.

Set Operations

A list can be viewed as a set whose members are the top elements of the list. The list membership function is a basic function used to check whether an element is a member of a list or not. It is demonstrated with the help of the following code:

```
>(setq L '(A B C))

>(member 'b L) ; test if symbol b is a member (a top element) of L
(B C)          ; if yes, returns the sublist of L starting at the
               ; first occurrence of symbol b

>(member ‘b (cons 'b L))
(B A B C)
```

Note here that the mathematical definition of a set is different from the LISP definition. In Mathematics, a symbol cannot be repeated in a set whereas in LIST there is no such restriction.

If an element is not present in the list, it returns NIL as shown below.

```
>(member x L)
NIL            ; if no, returns NIL
```

Some of the other set operations are defined as below:

```
>(union L1 L2)          ; returns the union of the two lists
>(intersection L1 L2)   ; returns the intersection of the two lists
>(set-difference L1 L2) ; returns the difference of the two lists
```

Defining LISP functions

In LISP, defun is used to write a user-defined function. Note that different dialects of LISP may use different keywords for defining a function. The syntax of defun is as below:

(**defun** func-name (arg-1 ... Arg-n) func-body)

That is, a function has a name, list of arguments, and a body. A function returns the value of last expression in its body. This concept is demonstrated with the help of the following example:

```
>(defun y-plus (x) (+ x y))              ;definition of y-plus
```

This function adds x to y where x is the parameter passed to the function and y is a global variable since it is not defined inside the function. It work in the following manner:

```
>(setq y 2)
>(y-plus 23)
25
```

With this we introduce the concept of local and global variables. Local variables are defined in function body. Everything else is global.

Conditional control: if, when and **cond**

LISP has multiple conditional control statements. The set includes **if**, **when**, and **cond**. In the following pages we study these statements one by one.

if statement

The **if** statement has the following syntax:

(if <test> <then> <else>)

That is, an if statement has three parts: the **test**, the **then** part, and the **else** part. It works almost exactly like the if statement in C++. If the test is TRUE then the **then** part will be executed otherwise the **else** part will be executed. If there is no else part then if the test is not true then the if statement will simply return NIL. Here is an example that shows the **if** statement:

```
> (setq SCORE 78)
> 78
> (if (> score 85) ‘HIGH
                (if (and (< score 84) (> score 65)) ‘MEDIUM ‘LOW))
> MEDIUM
```

In the above **if** statement, the **then** part contains ‘HIGH and the **else** part is another **if** statement. So, with the help of nested **if** statements we can develop code with multiple branches.

cond statement

The **cond** statement in LISP is like the switch statement in C++. There is however a slight different as in this case each clause in the **cond** requires a complete Boolean test. That is, just like multiple else parts in the **if** statement where each needs a separate condition. Syntax of **cond** is as shown below:

```
>(cond (<test-1> <action-1>)
       (<test-2> <action-2>)
            ...
       (<test-k> <action-k>))
```

Each (<test-i> <action-i>) is called a clause. If test-i (start with i=1) returns T (or anything other than NIL), this function returns the value of action-i, else, it goes to the next clause. Usually, the last test is T, which always holds, meaning otherwise. Just like the **if** statement, **cond** can be nested (action-i may contain (cond ...))

This statement is explained with the help of the following example:

```
> (setf operation 'area L 100 W 50)
> 50

> (cond ((eq operation 'perimeter) (* 2 (+ L W)))
        (eq operation 'area) (* L W))
        (t 'i-do-not-understand-this)
        )
   )
> 5000
```

This program has three clauses. The first one checks whether the operation is the calculation of the perimeter or not. In this case it uses the length and width to calculate the perimeter. The control will come to the second clause if the first test is evaluated to be false or NIL. The second clause checks if the operation is about area and if it is true then it will calculate area. The third clause is the default case which will always be true. So if the first two tests fail, it will print
'i-do-not-understand-this

Recursion

Recursion is the main tool used for iteration. In fact, if you don't know recursion, you won't be able to go too far with LISP. There is a limit to the depth of the recursion, and it depends on the version of LISP. Following is an example of a recursive program in LISP. We shall be looking at a number of recursive functions in the examples.

```
(defun power (x y)
        (if (= y 0) 1 (* x (power x (1- y)))))
```

This function computes the power of x to y. That is, it computes x^y by recursively multiplying x with itself y number of times.

So

```
>(power 3 4)
81
```

Let us look at another example. In this example, we compute the length of a list, that is, we compute the number of elements in a list. The function is given as follows:

```
(defun length (x)
        (if (null x) 0
           (+ length (rest x) 1)
   )
)
```

```
> (Length '(a b c d))
> 4
```

Here are some more examples: The first function determines whether a symbol is present in a list or not and the second function computes the intersection of two lists. These functions are given below:

```
(defun member (x L)       ; determine whether x is in L or not
    (cond ((null L) nil)            ; base case 1: L is empty
        ((equal x (car L)) L)       ; base case 2: x=first(L)
        (t (member x (cdr L)))      ; recursion: test if x is in rest(L)
    ))

  (defun intersection (L1 L2)    ; compute intersection of two lists
    (cond ((null L1) nil)
          ((null L2) nil)
          ((member (car L1) L2)
              (cons (car L1) (intersection (cdr L1) L2)))
          (t (intersection (cdr L1) L2))
    ))
```

It may be noted that the intersection function is different from the mathematical definition of set intersection. In this case the function looks at all the elements of the first list, one at

a time, and checks whether that element is present in the second list. You may recall from our earlier discussion that our definition of set if different from the mathematical definition of a set where duplications are not allowed. This concept is elaborated with the help of the following examples where you pass same lists in different order and get different results.

```
> (intersection '(a b c) '(b a b c))
> (a b c)

> (intersection '(b a b c) '(a b c))
> (b a b c)
```

Following is yet another example to compute the set difference of two sets.

```
(defun set-difference (L1 L2)
        (cond ((null L1) nil)
                ((null L2) L1)
                ((not (member (car L1) L2))
                   (cons (car L1) (set-difference (cdr L1) L2)))
                (t (set-difference (cdr L1) L2))
        ))
```

Iteration: **dotimes** and **dolist**

Apart from recursion, in LISP we can write code involving loops using iterative non-recursive mechanism. There are two basic statements for that purpose: **dotimes** and **dolist.** These are discussed in the following paragraphs.

dotimes

dotimes is like a counter-control for loop. Its syntax is given as below:

(**dotimes** (count n result) body)

It executes the **body** of the loop n times where count starts with 0, ends with n-1.

The **result** is optional and is to be used to hold the computing result. If ***result*** is given, the function will return the value of ***result***. Otherwise it returns NIL. The value of the **count** can be used in the loop body.

dolist

The second looping structure is **dolist.** It is used to iterate over the list elements, one at a time. Its syntax is given below:

(**dolist** (x L result) body)

It executes the **body** for each top level element x in L. x is not equal to an element of L in each iteration, but rather x takes an element of L as its value. The value of x can be used in the loop body. As we have seen in the case of **dotimes,** the **result** is optional and is to be used to hold the computing result. If ***result*** is given, the function will return the value of ***result***. Otherwise it returns NIL.

To understand these concepts, let us look at the following examples:

```
> (setf cold 15 hot 35)
> 35
```

The following function use **dolist** to computes the number of pleasant days, that is, between cold and hot.

```
> (defun count-pleasant (list-of-temperatures)
                    (let ((count-is 0))                    ; initialize
                      (dolist (element my-list count-is)
                            (when (and (< element hot)
                                       (> element cold))
                              (setf count-is (+ count-is 1))))))

> (count-pleasant '(30 45 12 25 10 37 32))
> 3
```

Here is a very simple example which uses **dotimes:**

```
>   (defun product-as-sum (n m)
                    (let ((result 0))
                      (dotimes (count n result)
                        (setf result (+ result m)))))

> (product-as-sum 3 4)
> 12
```

Property Lists

Property lists are used to assign/access properties (attribute-value pairs) of a symbol. The following functions are used to manipulate a property list.

To assign a property: (**setf** (**get** object attribute) value)
To obtain a property: (**get** object attribute)
To see all the properties; (**symbol-plist** object)

Here is an example that elaborates this concept:

```
>(setf (get 'a 'height) 8)  ; cannot use "setq" here
                            ; setting the height property of symbol a.
8
```

```
>(get 'a 'height)  ; setting the height property of symbol a.
8

>(setf (get ‘a ‘weight) 20)
                              ; setting the weight property of symbol a.
20
```

Now we list all properties of **a**:

```
>(symbol-plist ‘a)
(WEIGHT 20 HEIGHT 8)
```

We can remove a property by using the **remprop** function as shown below:

```
> (remprop ‘a ‘WEIGHT)
T

 >(symbol-plist ‘a)
(HEIGHT 8)

> (remprop ‘a ‘HEIGHT)
T

 >(symbol-plist ‘a)
NIL
```

We can use the property list to build more meaningful functions. Here is one example:

Assume that if the name of a person’s father is known, the father’s name is given as the value of the father property. We define a function GRANDFATHER that returns the name of a person’s paternal grandfather, if known, or NIL otherwise. This function is given below:

```
(defun grandfather (x)
        (if (get x ‘father)
                (get (get x ‘father) ‘father)
                nil))
```

We now make a list of the paternal line:

```
(defun lineage (x)
        (if x
                (cons (x (lineage (get x ‘father))))))
```

The following would trace the family line from both father and mother’s side:

```
(defun ancestors (x)
        (if x
                (cons x (append (ancestors (get x 'father))
                                (ancestors (get x 'mother))))))
```

Arrays

Although the primary data structure in LISP is a list, it also has arrays. These are data type to store expressions in places identified by integer indexes. We create arrays by using the **linear-**array function as shown below:

```
(setf linear-array (make-array '(4)))
```

This statement creates a single dimension array of size 4 by the name of **linear-array** with indices from 0 to 3.

Here is an example of a two-dimensional array:

```
(setf chess-board (make-array '(8 8)))
```

Once created, we can store data at the desired index as follows:

```
(setf (aref chess-board 0 1) 'WHITE-KNIGHT)
(setf (aref chess-board 7 4) 'BLACK-KING)
```

Here, **aref** is the array reference. The above statements say that store symbol **WHITE-KNIGHT** at **chess-board** position **0 1** and store **BLACK-KING** at position **7 4**.

aref can be used to access any element in the array as shown below:

```
(aref chess-board 0 2)
WHITE-BISHOP

(aref chess-board 7 2)
BLACK-BISHOP
```

What made Lisp Different?

LISP was one of the earliest programming language. It was designed at MIT for artificial intelligence and it has since been the defacto standard language for the AI community, especially in the US.

LISP program composed of expression. They are in fact trees of expression, each of which returns a value. It has Symbol types: symbols are effectively pointer to strings stored in a hash table. One very important aspect of LISP is that the whole language is there. That is, there is no real distinction between read-time, compile-time and runtime. You can compile or run code while reading, read or run code while compiling, and read or compile code at runtime. That is, programs are expressed directly in the parse trees that get build behind the scenes when other languages are parsed, and these trees are make of

lists, which are Lisp date structures. This provides a very powerful feature allowing the programmer *to write programs that write programs*.

From a programming language design point of view, LISP was the first to introduce the following concepts:

- Conditionals: such as if-then-else construct.
- Function types: where functions are just like integers and strings
- Recursion: first language to support it.
- Dynamic typing: all variable are pointers.
- Garbage-Collection.
- Programs composed of expressions.
- A symbol type.
- The whole program is a mathematical function

Modern Programming Languages

Lecture 22-26

PROLOG - Pro**gramming in** Log**ic**

An Introduction

PROLOG stands for PROgramming in LOGic and was design in 1975 by Phillippe Roussell. It is a declarative programming language and is based upon Predicate Calculus. A program in PROLOG is written by defining predicates and rules and it has a built-in inference mechanism. Unfortunately, there is no effective standardization available.

PROLOG Paradigm

As mentioned earlier, PROLOG draws inferences from facts and rules. It is a declarative language in which a programmer only specifies facts and logical relationships. It is non-procedural. That is we only state "what" is to be done and do not specify "how" to do it. That is, we do not specify the algorithm. The language just executes the specification. In other words, program execution is carried out as a theorem proving exercise. It is similar to SQL used in databases with automated search and the ability to follow general rules.

One of the main features of this language is its ability to handle and process symbols. Hence it is used heavily in AI related applications. It is an interactive (hybrid compiled/interpreted) language and its applications include expert systems, artificial intelligence, natural language understanding, logical puzzles and games.

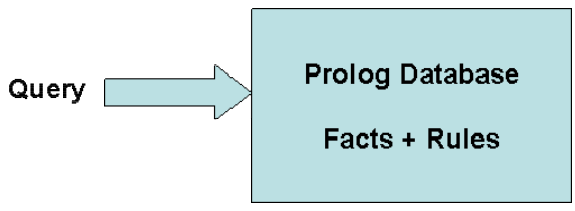
PROLOG Programming

PROLOG programming follows the following steps:

- Declaring some facts about objects and their relationships
- Defining some rules about objects and their relationships
- Asking questions about objects and their relationships

The PROLOG Programmer loads facts and rules into the database and makes queries to the database to see if a fact is in the database or can be implied from the facts and rules therein.

This is depicted in the following diagram.



Facts

- Facts are used to represent unchanging information about objects and their relationships.
- Only facts in the PROLOG database can be used for problem solving.
- A programmer inserts facts into the database by typing the facts into a file and loading (consulting) the file into a running PROLOG system.

Queries

Queries are used to retrieve information from the database. A query is a pattern that PROLOG is asked to *match* against the database and has the syntax of a compound query. It may contain variables. A query will cause PROLOG to look at the database, try to find a match for the query pattern, execute the body of the matching head, and return an answer.

Logic Programming

Logic programming is a form of declarative programming. In this form, a program is a collection of *axioms* where each axiom is a *Horn clause* of the form:

H :- B1, B2, ..., Bn.

where *H* is the *head term* and *Bi* are the *body terms*, meaning *H* is true if all *Bi* are true. A user of the program states a *goal* (a theorem) to be proven. The logic programming system attempts to find axioms using *inference steps* that imply the goal (theorem) is true. The basic proof technique is Modus Ponens i.e.

A -> B (If A is true then B is also true but it does not mean that if B is true then A is also true)

If Fact is given that

A is true

Then

B is also true

Resolution

To deduce a goal (theorem), the logic programming system searches axioms and combines sub-goals. For this purpose we may apply forward (from fact to goal) or backward (from goal to fact) chaining. For example, given the axioms:

C :- A, B.
D :- C.

Given that A and B are true, *forward chaining* (from fact to goal) deduces that *C* is true because *C :- A, B* and then *D* is true because *D :- C.*

Backward chaining (from goal to fact) finds that *D* can be proven if sub-goal *C* is true because *D* :- *C*. The system then deduces that the sub-goal is *C* is true because *C* :- *A*, *B*. Since the system could prove *C* it has proven *D*.

Prolog Uses backward chaining because it is more efficient than forward chaining for larger collections of axioms.

Examples

Let us look at some examples of facts, rules, and queries in Prolog.

Facts

Following table shows some example of facts.

English	PROLOG
"A dog is a mammal"	isa(dog, mammal).
"A sparrow is a bird"	isa(sparrow, bird).

Rules

Here are some examples of rules.

English	PROLOG
"Something is an animal	animal(X) :- isa(X, bird).
if it is a mammal or a bird"	animal(X) :- isa(X,mammal).

Queries

An now some queries:

English	PROLOG
"is a sparrow an animal?" answer: "yes"	?- animal(sparrow). yes
"is a table an animal?" answer: "no"	?- animal(table). no
"what is a dog?" answer: "a mammal"	?- isa(dog, X). X = mammal

----------END OF LECTURE 22----------

PROLOG syntax

Constants

There are two types of constants in Prolog: atoms and numbers.

Atoms:

- Alphanumeric atoms - alphabetic character sequence starting with a lower case letter. Examples: apple a1 apple_cart
- Quoted atoms - sequence of characters surrounded by single quotes. Examples: 'Apple' 'hello world'
- Symbolic atoms - sequence of symbolic characters. Examples: & < > * - + >>
- Special atoms. Examples: ! ; [ ] {}

Numbers:

Numbers include integers and floating point numbers.
Examples: 0 1 9821 -10 1.3 -1.3E102.

Variable Names

In Prolog, a variable is a sequence of alphanumeric characters beginning with an upper case letter or an underscore. Examples: Anything _var X _

Compound Terms (structures)

A compound term is an atom followed by an argument list containing terms. The arguments are enclosed within brackets and separated by commas.
Example: isa(dog, mammal)

Comments

In Prolog % is used to write a comment just like // in C++. That is after % everything till the end of the line is a comment.

Important points to note

- The names of all relationships and objects must begin with a lower case letter.
  - For example ***studies, ali, programming***
- The relationship is written first and the objects are enclosed within parentheses and are written separated by commas.
  - For example ***studies(ali, programming)***
- The full stop character '.' must come at the end of a fact.
- Order is arbitrary but it should be consistent.

An Example Prolog Program

This program has three facts and one rule. The facts are stated as follows:

```
rainy(columbo).
rainy(ayubia).
cold(ayubia).
```

The rule is:

```
snowy(X) :- rainy(X), cold(X).
```

Some queries and responses:

```
?- snowy(ayubia).
yes

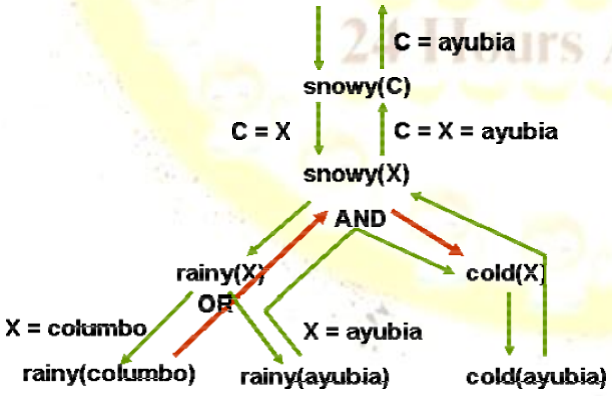
?- snowy(columbo).
no

?- snowy(lahore).
No

?- snowy(C).
C = ayubia
```

Because **rainy (ayubia)** and **cold (ayubia)** are sub-goals that are both true facts in the database, **snowy(X)** with **X=columbo** is a goal that fails, because **cold(X)** fails, triggering *backtracking.*

This inference mechanism based upon back tracking is depicted in the following diagram:



Logic Representation: Propositional Calculus

Propositional Logic is a set of Boolean statements. It involves logic connectives such as $\neg \wedge \vee \Leftrightarrow \Rightarrow$.

Example – a family

If we had the following atomic statements:

p: Ahmed is father of Belal
q: Ahmed is brother of Aslam
r : Aslam is uncle of Belal

Then the following table shows the meanings in English for some logical statements in propositional calculus:

Statement	Logical meaning
$\neg p$	Ahmed is not father of Belal
$p \vee q$	Ahmed is father of Belal or Ahmed is brother of Aslam
$p \wedge q \Rightarrow r$	If Ahmed is father of Belal and brother of Aslam then Aslam is uncle of Belal

Statements in predicate calculus can be mapped almost directly into prolog as shown below:

Predicate Calculus

- Predicates
  - man(Ahmed)
  - father(Ahmed, Belal)
  - brother(Belal, Chand)
  - brother(Chand, Delawar)
  - owns(Belal, car)
  - tall(Belal)
  - hates(Ahmed, Chand)
  - family()
- Formulae
  - $\forall X,Y,Z(man(X) \wedge man(Y) \wedge father(Z,Y) \wedge father(Z,X) \Rightarrow brother(X,Y))$
- Variables
  - X, Y and Z
- Constants
  - Ahmed, Belal, Chand, Delawar and car

Prolog code

```
Predicates
        man(symbol)
        father(symbol, symbol)
        brother(symbol, symbol)
        owns(symbol, symbol)
        tall(symbol)
        hates(symbol, symbol)
        family()
Clauses
        man(ahmed).
        father(ahmed, belal).
        brother(ahmed, chand).
        owns(belal, car).
        tall(belal).
        hates(ahmed, chand).
        family().

        brother(X,Y):-
            man(X),
            man(Y),
            father(Z,Y),
            father(Z,X).
Goal
        brother(ahmed,belal).
```

Sections of Prolog Program

Predicates
Predicates are declarations of relations or rules. They are just like function prototypes (declaration). A predicate may have zero or more arguments. Example of predicates is:

man(symbol)
family()
a_new_predicate (integer, char)

Clauses

Clauses are definition(s) of Predicate sentence or phrase. It has two types: rules and facts. A rule is a function definition. It may have 0 or more conditions. A fact has all parameters as constants and it cannot contain any condition.

Example – Fact:

```
brother(ahmed, chand).
```

Saying that ahmed and chand are brothers.

Example – Rule:

```
brother(X,Y):-
        man(X),
        man(Y),
        father(Z,Y),
        father(Z,X).
```

It says that two symbols, X and Y, are brothers if X is a man and Y is a man and their father is same.

Goal

- Goal is the objective of the program.
- There can be only and exactly one instance of the goal in a program.
- It is the only tentative section of the program.
- It may be thought as the main() function of prolog.
- The truth value of the goal is the output of the program.
- Syntactically and semantically, it is just another **clause.**
- It may be empty, simple (one goal), or compound (sub-goals).

Examples

1. brother(X,chand). % In this case the output will be the name of Chand's brother.
2. brother(ahmed,chand).% The output will be true or false.
3. brother(X,chand), father(X, Y). % The output will be Chand's and his children.

----------END OF LECTURE 23----------

Prolog Variables

Prolog variable are actually constant placeholders and NOT really variables. That is, a value to a variable can be bound only once and then it cannot be changed. These variables are loosely typed. That is, type of the variable is not defined at compile time. As mentioned earlier, a variable name starts with capital letter or underscore. Here are some examples of variable names:

- brother(ahmed, Ahmed) % ahmed is a constant whereas Ahmed is a variable
- brother(ahmed, _x) % _x is a variable
- Brother(ahmed, X) % X is a variable

Anonymous variable:
The _ is a special variable. It is called the anonymous variable. It is used as a place holder for a value that is not required. For example _ is used as a variable in the following clause:

brother(ahmed, _)

Other Syntactic Elements

Prolog has a number of other syntactic elements. Some of these are listed below:

- Special predicates
  - cut <or> !
  - not(predicate)
- Predefined predicates
  - write(arg1[,arg2[,arg3[,arg4[…..]]]]) % for output
  - nl % new line
  - readint(var) % read integer
  - readchar(var) % read char
  - readln(var) % read line into var
  - … many more related to i/o, arithmetic, OS etc
- Operators
  - Arithmetic
    - + - * div / mod
  - Relational
    - < <= => > = <> ><

PROLOG Lists

In Prolog, a list is a sequence of terms of the form

[t1, t2, t3, t4, ..., tn]

where term ti is the ith element of the list

Examples:
1. [a,b,c] is a list of three elements a, b and c.

2. [[a,list,of,lists], and, numbers,[1,2,3]]
is a four element list, the first and last are themselves lists.

3. [ ] is the 'empty list'. It is an atom not a list data type. This is a fixed symbol. And it will always have the same meaning that's why it is treated as atom not list.

Working with Lists

1. Lists are used to build compound data types.
2. A list can have elements of arbitrary type and size.
3. There is no size limits.
4. Representation in prolog is very concise (short).
5. **head – tail separator**
    a. The vertical bar '|' is used to separate the head and tail of a list as shown below:
        [<head>|<tail>]
    b. In this case, head is an element whereas tail is list of the same sort.
    c. The | has dual purpose in Prolog; it is used for list construction as well as list dismantling.

Here as some examples of lists:

```
[a, bc, def, gh, i]   % a simple list of five elements
[ ]                   % an empty list
[1,2,3]               % a simple list of three elements
[1, 2 | [3] ]         % same as above; a list of three elements with 1 and 2 as head and
                      % [3] as tail. When you join head and tail you get [1 2 3].
[1 | [2, 3] ]         % another way of writing the same thing again. This time the head
                      % is only 1 and the tail is [2 3]. Concatenation of these two gives
                      % [1 2 3].
[1, 2, 3 | [ ] ]      % same thing again but with a different head and tail formation
```

Recursion is the basic tool for writing programs in Prolog. Let us now look at some example programs in Prolog using lists.

Example 1: Calculating the length of a list by counting the first level elements

```
Domains
        list = Integer *
Predicates
        length (list, integer)
Clauses
        length([],0).                   %length of an empty list is zero

        length ([_|Tail], Len):-
                Length (Tail, TailLen),  %getting the tail length
                Len = TailLen + 1.       %the length of list is 1 plus the
                                         % length of tail
Goal
        X = [1,2,3],
        length(X,Len).
```

Example 2: Finding the last and second last element in a list

```
lastElement(X,[X]).                          % if the list has only one element
                                             % then it is the last element
lastElement(X,[_|L]) :- lastElement(X,L).    % recursive step

last_but_one(X,[X,_]).                       % similar to the above example
last_but_one(X,[_,Y|Ys]) :- last_but_one(X,[Y|Ys]).
```

----------END OF LECTURE 24----------

Example 3: Eliminate consecutive duplicates of list elements.

If a list contains repeated elements they should be replaced with a single copy of the element. The order of the elements should not be changed.

Example:

```
?- compress([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X).
X = [a,b,c,a,d,e]
```

Here is the code for this function:

```
compress([],[]).                    % there is nothing to be compressed in
compress([X],[X]).                  % an empty list or a list with only one
                                    % element

compress([X,X|Xs],Zs) :-            % if the first element is repeated, copy
        compress([X|Xs],Zs).        % it once in the new list

compress([X,Y|Ys],[X|Zs]) :-        % inspect the rest of the list and
        X \= Y,compress([Y|Ys],Zs). % repeat the process recursively
```

Sorting

Remember, in Prolog, we only specify what we want and not how to do it. Here is a classical example of this specification. In the following program segment, in order to sort a list, we only specify what is meant by sorting and not how to do it.

```
sorted ([ ]).               % an empty list or list with only one element
sorted ([X]).               % is already sorted
sorted ([X, Y | list]) :-   % recursive step
        X <= Y,             % a list is sorted if every element in the list
        sorted ([Y | list]). % is smaller than the next element in the list.
```

List Membership

```
member(X, [X|T]).           % X is a member of the list if it is the first
                            % element
```

```
member(X, [H|T]) :- member(X, T).
                        % otherwise recursively check it in the rest of
                        % the list
```

subset

```
subset( [ ], Y ).          % The empty set is a subset of every set.
subset( [ A | X ], Y ) :-  % recursive step
        member( A, Y ), subset( X, Y ).
```

intersection

```
% Assumes lists contain no duplicate elements.
intersection( [ ], X, [ ] ).
intersection( [ X | R ], Y, [ X | Z ] ) :-
                member( X, Y ), !, intersection( R, Y, Z ).
intersection( [ X | R ], Y, Z ) :- intersection( R, Y, Z ).
```

union

```
union( [ ], X, X ).
union( [ X | R ], Y, Z ) :- member( X, Y ), !, union( R, Y, Z ).
union( [ X | R ], Y, [ X | Z ] ) :- union( R, Y, Z ).
```

Puzzles- Who owns the zebra

According to an e-mail source, this puzzle originated with the German Institute of Logical Thinking, Berlin, 1981. [It's possible.]

1. There are five houses.
2. Each house has its own unique color.
3. All house owners are of different nationalities.
4. They all have different pets.
5. They all drink different drinks.
6. They all smoke different things.
7. The English man lives in the red house.
8. The Swede has a dog.
9. The Dane drinks tea.
10. The green house is on the left side of the white house.
11. In the green house, they drink coffee.
12. The man who smokes cigarette has birds.
13. In the yellow house, they smoke pipe.
14. In the middle house, they drink milk.
15. The Norwegian lives in the first house.
16. The man who smokes cigar lives in the house next to the house with cats.
17. In the house next to the house with the horse, they smoke pipe.
18. The man who smokes hukka drinks soda.
19. The German smokes bedi.
20. The Norwegian lives next to the blue house.

21. They drink water in the house that is next to the house where they smoke cigar.
22. Who owns the zebra?

Try solving this puzzle on your own. Now imagine writing a computer program to solve it. Which was more difficult?

This is an example of a completely specified solution which doesn't appear to be specified at all. The constraints are such that the answer is unique, but they are stated in such a way that it is not at all obvious (to this human, at least) what the answer is.

In Prolog, we could express each of these specifications, then let Prolog's search strategy (database search engine, automated theorem prover) search for a solution. We don't have to worry about *how* to solve the problem -- we only have the specify *what* is to be solved.

```
%  -*-prolog-*-
%
%  The Zebra puzzle:  Who owns the zebra?
%
%  According to a recent e-mail source, this puzzle originated with
%  the German Institute of Logical Thinking, Berlin, 1981. [It's
possible.]
%  The terms of the puzzle are included as comments in the code below.
%
%  Solution by Jonathan Mohr (mohrj@augustana.ca)
%  Augustana University College, Camrose, AB, Canada  T4V 2R3


%  Invoke this predicate if you just want to see the answer to the
%  question posed at the end of the puzzle.
solve :-
    solve(_).


%  Invoke this predicate (with a variable parameter) if you want to
%  see a complete solution.
solve(S) :-

% There are five houses.
% Each house has its own unique color.
% All house owners are of different nationalities.
% They all have different pets.
% They all drink different drinks.
% They all smoke different cigarettes.

% (The constraints that all colors, etc., are different can only be
% applied after all or most of the variables have been instantiated.
% See below.)

% S = [[Color1, Nationality1, Pet1, Drink1, Smoke1] |_]
% The order of the sublists is the order of the houses, left to right.
    S = [[C1,N1,P1,D1,S1],
         [C2,N2,P2,D2,S2],
         [C3,N3,P3,D3,S3],
         [C4,N4,P4,D4,S4],
         [C5,N5,P5,D5,S5]],

% The English man lives in the red house.
    member([red, 'English man', _, _, _], S),
```

```
% The Swede has a dog.
    member([_, 'Swede', dog, _, _], S),
% The Dane drinks tea.
    member([_, 'Dane', _, tea, _], S),
% The green house is on the left side of the white house.
    left_of([green |_], [white |_], S),
% In the green house, they drink coffee.
    member([green, _, _, coffee, _], S),
% The man who smokes cigarette has birds.
    member([_, _, birds, _, cigarette], S),
% In the yellow house, they smoke pipe.
    member([yellow, _, _, _, pipe], S),
% In the middle house, they drink milk.
    D3 = milk,
% The Norwegian lives in the first house.
    N1 = 'Norwegian',
% The man who smokes cigar lives in the house next to the house with
cats.
    next_to([_, _, _, _, cigar], [_, _, cats |_], S),
% In the house next to the house with the horse, they smoke pipe.
    next_to([_, _, _, _, pipe], [_, _, horse |_], S),
% The man who smokes hukka drinks soda.
    member([_, _, _, soda, hukka], S),
% The German smokes bedi.
    member([_, 'German', _, _, bedi], S),
% The Norwegian lives next to the blue house.
    next_to([_, 'Norwegian' |_], [blue |_], S),
% They drink water in the house that is next to the house
% where they smoke cigar.
    next_to([_, _, _, water,_], [_, _, _, _, cigar], S),

%
% The puzzle is so constrained that the following checks are not really
% required, but I include them for completeness (since one would not
% know in advance of solving the puzzle if it were fully constrained
% or not).
%
% Each house has its own unique color.
    C1 \== C2, C1 \== C3, C1 \== C4, C1 \== C5,
    C2 \== C3, C2 \== C4, C2 \== C5,
    C3 \== C4, C3 \== C5, C4 \== C5,
% All house owners are of different nationalities.
    N1 \== N2, N1 \== N3, N1 \== N4, N1 \== N5,
    N2 \== N3, N2 \== N4, N2 \== N5,
    N3 \== N4, N3 \== N5, N4 \== N5,
% They all have different pets.
    P1 \== P2, P1 \== P3, P1 \== P4, P1 \== P5,
    P2 \== P3, P2 \== P4, P2 \== P5,
    P3 \== P4, P3 \== P5, P4 \== P5,
% They all drink different drinks.
    D1 \== D2, D1 \== D3, D1 \== D4, D1 \== D5,
    D2 \== D3, D2 \== D4, D2 \== D5,
    D3 \== D4, D3 \== D5, D4 \== D5,
% They all smoke different cigarettes.
    S1 \== S2, S1 \== S3, S1 \== S4, S1 \== S5,
    S2 \== S3, S2 \== S4, S2 \== S5,
    S3 \== S4, S3 \== S5, S4 \== S5,

% Who owns the zebra?
    member([_, Who, zebra, _, _], S),
    write('The '), write(Who), write(' owns the zebra.\n').
```

```
% Or, replace the last line by:
%   format("The ~w owns the zebra.", Who).

left_of(L1, L2, [L1, L2 |_]).
left_of(L1, L2, [_| Rest ]) :- left_of(L1, L2, Rest).

next_to(L1, L2, S) :- left_of(L1, L2, S).
next_to(L1, L2, S) :- left_of(L2, L1, S).
```

Expert system

One of the main application domains for Prolog has been the development of expert systems. Following is an example of a simple medical expert system:

% clauses for relieves(Drug, Symptom). That is facts about drugs and relevant symptoms

relieves(aspirin, headache).
relieves(aspirin, moderate_pain).
relieves(aspirin, moderate_arthritis).
relieves(aspirin_codine_combination, severe_pain).
relieves(cough_cure, cough).
relieves(pain_gone, severe_pain).
relieves(anti_diarrhea, diarrhea).
relieves(de_congest, cough).
relieves(de_congest, nasal_congestion).
relieves(penicilline, pneumonia).
relieves(bis_cure, diarrhea).
relieves(bis_cure, nausea).
relieves(new_med, headache).
relieves(new_med, moderate_pain).
relieves(cong_plus, nasal_congestion).

% now we have clauses for side effects in the form of aggravates(Drug, Condition).

aggravates(aspirin, asthma).
aggravates(aspirin, peptic_ulcer).
aggravates(anti-diarrhea, fever).
aggravates(de_congest, high_blood_pressure).
aggravates(de_congest, heart_disease).
aggravates(de_congest, diabetes).
aggravates(de_congest, glaucoma).
aggravates(penicilline, asthma).
aggravates(de_congest, high_blood_pressure).
aggravates(bis_cure, diabetes).
aggravates(bis_cure, fever).

% now some rules for prescribing medicine using symptoms and side effects
should_take(Person, Drug) :-
        complains_of(Person, Symptom),
        relieves(Drug, Symptom),

```
        not(unsuitable_for(Person, Drug)).

unsuitable_for(Person, Drug) :-
        aggravates(Drug, Condition),
        suffers_from(Person, Condition).

% we now have some specific facts about a patient named Ali
complains_of(ali, headache).
suffers_from(ali, peptic_ulcer).


% now the query about the proper medicine and the answer by the expert system
?- should_take(ali, Drug).
Drug = new_med;
```

As mentioned earlier, Prolog has been used to write expert systems and expert system shells. Because of its inference mechanism and independence of rules and clauses it is relatively easy to achieve such task as compared to any imperative language.

Conclusions

Prolog is a declarative programming language where we only specify what we need rather than how to do it. It has a simple concise syntax with built-in tree formation and backtracking which generates readable code which is easy to maintain. It is relatively case and type insensitive and is suitable for problem solving / searching, expert systems / knowledge representation, language processing / parsing and NLP, and game playing. Efficiency is definitely is negative point. While writing programs one has to be aware of the left recursion, failing to do that may result in infinite loops. It has a steep learning curve and suffers from lack of standardization.

Java Programming Language (Lecture 27-30)

An Introduction

Java was developed at Sun in the early 1990s and is based on C++. It looks very similar to C++ but it is significantly simplified as compared to C++. That is why Professor Feldman says that Java is C++--. It supports only OOP and has eliminated multiple inheritance, pointers, structs, enum types, operator overloading, goto statement from C++. It includes support for applets and a form of concurrency.

The First Program

Here is the famous Hello World program in Java.

```
class HelloWorld {
        public static void main( String [ ] args )
 {
                System.out.println("Hello world!");
 }
}
```

It may be noted that as Java supports only OOP, in Java, every variable, constant, and function (including *main*) must be inside some class. Therefore, there are no global variables or functions in Java. In addition, the following may be noted:

- Function main is member of the class.
- Every class may have a main function; there may be more than one main function in a Java program.
- The main must be public static void.
- The main may have one argument: an array of String. This array contains the command-line arguments.
- There is no final semi-colon at the end of the class definition.

Java Files

In Java the source code is written in the .java file. The following restrictions apply:
(1) Each source file can contain at most one *public* class.
(2) If there is a public class, then the class name and file name must match.

Furthermore, every function must be part of a class and every class is part of a package. A *public* class can be used in *any* package and a non-public class can only be used in its own package.

The Java Bytecode is created by the Java compiler and is stored in .class file. This file contains ready to be exected Java bytecode which is executed by the Java Virtual Machine. For each class in a source file (both public and non-public classes), the compiler creates one ".class" file, where the file name is the same as the class name. When compiling a program, you type the full file name, including the ".java" extension; When running a program, you just type the name of the class whose *main* function you want to run.

Java Types

Java has two "categories" of types: *primitive types* and *reference* types.

Primitive Types
All the primitive types have specified sizes that are machine independent for portability. This includes:

boolean	same as bool in C++
char	holds one 16 bit unicode character
byte	8-bit signed integer
short	16-bit signed integer
int	32-bit signed integer
long	64-bit signed integer
float	floating-point number
double	double precision floating-point number

Reference Types
arrays classes

There are no struct, union, enum, unsigned, typedef, or pointers types in Java.

C++ Arrays vs Java Arrays

In C++, when you declare an array, storage for the array is allocated. In Java, when you declare an array, you are really only declaring a pointer to an array; storage for the array itself is not allocated until you use "new". This difference is elaborated as shown below:

C++

```
int A[10];          // A is an array of length 10
A[0] = 5;           // set the 1st element of array A
```

JAVA

```
int [ ] A;          // A is a reference to an array
A = new int [10];   // now A points to an array of length 10
A[0] = 5;           // set the 1st element of the array pointed to by A
```

In both C++ and Java you can initialize an array using values in curly braces. Here's example Java code:

```
int [ ] myArray = {13, 12, 11};
                  //  myArray points to an array of length 3
                  // containing the values 13, 12, and 11
```

In Java, a default initial value is assigned to each element of a newly allocated array if no initial value is specified. The default value depends on the type of the array element as shown below:

Type	**Value**
boolean	false
char	'\u0000'

byte, int, short, long, float, double	0
any reference	null

In Java, array bounds are checked and an out-of-bounds array index always causes a runtime error.

In Java, you can also determine the current length of an array (at runtime) using ".length" operator as shown below:

```
int [ ] A = new int[10];
...
A.length ...          // this expression evaluates to 10
A = new int[20];
...
A.length ...          // now it evaluates to 20
```

In Java, you can copy an array using the *arraycopy* function. Like the output function *println*, arraycopy is provided in java.lang.System, so you must use the name System.arraycopy. The function has five parameters:

src: the source array (the array from which to copy)
srcPos: the starting position in the source array
dst: the destination array (the array into which to copy)
dstPos: the starting position in the destination array
count: how many values to copy

Here is an example:

```
int [ ] A, B;
A = new int[10];

// code to put values into A
B = new int[5];
System.arraycopy(A, 0, B, 0, 5)     // copies first 5 values from A to B
System.arraycopy(A, 9, B, 4, 1)     // copies last value from A into last
                                    // element of B
```

Note that the destination array must already exist (i.e., *new* must already have been used to allocate space for that array), and it must be large enough to hold all copied values (otherwise you get a runtime error). Furthermore, the source array must have enough values to copy (i.e., the length of the source array must be at least srcPos+count). Also, for arrays of primitive types, the types of the source and destination arrays must be the same. For arrays of non-primitive types, System.arraycopy(A, j, B, k, n) is OK if the assignment B[0] = A[0] would be OK.

The arraycopy function also works when the source and destination arrays are the *same* array; so for example, you can use it to "shift" the values in an array:

```
int [ ] A = {0, 1, 2, 3, 4};
System.arraycopy(A, 0, A, 1, 4);
```

After executing this code, A has the values: [0, 0, 1, 2, 3].

As in C++, Java arrays can be *multidimensional*. For example, a 2-dimensional array is an array of arrays. However, a two-dimensional arrays need not be rectangular. Each row can be a different length. Here's an example:

```
int [ ][ ] A;          // A is a two-dimensional array
A = new int[5][];      // A now has 5 rows, but no columns yet
A[0] = new int [1];    // A's first row has 1 column
A[1] = new int [2];    // A's second row has 2 columns
A[2] = new int [3];    // A's third row has 3 columns
A[3] = new int [5];    // A's fourth row has 5 columns
A[4] = new int [5];    // A's fifth row also has 5 columns
```

C++ Classes vs Java Classes

In C++, when you declare a variable whose type is a class, storage is allocated for an object of that class, and the class's constructor function is called to initialize that instance of the class. In Java, you are really declaring a pointer to a class object; no storage is allocated for the class object, and no constructor function is called until you use "new".

This is elaborated with the help of the following example:

Let us assume that we have the following class:

```
class MyClass {
        ...
    }
```

Let us first look at C++

```
MyClass m;    // m is an object of type MyClass;
              //  the constructor function is called to initialize M.

MyClass  *pm;
              // pm is a pointer to an object of type MyClass
              // no object exists yet, no constructor function
              // has been called

pm = new MyClass;
              // now storage for an object of MyClass has been
              // allocated and the constructor function has been
              // called
```

Now the same thing in Java

```
MyClass m;    // pm is a pointer to an object of type MyClass
              // no object exists yet, no constructor function
              // has been called
m = new MyClass();
```

```
// now storage for an object of MyClass has been allocated
// and the constructor function has been called. Note
// that you must use parentheses even when you are not
// passing any arguments to the constructor function

// Also note that there is a simple '.' (dot) operator used to
// access members or send message. Java does not use
// -> operator.
```

Whereas, as opposed to Java, in C++ use have the following:

```
MyClass m;          // m is an object of type MyClass;
                    // the constructor function is called to initialize M.

MyClass  *pm;       // pm is a pointer to an object of type MyClass
                    // no object exists yet, no constructor function has
                    // been called

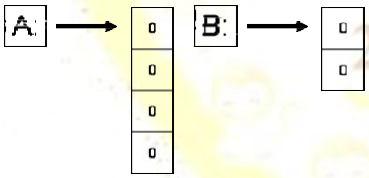
pm = new MyClass; // now storage for an object of MyClass has been
                    // allocated and the constructor function has been
                    // called
```

Aliasing Problems in Java

The fact that arrays and classes are really pointers in Java can lead to some problems. Here is a simple assignment that causes *aliasing*:

```
int [] A = new int [4];
Int [] B = new int [2];
```

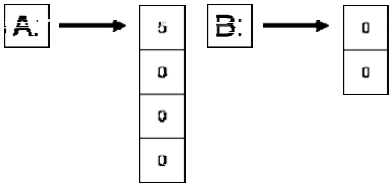
This is depicted as below:



Now, when we say:

```
A[0] = 5;
```

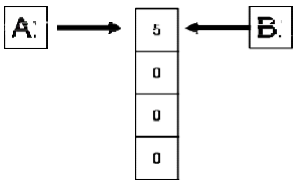
We get the following:



Now when we say:

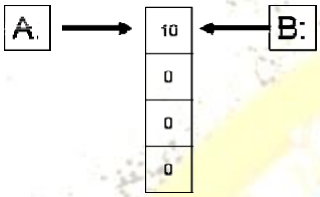
```
B = A;
```

B points to the same array as A and creates an alias. This is shown below:



Now if we make a simple assignment in B, we will also change A as shown below:

```
B[0] = 10;
```



This obviously creates problems. Therefore, as a programmer you have to be very careful when writing programs in Java.

In Java, all parameters are passed by value, but for arrays and classes the actual parameter is really a pointer, so changing an array element, or a class field inside the function *does* change the actual parameter's element or field.

This is elaborated with the help of the following example:

```
void f( int [ ] A ) {
        A[0] = 10;     // change an element of parameter A
        A = null;  // change A itself
}

void g() {
        int [ ] B = new int [3];
        B[0] = 5;
        f(B);
                // B is not null here, because B itself was passed by value
                // however, B[0] is now 10, because function f changed the
                // first element of the array
}
```

Note that, in C++, similar problems can arise when a class that has pointer data members is passed by value. This problem is addressed by the use of copy constructors, which can be defined to make copies of the values pointed to, rather than just making copies of the pointers. In Java, the solution is to use the *arraycopy* operation, or to use a class's *clone* operation. Cloning will be discussed later.

Type Conversion

Java is much stronger than C++ in the type conversions that are allowed.

Here we discuss conversions among primitive types. Conversions among class objects will be discussed later.

Booleans cannot be converted to other types. For the other primitive types (char, byte, short, int, long, float, and double), there are two kinds of conversion: *implicit* and *explicit*.

Implicit conversions:

An implicit conversion means that a value of one type is changed to a value of another type without any special directive from the programmer. A char can be implicitly converted to an int, a long, a float, or a double. For example, the following will compile without error:

```
char c = 'a';
int k = c;
long x = c;
float y = c;
double d = c;
```

For the other (numeric) primitive types, the basic rule is that implicit conversions can be done from one type to another if the range of values of the first type is a subset of the range of values of the second type. For example, a byte can be converted to a short, int, long or float; a short can be converted to an int, long, float, or double, etc.

Explicit conversions:

Explicit conversions are done via *casting*: the name of the type to which you want a value converted is given, in parentheses, in front of the value. For example, the following code uses casts to convert a value of type double to a value of type int, and to convert a value of type double to a value of type short:

```
double d = 5.6; int k = (int)d; short s = (short)(d * 2.0);
```

Casting can be used to convert among any of the primitive types except boolean. Note, however, that casting can lose information; for example, floating-point values are truncated when they are cast to integers (e.g., the value of k in the code fragment given above is 5), and casting among integer types can produce wildly different values (because upper bits, possibly including the sign bit, are lost).

JAVA CLASSES

Java classes contain *fields* and *methods*. A field is like a C++ data member, and a method is like a C++ member function. Each field and method has an *access level*:

- private: accessible only in this class
- (package): accessible only in this package
- protected: accessible only in this package and in all subclasses of this class
- public: accessible everywhere this class is available

Similarly, each class has one of two possible access levels:

- (package): class objects can only be declared and manipulated by code in this package
- public: class objects can be declared and manipulated by code in any package

Note: for both fields and classes, package access is the default, and is used when *no* access is specified.

A Simple Example Class

In the following example, a List is defined to be an ordered collection of items of any type:

```
class List {
        // fields
        private Object [ ] items;       // store the items in an array
        private int   numItems;         // the current # of items in the list
                                        // methods

        // constructor function
        public List()
        {
                items = new Object[10];
                numItems = 0;
        }

        // AddToEnd: add a given item to the end of the list
        public void AddToEnd(Object ob)
        {
        ...
        }
}
```

In Java, all classes (built-in or user-defined) are (implicitly) subclasses of the class **Object**. Using an array of Object in the List class allows any kind of Object (an instance of any class) to be stored in the list. However, primitive types (int, char, etc) cannot be stored in the list as they are not inherited from Object.

Constructor function:

As in C++, constructor functions in Java are used to initialize each instance of a class. They have no return type (not even void) and can be overloaded; you can have multiple constructor functions, each with different numbers and/or types of arguments. If you don't write any constructor functions, a default (no-argument) constructor (that doesn't do anything) will be supplied.

If you write a constructor that takes one or more arguments, no default constructor will be supplied (so an attempt to create a new object without passing any arguments will cause a compile-time error). It is often useful to have one constructor call another (for example, a constructor with no arguments might call a constructor with one argument, passing a

default value). The call must be the *first* statement in the constructor. It is performed using *this* as if it were the name of the method. For example:

```
this( 10 );
```

is a call to a constructor that expects one integer argument.

Initialization of fields:

If you don't initialize a field (i.e., either you don't write any constructor function, or your constructor function just doesn't assign a value to that field), the field will be given a default value, depending on its type. The values are the same as those used to initialize newly created arrays (see the "Java vs C++" notes).

Access Control:

Note that the access control must be specified for every field and every method; there is no grouping as in C++. For example, given these declarations:

```
public
    int x;
    int y;
```

only x is public; y gets the default, package access.

Static Fields and Methods

Fields and methods can be declared *static*. If a field is static, there is only one copy for the entire class, rather than one copy for each instance of the class. (In fact, there is a copy of the field even if there are *no* instances of the class.) A method should be made static when it does not access any of the non-static fields of the class, and does not call any non-static methods. (In fact, a static method *cannot* access non-static fields or call non-static methods.) Methods that would be "free" functions in C++ (i.e., not members of any class) should be static methods in Java. A public static field or method can be accessed from outside the class.

Final Fields and Methods

Fields and methods can also be declared *final*. A final method cannot be overridden in a subclass. A final field is like a constant: once it has been given a value, it cannot be assigned to again.

Some Useful Built-in Classes

Following is a list of some useful classes in Java. These classes are not really part of the language; they are provided in the package *java.lang*

1. String

String Creation:

```
String S1 = "hello",          // initialize from a string literal
S2 = new String("bye"),       // use new and the String constructor
S3 = new String(S1);          // use new and a different constructor
```

String concatenation

```
String S1 = “hello ” + “world”;
String S2 = S1 + “!”;
String S3 = S1 + 10;
```

2. Object

Object is the Base class for all Java Classes.

3. Classes for primitive types

In Java, we have classes also for primitive types. These are: Boolean, Integer, Double, etc. Note that variable of bool, int, etc types are not objects whereas variable of type Boolean, Integer, etc are objects.

Packages

Every class in Java is part of some *package*. All classes in a file are part of the same package. You can specify the package using a *package declaration*:

package *name* ;

as the first (non-comment) line in the file. Multiple files can specify the same package name. If no package is specified, the classes in the file go into a special unnamed package (the same unnamed package for all files). If package *name* is specified, the file must be in a subdirectory called *name* (i.e., the directory name must match the package name).

You can access public classes in another (named) package using:

package-name.class-name

You can access the public fields and methods of such classes using:

package-name.class-name.field-or-method-name

You can avoid having to include the *package-name* using import *package-name*.* or import *package-name.class-name* at the beginning of the file (after the package declaration). The former imports all of the classes in the package, and the second imports just the named class.
You must still use the *class-name* to access the classes in the packages, and *class-name.field-or-method-name* to access the fields and methods of the class; the only thing you can leave off is the package name.

Inheritance

Java directly supports single inheritance. To support concept of a **interface** class is used. Inheritance is achieved as shown below:

```
class SuperClass {
        …
}

class SubClass extends SuperClass {
        …
}
```

When a class is inherited, all fields and methods are inherited. When the **final** reserved word is specified on a class specification, it means that class cannot be the parent of any class. *Private* fields are inherited, but cannot be accessed by the methods of the subclass. fields can be defined to be *protected*, which means that subclass methods can access them

Each superclass method (except its constructors) can be inherited, overloaded, or overridden. These are elaborated in the following paragraphs:

inherited: If no method with the same name is (re)defined in the subclass, then the subclass has that method with the same implementation as in the superclass.

overloaded: If the subclass defines a method with the same name, but with a different number of arguments or different argument types, then the subclass has *two* methods with that name: the old one defined by the superclass, and the new one it defined.

overridden: If the subclass defines a method with the same name, and the same number and types of arguments, then the subclass has only *one* method with that name: the new one it defined. A method cannot be overridden if it is defined as final in the superclass

Dynamic Binding

In C++, a method must be defined to be virtual to allow dynamic binding. In Java all method calls are dynamically bound unless the called method has been defined to be **final**, in which case it cannot be overridden and all bindings are static.

Constructors

A subclass's constructors *always* call a super class constructor, either explicitly or implicitly. If there is no explicit call (and no call to another of the subclass constructors), then the no-argument version of the superclass constructor is called before executing any statements.

Abstract Classes

An *abstract method* is a method that is declared in a class, but not defined. In order to be instantiated, a subclass must provide the definition. An *abstract class* is any class that includes an abstract method. It is similar to Pure virtual in C++.

If a class includes an abstract method, the class *must* be declared abstract, too. For example:

```
abstract class AbstractClass {
        abstract public void Print();
                // no body, just the function header
}
class MyConcreteClass extends AbstractClass {
        public void Print() { // actual code goes here } }
```

An abstract class cannot be instantiated. A subclass of an abstract class that does not provide bodies for all abstract methods must also be declared abstract. A subclass of a *non*-abstract class can override a (non-abstract) method of its superclass, and declare it abstract. In that case, the subclass must be declared abstract.

Interface

As mentioned earlier, multiple inheritance is achieved through Interface in Java. Inheritance implements the "is-a" relationship whereas an interface is similar to a class, but can only contain public, static, final fields (i.e., constants) and public, abstract methods (i.e., just method headers, no bodies).

An interface is declared as shown below:

```
public interface Employee {
        void RaiseSalary( double d );
        double GetSalary();
}
```

Note that both methods are implicitly public and abstract (those keywords can be provided, but are not necessary).

A class can *implement* one or more interfaces (in addition to extending one class). It must provide bodies for all of the methods declared in the interface, or else it must be abstract. For example:

```
public class TA implements Employee {
        void RaiseSalary( double d ) {
                // actual code here
        }
        double GetSalary() {
                // actual code here
        }
}
```

Public interfaces (like public classes) must be in a file with the same name. Many classes can implement the same interface thus achieving multiple inheritance. An interface can be a "marker" with *no* fields or methods, is used only to "mark" a class as having a property, and is testable via the *instanceof* operator.

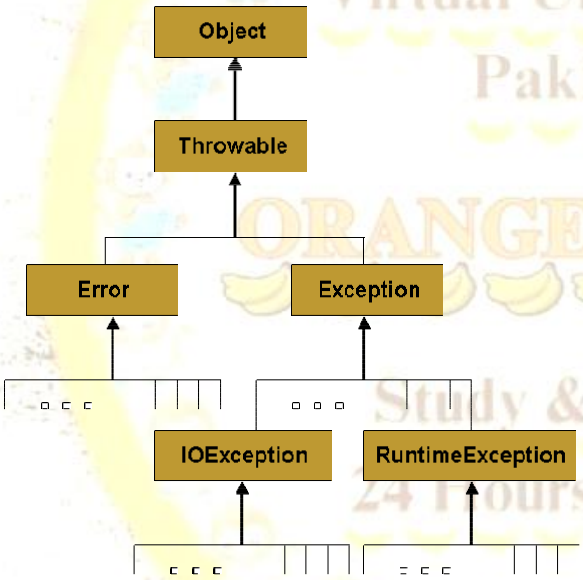
Exception Handling in Java

Exception handling in Java is based on C++ but is designed to be more in line with OOP. It includes a collection of predefined exceptions that are implicitly raised by the JVM. All java exceptions are objects of classes that are descendents of Throwable class. There are two predefined subclasses of Throwable: Error and Exception.

Error and its descendents are related to errors thrown by JVM. Examples include out of heap memory. Such an exception is never thrown by the user programs and should not be handled by the user.

User programs can define their own exception classes. Convention in Java is that such classes are subclasses of Exception. There are two predefined descendents of Exception: IOException and RuntimeException. IOException deals with errors in I/O operation.
In the case of RuntimeException there are some predefined exceptions which are, in many cases, thrown by JVM for errors such as out of bounds, and Null pointer.

The following diagram shows the exception hierarchy in Java.



Checked and Unchecked Exceptions

Exceptions of class Error and RuntimeException are called unchecked exceptions. They are never a concern of the compiler. A program can catch unchecked exceptions but it is not required. All other are checked exceptions. Compiler ensures that all the checked exceptions a method can throw are either listed in its throws clause or are handled in the method.

Exception Handlers
Exception handler in Java is similar to C++ except that the parameter of every **catch** must be present and its class must be descendent of Thrwoable. The syntax of **try** is exactly same as C++ except that there is **finally** clause as well. For example:

```
class MyException extends Exception {
        public MyException() { }
        public MyException(String message) {
                super (message);
        }
}
```

This exception can be thrown with

```
throw new MyException();
```

Or

```
MyException myExceptionObject = new MyException();
…
throw myExceptionObject;
```

Binding of exception is also similar to C++. If an exception is thrown in the compound statement of try construct, it is bound to the first handler (catch function) immediately following the try clause whose parameter is the same class as the thrown object or an ancestor of it. Exceptions can be handled and then re-thrown by including a throw statement without an operand at the end of the handler. To ensure that exceptions that can be thrown in a try clause are always handled in a method, a special handler can be written that matches all exceptions. For example:

```
catch (Exception anyException) {
        …
}
```

Other Design Choices

The exception handler parameter in C++ has a limited purpose. During program execution, the Java runtime system stores the class name of every object. getClass can be used to get an object that stores the class name. It has a getName method. The name of the class of the actual parameter of the throw statement can be retrieved in the handler as shown below.

```
anyException.getClass().getName();
```

In the case of a user defined exception, the thrown object could include any number of data fields that might be useful in the handler.

throws Clause

throws clause is overloaded in C++ and conveys two different meanings: one as specification and the other as command. Java is similar in syntax but different in semantics. The appearance of an exception class name in the **throws** clause of Java method specifies that the exception class or any of its descendents can be thrown by the method.

A C++ program unit that does not include a throw clause can throw any exceptions. A Java method that does not include a throws cannot throw any checked exception it does

not handle. A method cannot declare more exceptions in its throws clause than the methods it overrides, though it may declare fewer. A method that does not throw a particular exception, but calls another method that could throw the exception, must list the exception in its throws clause.

The finally clause

A Java exception handler may have a finally clause. A finally clause *always* executes when its try block executes (whether or not there is an exception). The finally clause is written as shown below:

```
try {
        …
}
catch (…) {
        …
}
…
finally {
        …
}
```

A finally clause is usually included to make sure that some clean-up (e.g., closing opened files) is done. If the finally clause includes a transfer of control statement (return, break, continue, throw) then that statement overrides any transfer of control initiated in the try or in a catch clause.

First, let's assume that the finally clause does not include any transfer of control. Here are the situations that can arise:

- No exception occurs during execution of the try, and no transfer of control is executed in the try. In this case the finally clause executes, then the statement following the try block.
- No exception occurs during execution of the try, but it *does* execute a transfer of control. In this case the finally clause executes, then the transfer of control takes place.
- An exception *does* occur during execution of the try, and there is no catch clause for that exception. Now the finally clause executes, then the uncaught exception is "passed up" to the next enclosing try block, possibly in a calling function.
- An exception *does* occur during execution of the try, and there *is* a catch clause for that exception. The catch clause does not execute a transfer of control. In this case the catch clause executes, then the finally clause, then the statement following the try block.
- An exception *does* occur during execution of the try, there *is* a catch clause for that exception, and the catch clause *does* execute a transfer of control. Here, the catch clause executes, then the finally clause, then the transfer of control takes place.

If the finally block *does* include a transfer of control, then that takes precedence over any transfer of control executed in the try or in an executed catch clause. So for all of the

cases listed above, the finally clause would execute, then *its* transfer of control would take place. Here's one example:

```
try {
        return 0;
} finally {
        return 2;
}
```

The result of executing this code is that 2 is returned. Note that this is rather confusing! The moral is that you probably do *not* want to include transfer-of-control statements in both the try statements and the finally clause, or in both a catch clause and the finally clause.

Java Threads

Java supports concurrency through threads which are lightweight processes. A thread is similar to a real process in that a thread and a running program are threads of execution. A thread takes advantage of the resources allocated for that program instead of having to allocate those resources again. A thread has its own stack and program counter. The code running within the thread works only within the context implied by the stack and PC so also called an execution context.

Creating Java Threads

There are two ways to create our own **Thread** object:

1. Subclassing the **Thread** class and instantiating a new object of that class
2. Implementing the **Runnable** interface

In both cases the **run()** method should be implemented. This is elaborated with the help of following examples:

Example of Extending Thread

```
public class ThreadExample extends Thread {
        public void run () {
        for (int i = 1; i <= 100; i++) {
                System.out.println("Thread: " + i);
    }
  }
}
```

Example of Implementing Runnable

```
public class RunnableExample implements Runnable {
        public void run () {
                for (int i = 1; i <= 100; i++) {
                System.out.println ("Runnable: " + i);
        }
```

```
        }
    }
```

It may be noted that both of these are very similar to each other with minor syntactic and semantic differences.

Starting the Threads

A thread is started by simply sending the **start** message to the thread. This is shown in the following example:

```
public class ThreadsStartExample {
    public static void main (String argv[]) {
        new ThreadExample ().start ();
            new Thread(new RunnableExample ()).start ();
    }
}
```

Thread Methods

Some of the common thread methods are listed below:

- start()
- sleep()
- yield()
- run()
- wait()
- notify()
- notifyAll()
- setPriority()

Thread Synchronization - Wait and Notify

Threads are based upon the concept of a Monitor. The **wait** and **notify** methods are used just like **wait** and **signal** in a Monitor. They allow two threads to cooperate and based on a single shared lock object. The following example of a producer-consumer problem elaborates this concept.

```
class Buffer {
        private int [ ] buf;
        private int head, tail, max, size;

        public Buffer (int buf_size) {
                buf = new int [buf_size];
                head =0; tail = 0; size = 0;
                max = buf_size;
        }

        public synchronized void deposit (int item) {
                try {
```

```
                        while (size ==max) wait();
                        buf [tail] = item;
                        tail = (tail + 1) % max;
                        size++;
                        notify();
                }
                catch
                (interruptedException e) {
                        // exception handler
                }
        }

        public synchronized int fetch() {
                int item = 0;
                try {
                        while (size == 0) wait();
                        item = buf [head] ;
                        head = (head + 1) % max;
                        size--;
                        notify();
                }
                catch
                   (interruptedException e) {
                        // exception handler
                }
                return item;
        }
class Producer extends Thread {
        private Buffer buffer;


        public Producer (Buffer buf) {
                buffer = buf;
        }

        public void run () {
                int item;
                while (true) {
                    // create a new item
                    buffer.deposit(item);
                }
        }
}

class Consumer implements Runnable {
        private Buffer buffer;

        public Consumer (Buffer buf) {
                buffer = buf;
        }
```

```
        public void run () {
            int item;
            while (true) {
                item = buffer.fetch();
                // consume item
            }
        }
    }
```

We can now instantiate these threads as shown below:

```
Buffer buffer = new Buffer(100);
Producer producer1 = new Producer(buffer);
Consumer comsumer = new Consumer(buffer);
Thread consumer1 = new Thread(consumer);
producer1.start();
consumer1.start();
```

In this case the **buffer** is a shared variable used by both producer and consumer.

notify and notifyAll

There is a slight difference between **notify** and **notifyAll**. As the name suggest, **notify()** wakes up a single thread which is waiting on the object's lock. If there is more than one thread waiting, the choice is arbitrary i.e. there is no way to specify which waiting thread should be re-awakened. On the other hand, **notifyAll()** wakes up ALL waiting threads; the scheduler decides which one will run.

Applet

Java also has support for web applications through Applets. An applet is a stand alone Java *application.* Java *applet* runs in a Java-enabled Web browser.

Java versus C++

Generally, Java is more robust than C++. Some of the reasons are:

- Object handles are initialized to **null** (a keyword)
- Handles are always checked and exceptions are thrown for failures
- All array accesses are checked for bounds violations
- Automatic garbage collection prevents memory leaks and dangling pointers
- Type conversion is safer
- Clean, relatively fool-proof exception handling
- Simple language support for multithreading
- Bytecode verification of network applets

C# Programming Language (Lecture 31-34)
An Introduction

C# was released by Microsoft in June 2000 as part of the .Net framework. It was co-authored by Anders Hejlsberg (who is famous for the design of the Delphi language), and Scott Wiltamuth. C# is a strongly-typed object-oriented language. It is Similar to Java and C++ in many respects. The .NET platform is centered on a Common Language Runtime (CLR - which is similar to a JVM) and a set of libraries which can be exploited by a wide variety of languages which are able to work together by all compiling to an intermediate language (IL).

Java and C# - Some Commonalities

Java and C# are very similar and have a number of commonalities.

a. Both of these languages compile into machine-independent language-independent code which runs in a managed execution environment.
b. Both C# and Java compile initially to an intermediate language: C# to Microsoft Intermediate Language (MSIL), and Java to Java bytecode. In each case the intermediate language can be run - by interpretation or just-in-time compilation - on an appropriate 'virtual machine'. In C#, however, more support is given for the further compilation of the intermediate language code into native code.
c. Both of these have garbage Collection coupled with the elimination of pointers (in C# restricted use is permitted within code marked unsafe).
d. Both have powerful reflection capabilities.
e. In case of both these languages, there is no header files, all code scoped to packages or assemblies, no problems declaring one class before another with circular dependencies.
f. Both of these support only OOP and classes all descend from object and must be allocated on the heap with new keyword.
g. Both support concurrency through thread support by putting a lock on objects when entering code marked as locked/synchronized.
h. Both have single inheritance and support for interfaces.
i. There are no global functions or constants, everything belongs to a class.
j. Arrays and strings with built-in bounds checking.
k. The "." operator is always used and there are no more ->, :: operators.
l. null and boolean/bool are keywords.
m. All values must be initialized before use.
n. Integers expressions cannot be used in ‘if’ statements and conditions in the loops.
o. In both languages try Blocks can have a finally clause.

Some C# features which are different from Java

a. C# has more primitive data types than Java and has more extension to the value types.
b. It supports safer enumeration types whereas Java does not have enumeration types.
c. It also has support for struct which are light weight objects.
d. There is support for operator overloading.
e. C# has the concept of 'delegates' which are type-safe method pointers and are used to implement event-handling.

f. It supports three types of arrays: single dimensional, multi-dimensional rectangular and multi-dimensional jagged arrays.
g. It has restricted use of pointers. The 'switch' statements in C# have been changed so that 'fall-through' behavior is disallowed.
h. It also has support for class 'properties'.

C# Hello World

Let us have a look at the Hello World Program in C#.

```
using System;                    // System namespace
public class HelloWorld
{
   public static void Main()              // the Main function starts
                                                  // with capital M
   {
            Console.WriteLine("Hello World!”);
    }
}
```

In this example, it may be noted that just like Java and C++, C# is case sensitive. As we have in Java, everything in C# has to be inside a class and there is no semicolon at the end of the class. However, unlike Java, name of the class and the name of the file in which it is saved do not need to match up. You are free to choose any extension for the file, but it is usual to use the extension '.cs'. It supports single line and multiple line comments.

Variable Types: Reference Types and Value Types

As mentioned earlier, C# is a type-safe language. Variables can hold either value types or reference types, or they can be pointers. When a variable v contains a value type, it directly contains an object with some value and when it contains a reference type, what it directly contains is something which refers to an object.

Built-in Types

The following table lists the built-in C# types and their ranges.

C# Type	.Net Framework (System) type	Signed?	Bytes Occupied	Possible Values
sbyte	System.Sbyte	Yes	1	-128 to 127
short	System.Int16	Yes	2	-32768 to 32767
int	System.Int32	Yes	4	-2147483648 to 2147483647
long	System.Int64	Yes	8	-9223372036854775808 to 9223372036854775807
byte	System.Byte	No	1	0 to 255

ushort	System.Uint16	No	2	0 to 65535
uint	System.UInt32	No	4	0 to 4294967295
ulong	System.Uint64	No	8	0 to 18446744073709551615
float	System.Single	Yes	4	Approximately ±1.5 x 10-45 to ±3.4 x 1038 with 7 significant figures
double	System.Double	Yes	8	Approximately ±5.0 x 10-324 to ±1.7 x 10308 with 15 or 16 significant figures
decimal	System.Decimal	Yes	12	Approximately ±1.0 x 10-28 to ±7.9 x 1028 with 28 or 29 significant figures
char	System.Char	N/A	2	Any Unicode character (16 bit)
bool	System.Boolean	N/A	1 / 2	true or false

Structs

1. Java did not have structs which have been brought back by C#. However, as compared to C++, they are significantly different in C#.
2. In C++ a struct is exactly like a class, except that the default inheritance and default access are public rather than private.
3. In C# structs are very different from classes.
4. Structs in C# are designed to encapsulate lightweight objects.
5. They are value types (not reference types), so they're passed by value.
6. They are sealed, which means they cannot be derived from or have any base class other than System.ValueType, which is derived from Object.
7. Structs cannot declare a default (parameterless) constructor.
8. Structs are more efficient than classes, that's' why they are perfect for the creation of lightweight objects.

Properties

C# has formalized the concept of getter/setter methods. The relationship between a get and set method is inherent in C#, while has to be maintained in Java or C++. For example, in Java/C++ we will have to write code similar to the one shown below:

```
public int getSize() {
        return size;
}

public void setSize (int value) {
        size = value;
}

foo.setSize (getSize () + 1);
```

In C# we can define a property and can use it as if we were using a public variable. This is shown below:

```
public int Size {
        get {return size; }
        set {size = value; }
}

foo.size = foo.size + 1;
```

Reference types

In C#, the pre-defined reference types are object and string. As mentioned earlier, object is the ultimate base class of all other types. New reference types can be defined using 'class', 'interface', and 'delegate' declarations. Reference types actually hold the value of a memory address occupied by the object they reference.

Reference types however suffer from the problem of aliasing as shown below:

```
object x = new object();
x.myValue = 10;
object y = x;
y.myValue = 20;        // after this statement both x.myValue and y.myValue
                       // equal 20
```

There is however no aliasing in string. That is, strings are immutable. The properties of an immutable object can't be modified. So in order to change what a string variable references, a new string object must be created. Following is an example that elaborates this concept:

```
string s1 = "hello";
string s2 = s1; // s2 points to the same strings as s1
s1 = “world”;          // a new string is created and s1 points to it.
                       // s2 keeps pointing to the old strings
```

----------END OF LECTURE 31----------

Pointers

Pointers were present in C++ but Java designers took them out. C# has brought them back. However, C#, pointers can only be declared to hold the memory addresses of value types. That is, we cannot have pointers for reference types. The rest is very similar to C++. This is illustrated with the help of the following example:

```
int i = 5;
int *p;
p = &i;          // take address of i
*p = 10;         // changes the value of i to 10
```

One major difference between C++ and C# is that the '*' applies to the type. That is, as opposed to C++, in C#, the following statement would declare two pointers p1 and p2:

```
int * p1, p2;
```

Just like C++, the dereference operator '->' is used to access elements of a struct type.

Pointers and unsafe code

In C#, we have two modes for the code, the managed and unmanaged. Theses are elaborated as below:

Managed code

1. Managed code is executed under the control of Common Language Runtime (CRL).
2. It has automatic garbage collection. That is, the dynamically allocated memory area which is no longer is in use is not destroyed by the programmer explicitly. It is rather automatically returned back to heap by the built-in garbage collector.
3. There is no explicit memory's allocation and deallocation and there is no explicit calls to the garbage collector i.e. call to destructor.

Unmanaged code (Java does not have this concept**)**

The unmanaged code provides access to memory through pointers just like C++. It is useful in many scenarios. For example:

- Pointers may be used to enhance performance in real time applications.
- **External Functions:** In non-.net DLLs some external functions requires a pointer as a parameter, such as Windows APIs that were written in C.
- **Debugging:** Sometimes we need to inspect the memory contents for debugging purposes, or you might need to write an application that analyzes another application process and memory.

unsafe

The keyword unsafe is used while dealing with pointer. The name reflects the risks that you might face while using it. We can declare a whole class as unsafe as shown below:

```
unsafe class Class1 {
        //you can use pointers here!
}
```

Or only some class members can be declared as unsafe:

```
class Class1 {
        //pointer
        unsafe int * ptr;
        unsafe void MyMethod() {
                //you can use pointers here
        }
}
```

To declare unsafe local variables in a method, you have to put them in unsafe blocks as the following:

```
static void Main() {
        //can't use pointers here
        unsafe
        {
                //you can declare and use pointer here
        }
        //can't use pointers here
}
```

Disadvantages of using unsafe code in C#:

- Complex code
- Harder to use
- Pointers are harder to debug
- You may compromise type safety
- Misusing might lead to the followings:
    - Overwrite other variables
    - Illegal access to memory areas not under your control
    - Stack overflow

Garbage Collector Problem in C# and the fixed keyword

When pointers are used in C#, Garbage collector can change physical position of the objects. If garbage collector changes position of an object the pointer will point at wrong place in memory. To avoid such problems C# contains '*fixed*' keyword – informing the system not to relocate an object by the garbage collector.

Arrays

Arrays in C# are more similar to Java than to C++. We can create an array as did in Java by using the new operator. Once created, the array can be used as usual as shown below:

```
int[] i = new int[2];
i[0] = 1;
i[1] = 2;
```

By default all arrays start with their lower bound as 0. Using the .NET framework's System.Array class it is possible to create and manipulate arrays with an alternative initial lower bound.

Types of Arrays:

- Single Dimensional Arrays
- Multidimensional arrays
  - Rectangular
  - Jagged

Rectangular Arrays:

A rectangular array is a single array with more than one dimension, with the dimensions' sizes fixed in the array's declaration. Here is an example:

```
int[,] squareArray = new int[2,3];
```

As with single-dimensional arrays, rectangular arrays can be filled at the time they are declared.

Jagged Arrays (Similar to Java jagged arrays**)**

Jagged arrays are multidimensional arrays with irregular dimensions. This flexibility derives from the fact that multidimensional arrays are implemented as arrays of arrays.

```
int[][] jag = new int[2][];
jag[0] = new int [4];
jag[1] = new int [6];
```

Each one of jag[0] and jag[1] holds a reference to a single-dimensional int array.

Pointers and Arrays

Just as in C++, a pointer can be declared in relation to an array:

```
int[] a = {4, 5};
int *b = a;
```

In the above example memory location held by b is the location of the first type held by a. This first type must, as before, be a value type. If it is reference type then compiler will give error.

Enumerations

C# brought back enumerations which were discarded by Java designers. However, they are slightly different from C++.

- An enumeration is a special kind of value type limited to a restricted and unchangeable set of numerical values.
- By default, these numerical values are integers, but they can also be longs, bytes, etc. (any numerical value except char) as will be illustrated below.
- Type safe
- Consider the following example:

```
public enum DAYS {
        Monday=1,
        Tuesday,
        Wednesday,
        Thursday,
        Friday,
        Saturday,
        Sunday
}
```

In C# enumerations are type-safe, by which we mean that the compiler will do its best to stop you assigning illicit values to enumeration typed variables. For instance, the following code should not compile:

```
int i = DAYS.Monday;
DAYS d = i;
```

In order to get this code to compile, you would have to make explicit casts both ways (even converting from DAYS to int), ie:

```
int i = (int)DAYS.Monday;
DAYS d = (DAYS)i;
```

A useful feature of enumerations is that one can retrieve the literal as a string from the numeric constant with which it is associated. In fact, this is given by the default ToString() method, so the following expression comes out as true:

```
DAYS.Monday.ToString()=="Monday"
```

END OF LECTURE 32

Boolean

- In C#, Boolean values do not convert to integers and Boolean values (true, false) do not equate to integer variables. Thus, you may not write:

```
if ( someFuncWhichReturnsAnIntegerValue() )
```

- No arithmetic expression is allowed where Boolean expressions are expected. Thus, if you write:

```
if (x = y)     // compilation error
```

It will not be compiled.

Operator evaluation order

As discussed earlier, in C/C++ the operator evaluation order is not specified. We have discussed at length how that creates problems. In C#, it is strictly from left to right and hence there is no ambiguity.

For example,

```
a=5;
x=a++ + --a;
```

If evaluated left-to-right

```
x=10
```

If evaluated right-to-left

```
x=8
```

Conversion

Like Java supports implicit widening conversion only. For narrowing conversion, fro example from float to int, the programmer has to explicitly state his intentions.

Loops

C# provides a number of the common loop statements. These include while, do-while, for, and foreach.

The syntax of while, do-while, and for loops is similar to C++. The only difference is that the loop control expression must be of Boolean type.

foreach loop

The 'foreach' loop is used to iterate through the values contained by any object which implements the IEnumerable interface. It has the following syntax:

```
foreach (variable1 in variable2) statement[s]
```

When a 'foreach' loop runs, the given variable1 is set in turn to each value exposed by the object named by variable2. Here is an example:

```
int[] a = new int[]{1,2,3};
foreach (int b in a)
System.Console.WriteLine(b);
```

Other Control Flow Statements

C# supports a number of control statements including break, continue, goto, if, switch, return, and throw.

switch statement

They are more or less similar to their counterparts in C++. The switch statement is however significantly different is explained below. The syntax of the switch statement is given below:

```
switch (expression)
{
        case constant-expression:
        statements
        jump statement
        [default:
                statements
                jump statement
        ]
}
```

The expression can be an integral or string expression. Control does not fall through. Jump statement is required for each block – even in default block. Fall through is allowed to stack case labels as shown in the following example:

```
switch(a){
case 2:
```

```
            Console.WriteLine("a>1 and ");
            goto case 1;
case 1:
            Console.WriteLine("a>0");
            break;
default:
            Console.WriteLine("a is not set");
            break;
}
```

As mentioned earlier, we can also use a string in the switch expression. This is demonstrated with the help of the following example:

```
void func(string option){
   switch (option)
   {
   case "label":
      goto case "jump":
   case "quit":
      return;
   case "spin":
      for(;;){ }
    case "jump":
    case "unwind":
      throw new Exception();
   default:
      break;
   }
}
```

In this example, note that jump and unwind are stacked together and there is no jump statement in the case of jump. When a case is empty, that is there is no statement in the body of the case then it may not have any jump statement either.

Class

1. **Class Attributes:**
   - Attributes are used to give information to .Net compiler
     - E.g. it is possible to tell the compiler that a class is compliant with >Net Common Language Specification.

```
[CLSCompliant (true)]
publc class MyClass
{
        //class code
}
```

2. **Class Modifiers**

Support for OOP is provided through classes. In C#, we have the following modifiers:

- **public**: same as C++
- **internal**: internal is accessible only to types within the same assembly which is similar to package in Java.
- **protected**: same as C++
- **internal protected**: protected within the same assembly
- **private**: same as C++
- **new:**
  - The 'new' keyword can be used for 'nested' classes.
  - A nested class is one that is defined in the body of another class; it is in most ways identical to a class defined in the normal way, but its access level cannot be more liberal than that of the class in which it is defined.
  - Classes are usually specified independently of each other. But it is possible for one class to be specified within another's specification. In this case, the latter class is termed a nested class.
  - A nested class should be declared using the 'new' keyword just in case it has the same name as (and thus overrides) an inherited type.
- **abstract**: A class declared as 'abstract' cannot itself be instanced - it is designed only to be a base class for inheritance.
- **sealed**: A class declared as 'sealed' cannot be inherited from. It may be noted that structs can also not be inherited from.But it can inherit from other class.

END OF LECTURE 33

Method Modifiers

C# provides the following methods modifiers:

- **static:** The 'static' modifier declares a method to be a class method.
- **new, virtual, override**:
- **extern:** Similar to C extern. Mean that the method is defined externally, using a language other than C#

Hiding and Overriding

The main difference between hiding and overriding relates to the choice of which method to call where the declared class of a variable is different to the run-time class of the object it references.

For example:

```
public virtual double getArea()
{
        return length * width;
}

public override double getArea()
{
        return length * length;
}
```

For one method to override another, the overridden method must not be static, and it must be declared as either 'virtual', 'abstract' or 'override'. Now look at the following:

```
public double getArea()
{
        return length * width;
}

public new double getArea()
{
        return length * length;
}
```

Where one method 'hides' another, the hidden method does not need to be declared with any special keyword. Instead, the hiding method just declares itself as 'new'.

http://www.akadia.com/services/dotnet_polymorphism.html

Method Hiding

A 'new' method only hides a super-class method with a scope defined by its access modifier. In this case method calls do not always 'slide through' in the way that they do with virtual methods. So, if we declare two variables thus if we have:

```
Square sq = new Square(4);
Rectangle r = sq;
```

then

```
double area = r.getArea();
```

the getArea method run will be that defined in the Rectangle class, not the Square class.

Method parameters

In C#, as in C++, a method can only have one return value. You overcome this in C++ by passing pointers or references as parameters. In C#, with value types, however, this does not work. If you want to pass the value type by reference, you mark the value type parameter with the ref keyword as shown below.

```
public void foo(int x, ref int y)
```

Note that you need to use the ref keyword in both the method declaration and the actual call to the method.

```
someObject.f00(a, ref b);
```

Out Parameters

C# requires definite assignment, which means that the local variables, a, and b must be initialized before foo is called.

```
int a, b;
b = 0;
someObject.f00(ref a, b);       // not allowed
                                                    // a has not been
                                                    // initialized
a = 0; b =0;
someObject.f00(ref a, b);       // now it is OK
```

This is unnecessarily cumbersome. To address this problem, C# also provides the out keyword, which indicates that you may pass in un-initialized variables and they will be passed by reference.

```
public void foo(int x, ref int y, out int z)

a = 0; b = 0;
someObject.f00(a, ref b, out c);     // no need to initialize c
```

params

One can pass an arbitrary number of types to a method by declaring a parameter array with the 'params' modifier. Types passed as 'params' are all passed by value. This is elaborated with the help of the following example:

```
public static void Main(){
        double a = 1;
        int b = 2;
        int c = 3;
        int d = totalIgnoreFirst(a, b, c);
}

public static int totalIgnoreFirst(double a, params int[] intArr){
        int sum = 0;
        for (int i=0; i < intArr.Length; i++)
                sum += intArr[i];
        return sum;
}
```

Readonly fields

Readonly field are instance fields that cannot be assigned to. That is, their value is initialized only once and then cannot be modified. This is shown in the following example:

```
class Pair
{
   public Pair(int x, int y)
   {
      this.x = x;
      this.y = y;
   }
   public void Reset()
   {
      x = 0;                    // compile time errors
      y = 0;
   }
   private readonly int x, y; // this declares Pair as an immutable read only object
}
```

The is operator

- The **'is'** operator supports run time type information.
- It is used to test if the operator/expression is of certain type
  - **Syntax**: expression is type
- Evaluates to a Boolean result.
- It can be used as conditional expression.
- It will return true
  - if the expression is not NULL
  - the expression can be safely cast to type.
- The following example illustrates this concept:

```
class Dog {
…
}

class Cat {
…
}

…
// object o;
if (o is Dog)
        Console.Writeline("it's a dog");
else if (o is Cat)
        Console.Writeline("it's a cat");
else
        Console.Writeline("what is it?");
```

The as operator

- The as operator attempts to cast a given operand to the requested type.
  - **Syntax**: expression as type
- The normal cast operation – (T) e – generates an InvalidCastException when there is no valid cast.
- The as operator does not throw an exception; instead the result returned is null as shown below:
  - expression is type ? (type) expression : (type) null

The new operator

- In C++, the new keyword instantiates an object on the heap.
- In C#, with reference types, the new keyword does instantiate objects on the heap but with value types such as structs, the object is created on the stack and a constructor is called.
- You can, create a struct on the stack without using new, but be careful! New initializes the object.
- If you don't use new, you must initialize all the values in the struct by hand before you use it (before you pass it to a method) or it won't compile.

Boxing

Boxing is converting any value type to corresponding object type and convert the resultant 'boxed' type back again.

```
int i = 123;
object box = i;     // value of i is copied to the object box
if (box is int)     // runtime type of box is returned as boxed value type
{
        Console.Write("Box contains an int");  // this line is printed
}
```

Interfaces

- Just like Java, C# also has interfaces contain method signatures.
- There are no access modifier and everything is implicitly ***public.***
- It doest not have any fields, not even ***static*** ones.
- A ***class*** can implement many ***interface***s but unlike Java there is no implements keyword .
- Syntax notation is positional where we have base ***class*** first, then base ***interface***s as shown below:
  - class X: CA, IA, IB
    {
    …
    }

Delegates

Delegates are similar to function pointers. C/C++ function pointers lack instance-based knowledge whereas C# delegate are event based can be thought of a call-back mechanism where a request is made to invoke a specified method when the time is right.

Delegates are reference types which allow indirect calls to methods. A delegate instance holds references to some number of methods, and by invoking the delegate one causes all of these methods to be called. The usefulness of delegates lies in the fact that the functions which invoke them are blind to the underlying methods they thereby cause to run.

An example of delegates is shown below:

```
public delegate void Print (String s);
…

public void realMethod (String myString)
{
   // method code
}
…
```

Another method in the class could then instantiate the 'Print' delegate in the following way, so that it holds a reference to 'realMethod':

```
Print delegateVariable = new Print(realMethod);
```

Exception Handling and Multithreading

Exception handling is similar to java but is less restrictive. Multithreading is similar to java.

PHP – Personal Home Page PHP: Hypertext Preprocessor (Lecture 35-37)

A Server-side Scripting Programming Language

An Introduction

What is PHP?

PHP stands for "**P**HP: **H**ypertext **P**reprocessor". It is a server-side scripting language. PHP scripts are executed on the server and it is especially suited for Web development and can be embedded into HTML. The main goal of the language is to allow web developers to write dynamically generated web pages quickly. Its syntax is very similar to C/C++ and it has support for almost all major OS, web-servers, and databases. PHP is an open source software (OSS) and is free to download and use.

What is a PHP File?

PHP files may contain text, HTML tags and scripts. PHP files are returned to the browser as plain HTML. PHP files have a file extension of ".php", ".php3", or ".phtml". As mentioned earlier, a PHP script is run on the web server, not on the user's browser. Therefore, client-side compatibility issues are not of concern of the programmer as simply return an HTML document for the browser to process. You need three things to make this work: the PHP parser (CGI or server module), a web server and a web browser.

PHP Hello World

Following is the PHP Hello World program.

```
<html>
 <head>
   <title>PHP Test</title>
 </head>
 <body>
        <?php // start of php code
        echo '<p>Hello World</p>';
        ?>
 </body>
</html>
```

The code written within "php" tags is parsed by the PHP processor and the rest is ignored. Hence, a PHP parser would convert the above code into the following HTML document and send it to the browser.

```
<html>
  <head>
   <title>PHP Test </title>
  </head>
```

```
<body>
    <p>Hello World</p>
</body>
</html>
```

It may be noted that PHP supports C++ type of comments and requires a semicolon ";" at the end of each statement.

PHP Opening and Closing Tags

PHP scripts are always enclosed in between two PHP tags. This tells your server to parse the information between them as PHP. The three different forms are as follows:

1.
```
<?php
    echo "this is the recommended style";
?>
```

2.
```
<script language="php">
    echo "this style is also available";
</script>
```

3.
```
<?
    echo "this is short tags style; it needs to be configured";
?>
```

There is also another form which may or may not be supported on some browsers. It is shown as below:

4.
```
<%
    echo 'this is ASP-style tags; it may not be available';
%>
```

Data types

The type of a variable is decided at runtime by PHP depending on the context in which that variable is used. PHP supports the following data types:

- Boolean
- integer
- float or double
- string
- array
- object
- resource
- NULL

Integers

The size of an integer is platform-dependent, although a maximum value of about two billion is the usual value (that's 32 bits signed). 2147483647 = $2^{31} – 1$. PHP does not support unsigned integers. It is important to note that, unlike C++ or Java, if you specify a number beyond the bounds of the integer type, it will be interpreted as a float instead. If an operation results in a number beyond the bounds of the **integer** type, a **float** will be returned instead. Also, there is no integer division operator in PHP. *1/2* yields the **float** *0.5*.

Strings

PHP strings are created using single quote or double quote. They can also be created by <<< which is called heredoc. One should provide an identifier after <<<, then the string, and then the same identifier to close the quotation. The closing identifier *must* begin in the first column of the line.

Variable

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive. A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thus: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'.

Note: For our purposes here, a letter is a-z, A-Z, and any ASCII characters from 127 through 255 (0x7f-0xff). Therefore, $Inzi½ is a legal variable name. It is not necessary to initialize variables in PHP. Un-initialized variables have a default value of their type - FALSE, zero, empty string or an empty array.

String conversion to numbers

When a string is evaluated as a numeric value, the resulting value and type are determined as follows.

- The string will evaluate as a **float** if it contains any of the characters '.', 'e', or 'E'. Otherwise, it will evaluate as an integer.
- The value is given by the initial portion of the string. If the string starts with valid numeric data, this will be the value used. Otherwise, the value will be 0 (zero).

Here is an example:

```
<?php
        $foo = 2 + "10.5";                  // $foo is float (12.5)
        $foo = 1 + "-1.1e3";            // $foo is float (-1099)
        $foo = 1 + “Ali-1.3e3";             // $foo is integer (1)
        $foo = 1 + "10 Small Cats";  // $foo is integer (11)
        $foo = "10.0 cats " + 1;        // $foo is float (11)
?>
```

Using strings in numeric expressions provides flexibility but it should be obvious that it also is a source to numerous programming errors and mistakes. This kind of error is also very hard to debug and detect.

Arrays

An array in PHP is an ordered map. An **array** can be created by the **array()** construct. It takes a certain number of comma-separated *key => value* pairs. *key* may be an **integer** or **string.** The following example illustrates this concept:

```
<?php
        $arr = array("foo" => "bar", 12 => true);
        echo $arr["foo"];               // bar
        echo $arr[12];          // 1
?>
```

If no key is specified in the assignment, then the maximum of the existing integer indices is taken, and the new key will be that maximum value + 1. If the current maximum is negative then the next key created will be zero. If no integer indices exist yet, the key will be *0* (zero). Note that the maximum integer key used for this *need not currently exist in the array*. It simply must have existed in the array at some time since the last time the array was re-indexed. If the specified key already has a value assigned to it, that value will be overwritten. These things are demonstrated in the following example:

```
<?php
        $arr = array(5 => 1, 12 => 2);
        $arr[] = 56;     // This is the same as $arr[13] = 56;
                                // at this point of the script
        $arr["x"] = 42;          // This adds a new element to
                        // the array with key "x“
        unset($arr[13]);         // This removes the element from the array
        $arr[] = 56;     // This is the same as $arr[14] = 56;
                                // at this point of the script
        unset($arr);     // This deletes the whole array
```

```
?>
```

class

A class is defined as shown below:

```
<?php
        class Cart {
                var $items; // Items in our shopping cart
                        // Add $num articles of $artnr to the cart

                 function add_item($artnr, $num) {
                        $this->items[$artnr] += $num;
                }
                function remove_item($artnr, $num) {
                        if ($this->items[$artnr] > $num) {
                                $this->items[$artnr] -= $num;
                                return true;
                        }
                         elseif ($this->items[$artnr] == $num) {
                                unset($this->items[$artnr]);
                                return true;
                        }
                        else { return false; }
                }
        }
?>
```

Aliasing

Aliasing is used to assign by reference. To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable) as shown below:

```
<?php
        $foo = ‘Fakhar';
        $bar = &$foo;                   // Reference $foo via $bar.
        $bar = "My name is $bar";       // Alter $bar...
        echo $bar;
        echo $foo;                              // $foo is altered too.
?>
```

Variable variable

A variable variable is like pointers that is, it maintains the address of a variable in it. This is shown in the example below:

```
<?php
        $a = ‘hello’;
        $$a = 'world';
        echo "$a ${$a}“;                  // ‘hello world”
```

```
?>
```

Constants

A constant is an identifier (name) for a simple value that cannot change during the execution of the script. A constant is case-sensitive by default. By convention, constant identifiers are always uppercase. The name of a constant follows the same rules as any label in PHP. A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thusly: *[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]* .*

Logical operators

PHP supports the following logical operators.

and, or, xor, !, &&, ||

The reason for the two different variations of "and" and "or" operators is that they operate at different precedence. This is demonstrated with the help of the following example:

```
$a and $b or $c     //      ($a and $b) or $c
$a && $b || $c
$a and $b || $c     //      $a and ($b || $c)
```

String operators

There are two **string** operators. The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments. The second is the concatenating assignment operator ('.='), which appends the argument on the right side to the argument on the left side.

```
<?php
        $a = "Hello ";
        $b = $a . "World!"; // now $b contains "Hello World!"
        $a = "Hello ";
        $a .= "World!";         // now $a contains "Hello World!"
?>
```

Array Operators

PHP provides a number of operators to manipulate arrays. These are:

- $a + $b Union – Union of $a and $b.
- $a == $b Equality – **TRUE** if $a and $b have the same key/value pairs.
- $a === $b Identity - **TRUE** if $a and $b have the same key/value pairs in the same order and of the same types.
- $a != $b Inequality - **TRUE** if $a is not equal to $b.
- $a <> $b Inequality - **TRUE** if $a is not equal to $b.
- $a !== $b Non-identity - **TRUE** if $a is not identical to $b.

Note the difference between == and ===. The former checks if the same key-value pairs are present in the two arrays and the order does not matter whereas the latter requires that in order to be identical the pairs have to be in the same order.

The + operator appends the right handed array to the left handed, whereas duplicated keys are NOT overwritten. This concept is elaborated as below:

```
$a = array("a" => "apple", "b" => "banana");
$b = array("a" => "pear", "b" => “date", "c" => “mango");
$c = $a + $b; // Union of $a and $b
```

This will result in an array c with the following values:

["a"]=> "apple" ["b"]=> "banana" ["c"]=> “mango"

Control Statements

PHP supports the following control statements:

- if
- while
- do while
- for
- switch
- foreach
- break
- continue

Most control structures work in a manner similar to C. The differences are highlighted in the following paragraphs.

If statements

The if statement in PHP has the following shape and form:

```
<?php
        if ($a > $b) { echo "a is bigger than b"; }
        elseif ($a == $b) { echo "a is equal to b"; }
        else { echo "a is smaller than b"; }
?>
```

There may be several *elseif*s within the same *if* statement. The first *elseif* expression (if any) that evaluates to **TRUE** would be executed. In PHP, you can also write 'else if' (in two words) and the behavior would be identical to the one of 'elseif' (in a single word).

foreach

foreach statement works only on arrays, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable. It gives an easy way to iterate over arrays. There are two syntaxes; the second is a minor but useful extension of the first:

- foreach (array_expression as $value) statement
- foreach (array_expression as $key => $value) statement

The first form loops over the array given by *array_expression*. On each loop, the value of the current element is assigned to *$value* and the internal array pointer is advanced by one. The second form does the same thing, except that the current element's key will also be assigned to the variable *$key* on each loop. As of PHP 5, it is possible to iterate objects too.

Note: Unless the array is referenced, *foreach* operates on a copy of the specified array and not the array itself. Therefore changes to the array element returned are not reflected in the original array. As of PHP 5, array's elements can be modified by preceding *$value* with &. This will assign reference instead of copying the value.

```
<?php
        $arr = array(1, 2, 3, 4);
        foreach ($arr as &$value)
                { $value = $value * 2; }
                        // $arr is now array(2, 4, 6, 8)
?>
```

This is possible only if iterated array can be referenced (i.e. is variable).

each

each return the current key and value pair from an array and advance the array cursor. It is used in the following manner:

array **each** (array &array)

Returns the current key and value pair from the array *array* and advances the array cursor. This pair is returned in a four-element array, with the keys *0*, *1*, *key*, and *value*. Elements *0* and *key* contain the key name of the array element, and *1* and *value* contain the data. If the internal pointer for the array points past the end of the array contents, **each()** returns **FALSE**.

break

break ends execution of the current *for*, *foreach*, *while*, *do-while* or *switch* structure. In PHP, *break* accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of. The following example illustrates this concept:

```
<?php
        $i = 0;
        while (++$i) {
                switch ($i) {
                        case 5:
                                echo "At 5<br />\n";
                                break 1; // Exit only the switch.
                        case 10:
                                echo "At 10; quitting<br />\n";
                                break 2; // Exit the switch and the while.
                        default:
                                break;    // Exit only the switch.
                }
        }
?>
```

continue

continue is used within looping structures to skip the rest of the current loop iteration and continue execution at the condition evaluation and then the beginning of the next iteration. **Note:** Note that in PHP the switch statement is considered a looping structure for the purposes of *continue*. Like break, *continue* accepts an optional numeric argument which tells it how many levels of enclosing loops it should skip to the end of.

Omitting the semicolon after *continue* can lead to confusion as shown below:

```
<?php
        for ($i = 0; $i < 5; ++$i) {
                if ($i == 2) continue print "$i\n";
        }
?>
```

One can expect the result to be : 0 1 3 4 but this script will output : 2 because the return value of the **print()** call is *int(1)*, and it will look like the optional numeric argument mentioned above.

Alternative syntax for control statements

PHP offers an alternative syntax for some of its control structures; namely, *if*, *while*, *for*, *foreach*, and *switch*. In each case, the basic form of the alternate syntax is to change the opening brace to a colon (:) and the closing brace to *endif;*, *endwhile;*, *endfor;*, *endforeach;*, or *endswitch;*, respectively. An example is shown below:

```
<?php
        if ($a == 5):
                echo "a equals 5";
                echo "...";
        elseif ($a == 6):
                echo "a equals 6";
                echo "!!!";
        else: echo "a is neither 5 nor 6";
        endif;
?>
```

User defined functions

Just like all programming languages, one can define functions in PHP as shown in the following example:

```
<?php
        function foo($arg_1, $arg_2, /* ..., */ $arg_n)
        {
                echo "Example function.\n";
                return $retval;
        }
?>
```

PHP also allows functions to be defined inside functions. It may be noted that C and its descendents do not support this feature but descendants of Algol generally support it. The following example shows this concept:

```
<?php
        function foo() {
                function bar() {
                echo "I don't exist until foo() is called.\n";
                }
        }

        /* We can't call bar() yet since it doesn't exist. */

        foo();

        /* Now we can call bar(), foo()'s processesing has made it
          accessible. */

        bar();
```

```
?>
```

PHP also supports recursive functions as shown below:

```
<?php
        function recursion($a) {
                if ($a < 20) {
                        echo "$a\n"; recursion($a + 1);
                }
        }
?>
```

Function arguments

PHP supports passing arguments by value (the default), passing by reference, and default argument values. If you want an argument to a function to always be passed by reference, you can prepend an ampersand (&) to the argument name in the function definition:

```
<?php
        function add_some_extra(&$string) {
                $string .= 'and something extra.';
        }
        $str = 'This is a string, ';
        add_some_extra($str);
        echo $str; // outputs 'This is a string, and
                        // something extra.‘
?>
```

You can't return multiple values from a function, but similar results can be obtained by returning a list as shown below:

```
<?php
                function small_numbers() {
                        return array (0, 1, 2);
                }
                list ($zero, $one, $two) = small_numbers();
?>
```

To return a reference from a function, you have to use the reference operator & in both the function declaration and when assigning the returned value to a variable:

```
<?php
                function &returns_reference() {
                        return $someref;
                }
                $newref =& returns_reference();
?>
```

Classes and Objects

Classes and objects are similar to Java. A variable of the desired type is created with the *new* operator. It supports Single inheritance only and uses the keyword extends to inherit from a super class. The inherited methods and members can be overridden, unless the parent class has defined a method as final.

Databases

One of the strongest features of PHP is its support for providing web pages access to database. The following example shows database manipulation through ODBC in PHP:

```
<html>
<body>
<?php
        $conn=odbc_connect('northwind',",");
        if (!$conn)  {
                exit("Connection Failed: " . $conn);
        }
        $sql="SELECT * FROM customers";
        $rs=odbc_exec($conn,$sql);
        if (!$rs) {exit("Error in SQL");}
        echo "<table><tr>";
        echo "<th>Companyname</th>";
        echo "<th>Contactname</th></tr>“;
        while (odbc_fetch_row($rs)){
                $compname=odbc_result($rs,"CompanyName");
                $conname=odbc_result($rs,"ContactName");
                echo "<tr><td>$compname</td>";
                echo "<td>$conname</td></tr>";
        }
        odbc_close($conn);
        echo "</table>";
?>
</body>
</html>
```

Modern Programming Languages-JavaScript
Lecture 38

The <script> Tag

The <script> tag (<script>….</script>) in all major browsers interprets contents as JavaScript unless one of the following occurs

Inclusion of language attribute

<script language="VBS">….</script>
Inclusion of type attribute
<script type="text/javascript">….</script>

The type attribute is W3C recommended, it makes the language more common and in many ways more useful

<script> tag is used to delimit the script code from the HTML

- **The script tag causes the browser's JavaScript interpreter to be invoked, the script run and any output produced**
- **The browser is considered the "host" environment**
- **There are other hosts for JavaScript and its variants**

Location of Code

JavaScript may be placed at three places
In the <head> element

Place scripts to be called or when event is triggered here
Ensures script is loaded before called

```
<html>
 <head>
  <script type="text/javascript">
                //script statements
  </script>
 </head>
```

Location of Code

In the <body> element
Place scripts to be executed when the page loads here
Script generates some or all of the contents of the page

```
 <body>
  <script type="text/javascript">
                //script statements
  </script>
 </body>
```

Location of Code

External to the HTML file

```
<head>
  <script src="myfile.js">
  </script>
</head>
```

Could be in <head> or <body>
External script should not contain <script> tag

- You can use as many **<script>** tags as you like in both the **<head>** and **<body>** and they are executed sequentially.

```
<h1>Ready start</h1>
<script language="Javascript" type="text/javascript">
    alert("First Script Ran");
</script>
<h2>Running...</h2>
<script language="Javascript" type="text/javascript">
    alert("Second Script Ran");
</script>
<h2>Keep running</h2>
<script language="Javascript" type="text/javascript">
    alert("Third Script Ran");
</script>
</h1>Stop!</h1>
</body>
```

Statements

- A script is made up of individual statements
- Javascript statements are terminated by returns or semi-colons same as

- So, prefer to use semi-colons

```
x=x+1    alert(x) //throws an error while
x=x+1; alert(x); //does not
```

- Every variable has a data type that indicates what kind of data the variable holds
- Basic data types in JavaScript
- Strings

Strings may include special escaped characters

- Numbers (integers, floats)
- Booleans (true, false)
- 'null' and 'undefined' are also considered as types

- Define a variable using the var statement
  - `var x;`
- If undefined a variable will be defined on its first use
- Variables can be assigned at declaration time
  - `var x = 5;`
- Commas can be used to define many variables at once
  - `var x, y = 5, z;`

- JavaScript is a weekly typed language meaning that the contents of a variable can change from one type to another
- Example

x = "hello"; x = 5; x=false;

While week typing seems beneficial to a programmer it can lead to problems

Arrays

- An ordered set of values grouped together with a single identifier
- Defining Arrays

- var myArray = [1,5,1968,3];
- var myArray2 = ["fakhar",true,3,-47.2];
- var myArray3 = new Array ();
- var myArray4 = new Array (10);

Arrays

- Arrays in JavaScript are 0 based
- We access arrays by index values

- var myArray = [1,5,1968,3];
- myArray[3] is '3'

Difference

Array types (composite type as well) are reference types, so problem of aliasing is there

```
var firstArray = ["Mars", "Jupiter", "Saturn" ];
var secondArray = firstArray;
secondArray[0] = "Neptune";
alert(firstArray); // it has been changed
```

Operators

- Basic Arithmetic
  + , - , / , * , %
- Increment decrement
  ++, - -
- Comparison
  < , > , >= , <= , != , == , === (type equality)
- Logical
  &&, ||, !
- Bitwise Operators
  & , | , ^
- String Operator

- + (used for concatenation)
- document.write("JavaScript" + "is" + "great!");

Type Conversion

- **Converting one type of data to another is both useful and a source of numerous errors in JavaScript**
- var x = "10" – 2 ; //result is 8
- var x = "2" – "2" ; //result is 0
- var x = "2" + "2" ; //result is "22"

Control Statements

- **If**
- **Switch**
- **While**

- **Do-while**
- **For**
- **Continue**
- **Break**
- **For (in)**

Labels and Flow Control

```
outerloop:
  for (var i=0; i < 3; i++)
  {
        document.write("Outerloop: "+i+"<br />");
        for (var j = 0; j < 5; j++)
          {
            if ( j == 3)
              break outerloop;
            document.write("Innerloop: "+j+"<br />");
          }
  }
  document.write("All loops done"+"<br>");
```

```
function name(parameter list)
  {
  function statement(s)
  return;
  }
```

Local Functions

```
function testFunction()
    {

    function inner1() {  document.write("testFunction-inner1<br />"); }

    function inner2() {  document.write("testFunction-inner2<br />");  }

     document.write("Entering testFunction<br />");
     inner1();
     inner2();
     document.write("Leaving testFunction<br />");
    }

    document.write("About to call testFunction<br />");
    testFunction();
    document.write("Returned from testFunction<br />");


  /* Call inner 1 or inner2 here and error */
    inner1();
```

Modern Programming Languages-JavaScript
Lecture 39

Objects

- **An object is a collection of data types as well as functions in one package**
- **The various data types called properties and functions called methods are accessed using the dot notation**

- **objectname.propertyname**

Objects

- **There are many types of objects in JavaScript**

- **Built-in objects (primarily type related)**
- **Browser objects (navigator, window etc.)**
- **Document objects (forms, images etc.)**
- **User defined objects**

Two Object Models

- **In JavaScript, two primary object models are employed**
- **Browser Object Model (BOM)**

- **The BOM provides access to the various characteristics of a browser such as the browser window itself, the screen characteristics, the browser history and so on.**
- **Document Object Model (DOM)**
- **The DOM on the other hand provides access to the contents of the browser window, namely the documents including the various HTML elements ranging from anchors to images as well as any text that may be enclosed by such element.**

The Big Picture

- Looking at the "big picture" of all various aspects of JavaScript including its object models. We see four primary pieces:
  1. The core JavaScript language (e.g. data types, operators, statements, etc.)
  2. The core objects primarily related to data types (e.g. Date, String, Math, etc.)
  3. The browser objects (e.g. Window, Navigator, Location, etc.)
  4. The document objects (e.g. Document, Form, Image, etc.)

Four Models

- By studying the history of JavaScript we can bring some order to the chaos of competing object models. There have been four distinct object models used in JavaScript including:
  1. Traditional JavaScript Object Model (NS 2 & IE 3)
  2. Extended Traditional JavaScript Object Model (NS 3)
  3. Dynamic HTML Flavored Object Models
     1. a. IE 4
     2. b. NS 4
  4. Traditional Browser Object Model + Standard DOM (NS6 & Explorer 5)

Overview of Core Objects

Object	Description
Window	The object that relates to the current browser window.
Document	An object that contains the various HTML elements and text fragments that make up a document. In the traditional JavaScript object model, the **Document** object relates roughly the HTML <body> tag.
Frames[ ]	An array of the frames in the Window contains any. Each frame in turn references another **Window** object that may also contain more frames.
History	An object that contains the current window's history list, namely the collection of the various URLs visited by the user recently.
Location	Contains the current location of the document being viewed in the form of a URL and its constituent pieces.
Navigator	An object that describes the basic characteristics of the browser, notably its type and version.

Document Object

- The **Document** object provides access to page elements such as anchors, form fields, and links as well as page properties such as background and text color.
- Consider
  - document.alinkColor, document.bgColor, document.fgColor, document.URL
  - document.forms[ ], document.links[ ], document.anchors[ ]
- We have also used the methods of the **Document** object quite a bit
  - document.write( ) , document.writeln( ), document.open( ), document.close( )

Object Access by Document Position

- HTML elements exposed via JavaScript are often placed in arrays or collections. The order of insertion into the array is based upon the position in the document.
- For example, the first **<form>** tag would be in document.forms[0], the second in document.forms[1] and so on.
- Within the form we find a collection of elements[ ] with the first **<input>, <select>** or other form field in document.forms[0].elements[0] and so on.
- As arrays we can use the length property to see how many items are in the page.
- The downside of access by position is that if the tag moves the script may break

Object Access by Name

- When a tag is named via the **name** attribute (HTML 4.0 - **<a>**, **<img>**, embedded objects, form elements, and frames) or by **id** attribute (pretty much every tag) it should be scriptable.
- Given

```
<form id="myform" name="myform">
  <input type="text" name="username" id="username">
</form>
```

we can access the form at window.document.myform and the first field as

window.document.myform.username

Object Access by Associative Array

- The collection of HTML objects are stored associatively in the arrays.
- Given the form named "myform" we might access it using

  window.document.forms["myform"]

- In Internet Explorer we can use the **item( )** method like so

  window.document.forms.item("myform")

Object Statement: With

```
document.write("Hello World");
document.write("<br />");
document.write("this is another write statement");

with (document) {
        write("Hello World");
         write("<br />");
         write("this is another write statement");
}
```

Object Statements: for..in

- The **for...in** statement is used to loop through the properties of an object (if they can be enumerated)
- Syntax

  for (*variablename* in *object*)

  statement or block to execute

- Example

```
var aProperty;
 document.write("<h1>Navigator Object Properties</h1>");
 for (aProperty in navigator)
 {
   document.write(aProperty);
   document.write("<br />");
 }
```

Events

- One of the primary uses of JavaScript is to make Web pages interactive.
  - Responsive to user actions
- JavaScript provides event handlers.
  - Execute segment of code based on events occurring within the application
  - E.g., `onLoad` or `onClick`
- Handlers associated with elements.
- Not all elements support all event handlers.

16

Events (cont.)

```
<input type="button"
   name="clickme"
   value="Click Here"
   onClick=
      "window.status='Thanks';
     return true;">
```

17

Events (cont.)

- Event handlers can be categorized into interactive and non-interactive.
- Interactive: Depends on a user action.
  - E.g., `onClick`
- Non-interactive: Non-user event.
  - E.g., `onLoad`

18

Events (cont.)

- `onAbort`: Image loading is interrupted.
- `onBlur`: Element loses input focus.
- `onChange`: User selects or deselects item.
- `onClick`: User clicks once.
- `onDragDrop`:
- `onError`: Image doesn't load properly.

19

Events (cont.)

- onFocus: Element is given input focus.
- onKeyPress:
- onKeyUp:
- onLoad:
- onMouseDown:
- onMouseOver:
- onMouseOut:
- onMouseUp:

Events
Supporting Events

1. Give the target HTML element a name attribute.

```
<input type="text"
       name="price"  />
```

2. Give activating HTML element event attribute that calls function.

```
<input type="submit"
    value="Calculate total."
    onClick="calcTotal()"/>
```

22

Modern Programming Languages
Lecture # 40

Names

- Design issues:
  - Maximum length?
  - Are connector characters allowed?
  - Are names case sensitive?
  - Are special words reserved words or keywords?

Special Words

- There are two types of special words
  - Keyword
  - Reserved word
- A keyword is a word that is special only in a certain context
  - REAL X
  - REAL = 44.7
- Disadvantage: poor readability

- A reserved word is a special word that cannot be used as a user-defined name
  - Type

- **Type:** Determines the range of values of variables and the set of operations that are defined for values of that type; in the case of floating point, type also determines the precision
  - Value
- **Value:** The contents of the location with which the variable is associated
  - Binding

- **Binding :** A binding is an association, such as between an attribute and an entity, or between an operation and a symbol

  - Binding Time
    It is the time at which a binding takes place

Possible binding times

1. Language design time - e.g. bind operator symbols to operations
2. Language implementation time - e.g. bind fl. pt. type to a representation
3. Compile time - e.g. bind a variable to a type in C or Java
4. Load time- e.g.
5. Runtime - e.g. bind a non-static local variable to a memory cell

Static and Dynamic Binding

- A binding is static if it occurs before run time and remains unchanged throughout program execution

- A binding is dynamic if it occurs during execution or can change during execution of the program

Type Bindings

- How is a type specified?
- When does the binding take place?
- If static, type may be specified by either an explicit or an implicit declaration
  - An explicit declaration is a program statement used for declaring the types of variables
  - An implicit declaration is a default mechanism for specifying types of variables (the first appearance of the variable in the program)
- FORTRAN, PL/I, BASIC, provide implicit declarations
  - Advantage: writability
  - Disadvantage: reliability

Dynamic Type Binding

- Specified through an assignment statement e.g. in SNOBOL
  - LIST = 'my name'
  - LIST = 4 + 5
- Advantage
  - Flexibility (generic program units)
- Disadvantages:
  - Type error detection by the compiler is difficult
  - High cost (dynamic type checking and interpretation)

Storage Bindings

- At what time we associate the memory location with a name

Categories of variables by lifetimes

1. **Static Storage Binding**

- Bound to memory cells before execution begins and remains bound to the same memory cell throughout execution
- Example
  - In FORTRAN 77 all variables are statically bound
  - C static variables
- Advantage
  - Efficiency (direct addressing), history-sensitive, subprogram support

- Disadvantage
  - Lack of flexibility (no recursion)

Stack Dynamic Variables:

- Bound to storages when execution reaches the code to which the declaration is attached. (But, data types are statically bound.) That is, stack-dynamic variables are allocated from the **run-time stack**.
- Example
  - E.g. a Pascal procedure consists of a declaration section and a code section.
  - E.g. FORTRAN 77 and FORTRAN 90 use SAVE list for stack-dynamic list.
  - E.g. C and C++ assume local variables are static-dynamic.
- Advantage
  - allows recursion; conserves storage
- Disadvantages
  - Overhead of allocation and deallocation
  - Subprograms cannot be history sensitive
  - Inefficient references (indirect addressing)

Explicit Heap Dynamic Variables

- Allocated and de-allocated by explicit directives, specified by the programmer, which take effect during execution
- Referenced only through pointers or references
- Examples
  - Dynamic objects in C++ (via new and delete)
  - All objects in Java
- Advantage
  - Provide dynamic storage management
- Disadvantage
  - Inefficient and unreliable

Implicit Heap Dynamic Variables

- Allocation and de-allocation is caused by assignment statements
- Example
  - All variables in SNOBOL
- Advantage
  - Flexibility
- Disadvantages
  - Inefficient, because all attributes are dynamic
  - Loss of error detection

Modern Programming Languages Lecture 41

Type Checking

- Generalizes the concept of operands and operators to include subprograms and assignments
- Type checking is the activity of ensuring that the operands of an operator are of compatible types
- A compatible type is one that is either legal for the operator, or is allowed under language rules to be implicitly converted, by compiler generated code, to a legal type. This automatic conversion is called **coercion**
- A type error is the application of an operator to an operand of an inappropriate type
- If all type bindings are static, nearly all type checking can be static
- If type bindings are dynamic, type checking must be dynamic
- A programming language is strongly typed if type errors are always detected
- A programming language which does all the type checking statically then it is a strongly typed languages otherwise weakly typed language.

Strongly Typed Languages?

- FORTRAN 77 is not strongly typed
  - Parameters, EQUIVALENCE
- C and C++ are not
  - Parameter type checking can be avoided
  - Unions are not type checked
- Pascal is not
  - Variant records
- Modula-2 is not
  - Variant records
- Ada is, almost
  - UNCHECKED CONVERSION is a loophole
  - Coercion rules strongly affect strong typing; they can weaken it considerably (C++ versus Ada)
- Advantage of strong typing
  - Allows the detection of the misuse of variables that result in type errors

Type Compatibility

- Type compatibility by name means that the two variables have compatible types if they are in either the same declaration or in declarations that use the same type name
- Easy to implement but highly restrictive
  - Sub ranges of integer types are not compatible with integer types

Data Types

- Design Issues
  - What is the syntax of references to variables?
  - What operations are defined and how are they specified?

Primitive Data Types

- Not defined in terms of other data types

1. **Integer**
   - Almost always an exact reflection of the hardware, so the mapping is trivial
   - There may be as many as eight different integer types in a language

2. **Floating Point**
   - Models real numbers, but only as approximations
   - Languages for scientific use support at least two floating-point types; sometimes more
   - Usually exactly like the hardware, but not always; some languages allow accuracy specs in code e.g. (Ada)
     - type SPEED is digits 7 range 0.0..1000.0;
     - type VOLTAGE is delta 0.1 range -12.0..24.0;

3. **Decimal**
   - For business applications (money)
   - Store a fixed number of decimal digits
   - Advantage
     - Accuracy
   - Disadvantages
     - Limited range, wastes memory
   - COBOL and ADA supports decimal
   - C++ and java does not support decimal

4. **Boolean**
   - Could be implemented as bits, but often as bytes
   - Advantage
     - Readability

Character String Types

- Values are sequences of characters
- Design issues:
  - Is it a primitive type or just a special kind of array?
  - Is the length of objects static or dynamic?
- Operations:
  - Assignment
  - Comparison (=, >, etc.)
  - Concatenation
  - Substring reference
  - Pattern matching

- Examples
  - Pascal---Not primitive; assignment and comparisons supported only
  - Ada, FORTRAN 77, FORTRAN 90 and BASIC
    - Somewhat primitive
    - Assignment, comparison, catenation
    - Substring reference
  - C and C++
    - Not primitive
    - Use char arrays and a library of functions that provide these operations
  - SNOBOL4 (a string manipulation language)
    - Primitive
    - Many operations, including elaborate pattern matching
  - Java; string class (not arrays of char)

Ordinal Types (user defined)

- An ordinal type is one in which the range of possible values can be easily associated with the set of positive integers
- Enumeration Types - one in which the user enumerates all of the possible values, which are symbolic constants
- Design Issue
  - Should a symbolic constant be allowed to be in more than one type definition?
- Examples
  - Pascal
    - Cannot reuse constants
    - They can be used for array subscripts e.g. for variables, case selectors
    - NO input or output
    - Can be compared
  - Ada
    - Constants can be reused (overloaded literals)
    - Disambiguates with context or type_name
    - CAN be input and output
  - C and C++
    - Like Pascal, except they can be input and output as integers
  - Java does not include an enumeration types
  - C# includes them

- Evaluation (of enumeration types)
  - Useful for readability
    - e.g. no need to code a color as a number
  - Useful for reliability
    - e.g. compiler can check operations and ranges of values

- Subrange Type
  - An ordered contiguous subsequence of an ordinal type
  - Pascal
    - Subrange types behave as their parent types; can be used as for variables and array indices
    - E.g. type pos = 0 .. MAXINT;
  - Ada
    - Subtypes are not new types, they are just constrained existing types (so they are compatible)
    - Can be used as in Pascal, plus case constants
    - **E.g.** subtype POS_TYPE is
    - **E.g.** INTEGER range 0 ..INTEGER'LAST;
  - Evaluation (of enumeration types)
    - Aid to readability
    - Reliability - restricted ranges add error detection

- Implementation of user-defined ordinal types
  - Enumeration types are implemented as integers
  - Subrange types are the parent types with code inserted (by the compiler) to restrict assignments to subrange variables

5. Arrays

- An array is an aggregate of homogeneous data elements in which an individual element is identified by its position in the aggregate, relative to the first element
- Design Issues
  - What types are legal for subscripts?
  - Are subscripting expressions in element references range checked?
  - When are subscript ranges bound?
  - When does allocation take place?
  - What is the maximum number of subscripts?
  - Can array objects be initialized?

6. Subscript Types

- FORTRAN, C, C++
  - int only
- Pascal
  - Any ordinal type (int, boolean, char, enum)
- Ada
  - int or enum (includes boolean and char)
- Java
  - integer types only

Four Categories of Arrays (based on subscript binding and binding to storage)

- **Static -** range of subscripts and storage bindings are static e.g. FORTRAN 77, some arrays in Ada
  - Advantage: execution efficiency (no allocation or de-allocation)
- **Fixed stack dynamic -** range of subscripts is statically bound, but storage is bound at elaboration time e.g. C local arrays are not static
  - Advantage: space efficiency
- **Stack-dynamic -** range and storage are dynamic, but fixed from then on for the variable's lifetime e.g. Ada declare blocks declare

  - STUFF : array (1..N) of FLOAT;
    begin
    ...
    end;
  - **Advantage:** flexibility - size need not be known until the array is about to be used

Modern Programming Languages Lecture 42

Records-(like structs in C/C++)

- **Description:**
  - Are composite data types like arrays
  - The difference between array and record is that array elements are of same data type where as record is a logical grouping of heterogeneous data elements.
  - Another difference is that access to array elements is much slower than access to record fields, because subscripts are dynamic (field names are static)
  - Dynamic subscripts could be used with record field access, but it would disallow type checking and it would be much slower

- **Design Issues:**
  - What is the form of references?
  - What unit operations are defined?

- **Record Definition Syntax**
  - COBOL uses level numbers to show nested records; others use recursive definitions

  - Examples:
    - COBOL

      field_name OF record_name_1 OF ... OF record_name_n

    - Others (dot notation)

      record_name_1.record_name_2. ... .record_name_n.field_name

  - Better Approach:
    - According to readability point of view COBOL record definition syntax is easier to read and understand but according to writability point of view other languages record definition syntax is easier to write and less time consuming.

- **Record Field References**

  - Two types of record field references:
    - Fully qualified references must include all record names

    - Elliptical references allow leaving out record names as long as the reference is unambiguous

  - Pascal and Modula-2 provide a with clause to abbreviate references

- **Record Operations**

  - Assignment
    - Pascal, Ada, and C allow it if the types are identical
    - In Ada, the RHS can be an aggregate constant

  - Initialization
    - - Allowed in Ada, using an aggregate constant

  - Comparison
    - - In Ada, = and /=; one operand can be an aggregate constant

Pointers

- **Description:**
  - A pointer type is a type in which the range of values consists of memory addresses and a special value, nil (or null)

- **Uses:**
  - Addressing flexibility
  - Dynamic storage management

- **Fundamental Pointer Operations:**
  - Assignment of an address to a pointer
  - References (explicit versus implicit dereferencing)

- **Design Issues**
  - What is the scope and lifetime of pointer variables?
  - What is the lifetime of heap-dynamic variables?
  - Are pointers restricted to pointing at a particular type?
  - Are pointers used for dynamic storage management, indirect addressing, or both?
  - Should a language support pointer types, reference types, or both?
  - Pointer arithmetic?

- **Problems with Pointers**

  - Dangling pointers
    - A pointer points to a heap-dynamic variable that has been deallocated

  - Lost Heap-Dynamic Variables
    - A heap-dynamic variable that is no longer referenced by any program pointer
    - The process of losing heap-dynamic variables is called memory leakage

  - Pointers are like goto's - they widen the range of cells that can be accessed by a variable
  - Pointers are necessary - so we can't design a language without them

- **Examples:**
  - **Pascal:** used for dynamic storage management only
    - Explicit dereferencing
    - Dangling pointers are possible (dispose)
    - Dangling objects are also possible

  - **Ada:** a little better than Pascal and Modula-2
    - Some dangling pointers are disallowed because dynamic objects can be automatically deallocated at the end of pointer's scope
    - All pointers are initialized to null
    - Similar dangling object problem (but rarely happens)

  - **C/C++:** Used for dynamic storage management and addressing
    - Explicit dereferencing and address-of operator
    - Can do address arithmetic in restricted forms
    - Domain type need not be fixed (void *)
    - ```
      float stuff[100];
      float *p;
      p = stuff;
      ```
    - *(p+5) is equivalent to stuff [5] and p [5]
    - *(p+i) is equivalent to stuff[i] and p[i]
    - void * - can point to any type and can be type checked (cannot be dereferenced)

  - **Fortran90:** Can point to heap and non-heap variables
    - Implicit dereferencing
    - Special assignment operator for non dereferenced references

  - **C++ Reference Types**
    - Constant pointers that are implicitly dereferenced
    - Used for parameters
    - Advantages of both pass-by-reference and pass-by-value

  - **Java - Only references**
    - No pointer arithmetic
    - Can only point at objects (which are all on the heap)
    - No explicit deallocator (garbage collection is used)
    - Means there can be no dangling references
    - Dereferencing is always implicit

Unions

- **Description:**

  - A union is a type whose variables are allowed to store different type values at different times during execution

- **Design Issues for unions:**

  - What kind of type checking, if any, must be done?
  - Should unions be integrated with records?
      ```

- **Examples:**

  - FORTRAN
    - EQUIVALENCE
  - Algol has "discriminated unions"
    - Use a hidden tag to maintain the current type
    - Tag is implicitly set by assignment
    - References are legal only in conformity clauses
    - This runtime type selection is a safe method of accessing union objects
  - Pascal
    - Both discriminated and non-discriminated unions are used e.g.

```
type intreal = record tag : Boolean of
                    true : (blint : integer);
                    false : (blreal : real);
               end;
```

    - Problem with Pascal's design is that type checking is ineffective
    - Reasons:
      - User can create inconsistent unions (because the tag can be individually assigned)

```
var blurb : intreal;
    x : real;
blurb.tagg := true; { it is an integer }
blurb.blint := 47;    { ok }
blurb.tagg := false;  { it is a real }
x := blurb.blreal;    { assigns an integer
                        to a real }
```

      - The tag is optional!

  - Ada
    - Discriminated unions are safer than Pascal & Modula-2
    - Reasons
      - Tag must be present
      - It is impossible for the user to create an inconsistent union (because tag cannot be assigned by itself)
      - All assignments to the union must include the tag value)

  - C and C++ have free unions (no tags)
    - Not part of their records
    - No type checking of references

  - Java has neither records nor unions but aggregate types can be created with classes, as in C++

  - Unions are potentially unsafe in most languages (except Ada)

Arithmetic Expressions

- **Description:**

  - Arithmetic evaluation was one of the motivations for the development of the first programming languages
  - Arithmetic expressions consist of operators, operands, parentheses, and function calls
  - 15 * (a + b) / log(x)

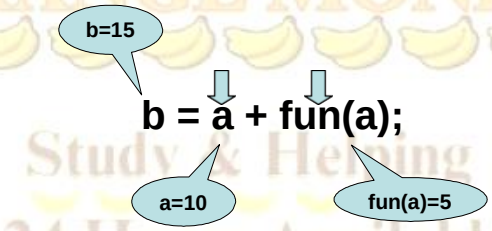
- **Design issues for arithmetic expressions:**

  - What are the operator precedence rules?
  - What are the operator associativity rules?
  - What is the order of operand evaluation?
  - Are there restrictions on operand evaluation side effects?
  - Does the language allow user-defined operator overloading?
  - What mode mixing is allowed in expressions?
  - Conditional expressions?

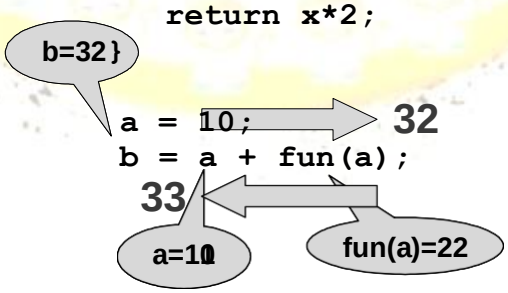
Modern Programming Languages Lecture 43

- **Arithmetic Expressions: Operators**
  - A unary operator has one operand
  - A binary operator has two operands
  - A ternary operator has three operands
- **Arithmetic Expressions: Operator Precedence Rules**
  - The operator precedence rules for expression evaluation define the order in which "adjacent" operators of different precedence levels are evaluated
  - Typical precedence levels
    - parentheses
    - unary operators
    - ** (if the language supports it)
    - *, /
    - +, -
- **Arithmetic Expressions: Potentials for Side Effects**
  - Functional side effects: when a function changes a two-way parameter or a non-local variable
  - Problem with functional side effects:
    - When a function referenced in an expression alters another operand of the expression; e.g., for a parameter change:

```
a = 10;
/* assume that fun changes its parameter */
b = a + fun(a);
```




```
int fun(int &x)
{
    x = x + 1;
    return x*2;
```



- **Functional Side Effects**
  - Two possible solutions to the problem
    - Write the language definition to disallow functional side effects
      - No two-way parameters in functions
      - No non-local references in functions
      - **Advantage:** it works!
      - **Disadvantage:** inflexibility of two-way parameters and non-local references
  - Write the language definition to demand that operand evaluation order be fixed
    - **Disadvantage:** limits some compiler optimizations

- **Operator Overloading**
  - Use of an operator for more than one purpose is called operator overloading
  - Some overloaded operators are common (e.g. '+' for int and float)
  - Some are potential trouble (e.g. '*' in C and C++)
    - * is also used for pointers
  - Loss of compiler error detection (omission of an operand should be a detectable error)
  - Some loss of readability
  - Can be avoided by introduction of new symbols (e.g., Pascal's div for integer division)
  - C++ and Ada allow user-defined overloaded operators
  - Potential problems
    - Users can define nonsense operations
    - Readability may suffer

- **Type Conversion**
  - Implicit Type Conversion
  - Explicit Type Conversion

- **Implicit Type Conversions**
  - A **narrowing conversion** is one that converts an object to a type that cannot include all of the values of the original type e.g., float to int
  - A **widening conversion** is one in which an object is converted to a type that can include at least approximations to all of the values of the original type e.g., int to float
  - A **mixed-mode expression** is one that has operands of different types
  - **Coercion** is an implicit type conversion
  - **Disadvantage of coercions:**
    - They decrease in the type error detection ability of the compiler
  - In most languages, all numeric types are coerced in expressions, using widening conversions
  - In Ada, there are virtually no coercions in expressions

- **Explicit Type Conversions**
  - Called casting in C-based language
  - Examples
    - C: (int) angle
    - Ada: Float (sum)
  - Note that Ada's syntax is similar to function calls

- **Errors in Expressions**
  - Causes
    - Coercions of operands in expressions
    - Inherent limitations of arithmetic e.g., division by zero
    - Limitations of computer arithmetic e.g. overflow or underflow
  - Division by zero, overflow, and underflow are run-time errors (sometimes called exceptions)

Relational Expressions

- Use relational operators and operands of various types

- Evaluate to some boolean representation

- Operator symbols used vary somewhat among different languages (!=, /=, .NE., <>, #)

Boolean Expressions

- Operands are Boolean and the result is also a Boolean

- Operators

FORTRAN 77	FORTRAN 90	C	Ada
AND	and	&&	and
OR	or	||	or
NOT	not	!	not
	xor		

Boolean Expressions

One odd characteristic of C's expressions
a < b < c

Short Circuit Evaluation

- A and B

- A or B

- *Example*

```
index := 1;
while (index <= length) and
      (LIST[index] <> value) do
  index := index + 1
```

C, C++, and Java*:* use short-circuit evaluation for the usual Boolean operators (&& and ||), but also provide bitwise Boolean operators that are not short circuit (& and |)

Ada: programmer can specify either (short-circuit is specified with 'and then' and 'or else')

FORTRAN 77: short circuiting is there, but any side affected place must be set to undefined

Problem with Short Circuiting

(a > b) || (b++ / 3)

Short-circuit evaluation exposes the potential problem of side effects in expressions

Modern Programming Languages Lecture 44

Control Structure

- Def: A *control structure* is a control statement and the statements whose execution it controls

- *Levels of Control Flow*
  1. Within expressions
  2. Among program units
  3. Among program statements

- *Overall Design Question*

What control statements should a language has, beyond selection and pretest logical loops?

Evolution

- FORTRAN I control statements were based directly on IBM 704 hardware
- Much research and argument in the1960s about the issue
- One important result: It was proven that all flowcharts can be coded with only two-way selection and pretest logical loops

Selection Statements
Design Issues

1. What is the form and type of the control expression?

2. What is the selectable segment form (single statement, statement sequence, compound statement)?

3. How should the meaning of nested selectors be specified?

Single-Way Selection Statement

FORTRAN IF
- IF (boolean_expr) statement

Problem
- *C*an select only a single statement;
- To select more, a goto must be used

FORTRAN example:

```
IF (.NOT. condition) GOTO 20
  ...
  ...
20 CONTINUE
```

ALGOL 60 if:

```
if (boolean_expr) then
  begin
  ...
  end
```

Two-way Selector

***ALGOL 60* if:**

```
if (boolean_expr)
  then statement (the then clause)
  else statement  (the else clause)
```

- The statements could be single or compound

Nested Selectors

Example (Pascal)

```
if ... then
if ... then
    ...
else ...
```

- Which then gets the else?

- Pascal's rule: else goes with the nearest then

Nested Selectors

ALGOL 60's solution - disallow direct nesting

```
if ... then          if ... then
  begin                begin
  if ...               if ... then ...
    then ...             end
    else ...         else ...
  end
```

FORTRAN 77, Ada, Modula-2 solution – closing special words

Example (Ada)

```
if ... then          if ... then
  if ... then          if ... then
    ...                ...
  else                 end if
    ...            else
  end if               ...
```

end if end if

- *Advantage:* flexibility and readability

- Modula-2 uses the same closing special word for all control structures (END)

- This results in poor readability

Multiple Selection Constructs

Design Issues

1. What is the form and type of the control expression?

2. What segments are selectable (single, compound, sequential)?

3. Is the entire construct encapsulated?

4. Is execution flow through the structure restricted to include just a single selectable segment?

5. What is done about un-represented expression values?

Early Multiple Selectors

1. FORTRAN arithmetic IF (a three-way selector)
   - IF (arithmetic expression) N1, N2, N3

   Bad aspects:
   - Not encapsulated
     (selectable segments could be anywhere)
   - Segments require GOTO's

Modern Multiple Selectors

1. Pascal case (from Hoare's contribution to ALGOL W)

```
case expression of
  constant_list_1 : statement_1;
  ...
  constant_list_n : statement_n
end
```

Design Choices

1. Expression is any ordinal type
   (int, boolean, char, enum)
2. Only one segment can be executed per execution of the construct

3. In Wirth's Pascal, result of an un-represented control expression value is undefined
(In 1984 ISO Standard, it is a runtime error)

- Many dialects now have otherwise or else clause

The C and C++ switch

```
switch (expression) {
  constant_expression_1 : statement_1;
  ...
  constant_expression_n : statement_n;
  [default: statement_n+1]
}
```

Any number of segments can be executed in one execution of the construct (there is no implicit branch at the end of selectable segments)

- Trade-off between reliability and flexibility (convenience)

- To avoid it, the programmer must supply a break statement for each segment

Ada's case is similar to Pascal's case, except:

1. Constant lists can include:
   - Subranges e.g., 10..15
   - Boolean OR operators
     e.g. 1..5 | 7 | 15..20

2. Lists of constants must be exhaustive
   - Often accomplished with others clause
   - This makes it more reliable

- Multiple Selectors can appear as direct extensions to two-way selectors, using else-if clauses (ALGOL 68, FORTRAN 77, Modula-2, Ada)

Example (Ada)

```
if ...
  then ...
elsif ...
  then ...
elsif ...
  then ...
  else ...
end if
```

- Far more readable than deeply nested if's
- Allows a boolean gate on every selectable group

Iterative Statements

- The repeated execution of a statement or compound statement is accomplished either by iteration or recursion; here we look at iteration, because recursion is a unit-level control

- *General design Issues for iteration control statements are:*
  1. How is iteration controlled?
  2. Where is the control mechanism in the loop?

Counter-Controlled Loops

Design Issues

1. What is the type and scope of the loop variable?

2. What is the value of the loop variable at loop termination?

3. Should it be legal for the loop variable or loop parameters to be changed in the loop body, and if so, does the change affect loop control?

4. Should the loop parameters be evaluated only once, or once for every iteration?

1. FORTRAN 77 and 90

- Syntax: DO label var = start, finish [, stepsize]

  - Stepsize can be any value but zero
  - Parameters can be expressions

- Design choices:
  1. Loop variables can be INTEGER, REAL, or DOUBLE
  2. Loop variable always has its last value
  3. The loop variable cannot be changed in the loop, but the parameters can; because they are evaluated only once, it does not affect loop control
  4. Loop parameters are evaluated only once

FORTRAN 90's 'Other DO'

- Syntax:
  [name:] DO variable = initial, terminal [, stepsize]

…
END DO [name]

- Loop variable must be an INTEGER

2. ALGOL 60

- Syntax: for var := <list_of_stuff> do statement
  where <list_of_stuff> can have:
  - List of expressions
  - expression step expression until expression
  - expression while boolean_expression

```
for index := 1 step 2 until 50,
          60, 70, 80,
          index + 1 until 100 do

(index = 1, 3, 5, 7, ..., 49, 60, 70, 80,
   81, 82, ..., 100)
```

ALGOL 60 Design choices

1. Control expression can be int or real; its scope is whatever it is declared to be

2. Control variable has its last assigned value after loop termination

3. Parameters are evaluated with every iteration, making it very complex and difficult to read

4. The loop variable cannot be changed in the loop, but the parameters can, and when they are, it affects loop control

3. Pascal

- Syntax:
  for variable := initial (to | downto) final do
  statement

- Design Choices:
  1. Loop variable must be an ordinal type of usual scope
  2. After normal termination, loop variable is undefined
  3. The loop variable cannot be changed in the loop; the loop parameters can be changed, but they are evaluated just once, so it does not affect loop control

4. Ada

- Syntax:

```
for var in [reverse] discrete_range loop
  ...
end loop
```

Ada Design choices

1. Type of the loop var is that of the discrete range; its scope is the loop body (it is implicitly declared)
2. The loop var does not exist outside the loop
3. The loop var cannot be changed in the loop, but the discrete range can; it does not affect loop control
4. The discrete range is evaluated just once

5. **C**

- Syntax:

```
for ([expr_1] ; [expr_2] ; [expr_3]) statement
```

- The expressions can be whole statements, or even statement sequences, with the statements separated by commas
- The value of a multiple-statement expression is the value of the last statement in the expression e.g.

```
for (i = 0, j = 10; j == i;  i++) ...
```

- If the second expression is absent, it is an infinite loop

-C Design Choices

1. There is no explicit loop variable
2. Irrelevant
3. Everything can be changed in the loop
4. Pretest
5. The first expression is evaluated once, but the other two are evaluated with each iteration

- This loop statement is the most flexible

6. C++

- Differs from C in two ways:
  1. The control expression can also be Boolean
  2. The initial expression can include variable definitions (scope is from the definition to the end of the function in which it is defined)

7. Java

- Differs from C++ in two ways:

1. Control expression must be Boolean
2. Scope of variables defined in the initial expression is only the loop body

Logically-Controlled Loops

- Design Issues
  1. Pre-test or post-test?
  2. Should this be a special case of the counting loop statement (or a separate statement)?

Examples

1. Pascal has separate pretest and post-test logical loop statements (while-do and repeat-until)

2. C and C++ also have both, but the control expression for the post-test version is treated just like in the pretest case (while - do and do - while)

3. Java is like C, except the control expression must be Boolean (and the body can only be entered at the beginning- Java has no goto)

4. Ada has a pretest version, but no post-test

5. FORTRAN 77 and 90 have neither

User-Located Loop Control Mechanisms

- Design issues

  1. Should the conditional be part of the exit?
  2. Should the mechanism be allowed in an already controlled loop?
  3. Should control be transferable out of more than one loop?

1. C , C++, and Java - break

- Unconditional; for any loop or switch; one level only (Java's can have a label)
- There is also a continue statement for loops; it skips the remainder of this iteration, but does not exit the loop

2. FORTRAN 90 - EXIT

- Unconditional; for any loop, any number of

levels
- FORTRAN 90 also has a CYCLE, which has the same semantics as C's continue

Examples:

1. Ada - conditional or unconditional exit; for any loop and any number of levels

```
for ... loop
  ...
  exit when ...
  ...
end loop
```

Example (Ada)

```
LOOP1:
  while ... loop
          ...
    LOOP2:
        for ... loop
    ...
    exit LOOP1 when ..
    ...
end loop LOOP2;
  ...
  end loop LOOP1;
```

Unconditional Branching

- All classical languages have it

- Problem: readability
  Edsger W. Dijkstra,
-Go To Statement Considered Harmful,
  CACM, Vol. 11, No. 3, March 1968, pp. 147-148
- Some languages do not have them e.g. Modula-2 and Java
- Advocates: Flexibility
  KNUTH D E, Structured programming with go to statements, ACM Computing Surveys 6, 4 (1974)

Conclusion

Choice of control statements beyond selection and logical pretest loops is a trade-off between language size and writability!

Connection between control statements and program verification is intimate

- Verification is impossible with goto's
- Verification is possible with only selection and logical pretest loops
- Verification is relatively simple with only guarded commands

Modern Programming Languages Lecture 45

Actual/Formal Parameter Correspondence:

1. Positional
2. Keyword

e.g. SORT(LIST => A, LENGTH => N);

Default Parameter Values

```
procedure SORT(LIST : LIST_TYPE;
               LENGTH : INTEGER := 100);
...
SORT(LIST => A);
```

Parameters and Parameter Passing

Semantic Models

- in mode, out mode, in-out mode

Conceptual Models of Transfer

1. Physically move a value
2. Move an access path

Implementation Models

1. Pass-by-value (in mode)

- Either by physical move or access path

2. Pass-by-result (out mode)

- Local's value is passed back to the caller
- Physical move is usually used
- Disadvantages: Collision

Example:

```
procedure sub1(y: int, z: int);
          ...

sub1(x, x);
```

3. Pass-by-reference (in-out mode)

i. Actual parameter collisions

e.g.

```
procedure sub1(a: int, b: int);
```

```
...
sub1(x, x);
```

ii. Array element collisions

e.g.

```
sub1(a[i], a[j]);  /* if i = j  */
```

iii. Collision between formals and globals

Root cause of all of these is: The called subprogram is provided wider access to Non-locals than is necessary

Type checking parameters

- Now considered very important for reliability

FORTRAN 77 and original C: none

Pascal, Modula-2, FORTRAN 90, Java, and Ada: it is always required

- ANSI C and C++: choice is made by the user

Implementing Parameter Passing

ALGOL 60 and most of its descendants use the run-time stack

- Value copied to the stack; references are indirect to the stack

- Result : same

- Reference : regardless of form, put the address on the stack

Design Considerations for Parameter Passing

1. Efficiency
2. One-way or two-way

- These two are in conflict with one another!

Good programming =>
limited access to variables, which means one-way whenever possible

Efficiency =>
pass by reference is fastest way to pass structures of significant size

Concluding Remarks

- Different programming languages for different problems
  Imperative, Declarative, Functional, Object-Oriented, Scripting
- Readability – maintainability

- Side effects
- Mainly because of the assignment statement and aliasing
- Reduces readability and causes errors
- Operand evaluation order
- Data types and data sizes
- Static-Dynamic activation records
- Necessary for recursion

http://www.cs.utep.edu/cheon/cs3360/exam/sample/chap7.php