CS201 Notes for Final Term (Lecture#31-38) preparation

Lecture No. 31

Operator Overloading is quite similar to Function Overloading.

There are two types of operators to overload: unary and binary.

Unary operators are the ones that require only one operator to work. Unary operators are applied to the left of the operand. For example, ^, &, ~ and!.

Binary operators require two operands on both sides of the operator. +, -, *, /, %, =, < and > are examples of binary operators.

A unary operator takes only one operand & one argument (++,--), while binary operator takes two operators & two arguments.

C++ built-in operators work for built-in (primitive) data types but for user defined data types, user has to write his/her own operators.

There are some restrictions while performing Operator Overloading. For example, only existing C++ operators are overloaded without creating a new one in the language.

Also, it should not impact the type, semantics (behavior), arity (number of operands required), precedence(high/low precedence) and associativity (left/right) of the operator.

For binary member operators,operands on the left drives (calls) the operation. Operator functions written as non-members but friends of the class, get both the operands as their arguments.

Operators can be written as non-members and even without making them friends. But this is tedious and less efficient way, therefore, it is not recommended.

Lecture No 33:

Both C and C++ have a set of rules for converting one type to another. These rules are used in the following situations:

- When assigning a value. For example, if you assign an integer to a variable of type long, the compiler converts the integer to a long.
- When performing an arithmetic operation. For example, if you add an integer and a floating-point value, the compiler converts the integer to a float before it performs the addition.
- When passing an argument to a function; for example, if you pass an integer to a function that expects a long.
- When returning a value from a function; for example, if you return a float from a function that has double as its return type.

In all of these situations, the compiler performs the conversion implicitly. The conversion explicit is possible by using a cast expression.

Can we do conversion with objects of our own classes?
The answer is yes. If we go to the basic definition of a class it is nothing but a user defined data type. As class is a user defined data type, we can also define conversion on it.

Lecture No. 34:

Array of Object: A class is a user-defined data type. Objects are instances of classes the way *int* variables are instances of *int*s. The declaration of arrays of user-defined data types is identical to the array of *primitive* data types.
New is called before the constructor, and that *delete* is called after the destructor.

A global array called *pool* that can store all the *Name* objects expected.

Copying a File in the Reverse Order means the last byte of the input file will be the first byte of the output file, second last byte of the input file will be the second byte of the output file until the first byte of the input file becomes the last byte of the output file.

- The *default constructor* is defined by the C++ compiler automatically for every class that has no default constructor (parameterless constructor) defined already.
- The default constructor (parameterless constructor) is called for each element in the array allocated with *new*.

- The *new* operator returns a *void **, accepts a parameter of type *size_t*.
- The *delete* operator returns nothing (*void*) and accepts a pointer of *void ** to the memory block.

- With *new* operator function, a block of memory is allocated first and then constructor is called. (Difference b/w new & delete operator)

- With *delete* operator, destructor of the object is called first and then memory block is deallocated.

- By overloading *new* and *delete* operators, only allocation and deallocation part can be overridden.

- The same pointer that is returned by the *new* operator, is passed as an argument to the *delete* operator. These rules apply to both, if operators (*new* and *delete*) are overloaded as member or non-member operators (as global operators).

- By overloading the array operator ([]), one can implement mechanism to check for array bound.

Lecture No. 35:

A **C++ stream** is a flow of data into or out of a program, such as the data written to ***cout*** or read from ***cin***. cin is an example of an istream. ostream is a general purpose output stream.
Streams are nothing but an ordered sequence of bytes. They allow data to move from one part of the computer to another which may be the screen or key board from and to, or from memory or files on disc and so on.
Byte stream is used to connect the source and the destination.

"Every stream has an associated source and a destination"

Input/output costs and the I/O devices (keyboard, monitor and disc etc) are slower as compared to the speed of the microprocessor and the memory being used. To overcome this speed difference, we use the mechanism, called buffered input/output.

Predefined Stream Objects:

Object Meaning

- cin Standard input
- cout Standard output
- cerr Standard error with unbuffered output.
- clog Standard error with buffered output
- caux Auxiliary (DOS only)
- cprn Printer(DOS only)

All these functions (getline, get, read, unget and peek) are implemented as member functions of the input class.

Lecture No. 36:

The *manipulators* are like something that can be inserted into stream, effecting a change in the behavior.

Non parameterized Manipulators do not take argument to control the formatting of input/output where as *parameterized manipulators* take argument for formatting.

Lecture No.37:

Extraction operator's (>>) Insertion operator (<<)

- If we overload insertion (<<) and extraction (>>) operators then the user of our class, does not need to know the specific names of the functions to input and display our objects

- *Stream insertion (<<) and extraction operators (>>) are always implemented as non-member functions.*

- Operator << *returns a value of type* ostream & *and* operator >> *returns a value of type* istream & *to support cascaded operations.*

- The first parameter to operator << is an ostream & object. cout is an example of an *ostream object*. Similarly first parameter to operator >> is an *istream & object*. cin is an example of an istream object. These *first parameters* are always passed by reference. The compiler won't allow you to do otherwise.

- *For operator >>, the second parameter must also be passed by reference*.

- The second parameter to *operator* << is an object of the class that we are overloading the operator for. Similar is the case for *operator* >>.

Lecture No. 38:

Right-hand side is normally a stream insertion operator.

Static-Keywords:

It refers to something that is stationary, stopped and not moveable. Static as the word implies are variables which exist for a certain amount of time, much longer than that by ordinary automatic variables.

Static data will be created once and initialized once for that class.
Static data members of a class can be public or private.

Most Important Useful Operators, functions in c++:

C++ Strings

Operator	Meaning
=	Assignment
+	Concatenation
+=	Concatenation assignment
==	Equality
!=	Inequality
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
>>	Reads
<<	Prints

HEADER FILES	TYPES (Full Forms)
iostream.h	Include all input and output streams
conio.h	All console input and output functions
math.h	Include all Mathematical functions
stdlib.h	Include all standard library functions
string.h	All string manipulation functions
ctype.h	All character manipulating function
iomanip.h	Include all input and output manipulators
ftype.h	Include all file type functions

Precedence level goes down the table		
Operator name	Associativity	Operators
Scope resolution (included in C++)	left to right	::
Primary	left to right	() [] . -> **dynamic_cast typeid**
Unary	right to left	++ -- + - ! ~ & * **(type_name) sizeof new delete**
Pointer to Member(C++)	left to right	*. ->
Multiplicative	left to right	* / %
Additive	left to right	+ -
Bitwise Shift	left to right	<< >>
Relational	left to right	< > <= >=
Equality	left to right	== !=
Bitwise AND	left to right	&
Bitwise Exclusive OR	left to right	^
Bitwise Inclusive OR	left to right	|
Logical AND	left to right	&&
Logical OR	left to right	||
Conditional	right to left	? :
Assignment	right to left	= += -= *= /= <<= >>= %= &= ^= |=
Comma	right to left	,

strlen - Finds out the length of a string
strlwr - It converts a string to lowercase
strupr - It converts a string to uppercase
strcat - It appends one string at the end of another
strncat - It appends first n characters of a string at the end of another.
strcpy - Use it for Copying a string into another
strncpy - It copies first n characters of one string into another
strcmp - It compares two strings
strncmp - It compares first n characters of two strings
strcmpi - It compares two strings without regard to case ("i" denotes that this function ignores case)
stricmp - It compares two strings without regard to case (identical to strcmpi)
strnicmp - It compares first n characters of two strings, Its not case sensitive
strdup - Used for Duplicating a string
strchr - Finds out first occurrence of a given character in a string
strrchr - Finds out last occurrence of a given character in a string
strstr - Finds first occurrence of a given string in another string
strset - It sets all characters of string to a given character
strnset - It sets first n characters of a string to a given character
strrev - It Reverses a string