CS201 Viva & Paper

Best Preparation By Vu Topper RM

For More Help Contact What's app 03224021365

1. Syntax of Code:

```
#include <iostream>
using namespace std;
int main ()
{
cout<< "Hello World!";
return 0;
}
```

2. Data types:

Datatype with Variable

Declaration: **Int a;**

Initiation: **Int a=10;**

Explanation:

Datatype **(Int),** Variable **(a),**
Assignment operator **(=),** Value **(10),**
End of statement **(;).**

Int: Store numerical data (1,2,3,4……)

String: Store Alphabetic data (A,a,b,c…… Ali)

Float: Store dot(.)value mean percentage (10.12).

Double: Store numerical data (12.400,14.800)

3. Variable:

Koi Bi Alphabetic Name Ho Sakta Hai.

For example, Ali, Ayesha, Hamza, Shan, A, B, C.

4. Function:

Input + Processing + Output

For example (A+B). **Input**(A), **Processing** (+), **Output** (Result). **Mean + is a function**

5. Classes:

3 Type Public, Protected, Private.

Class is a just template, blue print, structure.

Class Khud Kuch Bi Nhi Hoti.

6. Objects:

Object is a property of class and three type of object.

i) Attribute **ii)** Behavior **iii)** Exitance

For example, Ali is a object.

i) Attribute Hight, color, address,

ii) Behavior Walking, eating, sleeping

iii) Exitance Area cover.

7. Array:

Collection of Data and Always Start 0.

Declaration with [ ] brackets. **Honda**

For example, Declaration size of honda [4]:

Array Code:

```
// This program reads the input from user and store it into an array and stop at -1.
#include<iostream.h>
Main ( )
{
int c [ 100 ] ;
int z , i = 0 ;
do {
cout << "Please enter the number (-1 to end input) " << endl;
cin >> z ;
if ( z != -1 )
{
c[ i ] = z ;
}
i ++ ;
} while ( z != -1 && i < 100 ) ;
cout << " The total number of positive integers entered by user is " << i
-1;
}
```

Output:

```
Please enter the number (-1 to end input) 1
2
3
4
5
6
-1
The total number of positive integers entered by user is 6
```

8. Loop:

Three types:

i) While Loop ii) Do While Loop iii) For Loop

i) While Loop code:

Syntax

```
while (condition) {
  // code block to be executed
}
```

Example

```
int i = 0;
while (i < 5) {
  cout << i << "\n";
  i++;
}
```

Output:

```
0
1
2
3
4
```

ii) Do While Loop code:

Syntax

```
do {
  // code block to be executed
}
while (condition);
```

Example

```
int i = 0;
do {
  cout << i << "\n";
  i++;
}
while (i < 5);
```

Output:

0
1
2
3
4

iii) For Loop code:

Syntax

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

Example

```
for (int i = 0; i < 5; i++) {
  cout << i << "\n";
}
```

Output:

0
1
2
3
4

9. Condition:

Three types:

i) If ii) If else iii) Switch Statement

i) If Code:

Syntax

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

Example

```
int x = 20;
int y = 18;
if (x > y) {
  cout << "x is greater than y";
}
```

Output: **x is greater than y**

ii) If else code:

Syntax

```
if (condition) {
  // block of code if condition is true
}
else {
  // block of code if condition is false
}
```

Example

```
// Program to check whether an integer is positive or negative
// This program considers 0 as a positive number

#include <iostream>
using namespace std;
int main() {
  int number;
  cout << "Enter an integer: ";
  cin >> number;
  if (number >= 0) {
    cout << "You entered a positive integer: " << number << endl;
  }
  else {
    cout << "You entered a negative integer: " << number << endl;
  }
  cout << "This line is always printed.";
  return 0;
}
```

Output 1:

```
Enter an integer: 4
You entered a positive integer: 4.
This line is always printed.
```

Output 2:

```
Enter an integer: -4
You entered a negative integer: -4.
This line is always printed.
```

10. Pointer:

Pointer is a Reference variable and sign of pointer **&**

11. Operator:

(+,-,*,/,)

12. Call by value:

In call by value, original value is not modified.
In call by value, value being passed to the function is locally stored by the function parameter in stack memory location. If you change the value of function parameter, it is changed for the current function only. It will not change the value of variable inside the caller method such as main ().

Let's try to understand the concept of call by value in C++ language by the example given below:

```
#include <iostream>
using namespace std;
void change(int data);
int main()
{
int data = 3;
change(data);
```

```
cout << "Value of the data is: " << data<< endl;
return 0;
}
void change(int data)
{
data = 5;
}
```

Output: Value of the data is **3**

13. Call by reference:

In call by reference, original value is modified because we pass reference (address).
Here, address of the value is passed in the function, so actual and formal arguments share the same address space. Hence, value changed inside the function, is reflected inside as well as outside the function.

Note: To understand the call by reference, you must have the basic knowledge of pointers.

Let's try to understand the concept of call by reference in C++ language by the example given below:

```
#include<iostream>
using namespace std;
void swap(int *x, int *y)
{
int swap;
swap=*x;
*x=*y;
*y=swap; }
int main()  {
int x=500, y=100;
swap(&x, &y);  // passing value to function
cout<<"Value of x is: "<<x<<endl;
```

```
cout<<"Value of y is: "<<y<<endl;
return 0;  }
```

Output: Value of x is: 100. Value of y is: 500

14. Constructors:

A constructor in C++ is a **special method** that is automatically called when an object of a class is created.
To create a constructor, use the same name as the class, followed by parentheses ():

```
class MyClass {     // The class
public:             // Access specifier
   MyClass() {      // Constructor
     cout << "Hello World!";
   }
};
int main() {
  MyClass myObj;    // Create an object of My Class (this will call the constructor)
  return 0;
}
```

Output: Hello World

Note: The constructor has the same name as the class, it is always public, and it does not have any return value.

Ek Class K Under Us Ki Value Ko Set Karni Ko constructor Kehtay Hai.Class ka name constructor ka name same hota hai or constructor this ka key use karta hai .

15. Destructors:

A destructor is a member function that is invoked automatically when the object goes out of scope or is explicitly destroyed by a call to delete. A destructor has the same name as the class, preceded by a tilde (~). For example, the destructor for class String is declared: ~String ()

Declaring destructors

Destructors are functions with the same name as the class but preceded by a tilde (~).
Several rules govern the declaration of destructors. Destructors:

Do not accept arguments.
Do not return a value (or **void**).
Cannot be declared as **const**, **volatile**, or **static**. However, they can be invoked for the destruction of objects declared as **const**, **volatile**, or **static**.
Can be declared as **virtual**. Using virtual destructors, you can destroy objects without knowing their type — the correct destructor for the object is invoked using the virtual function mechanism. Note that destructors can also be declared as pure virtual functions for abstract classes.

16. Parameters:

The parameter is referred to as the **variables that are defined during a function declaration or definition**. These variables are used to receive the arguments that are passed during a function call. These parameters within the function prototype are used during the execution of the function for which it is defined.

// C code to illustrate Parameters

```
#include <stdio.h>
// Mult: Function definition
// a and b are the PARAMETERS
int Mult(int a, int b){
```

```
        // returning the multiplication
        return a * b;
}
// Driver code
int main ()
{
        int num1 = 10, num2 = 20, res;
        // Mult() is called with
        // num1 & num2 as ARGUMENTS.
        res = Mult(num1, num2);
        // Displaying the result
        printf("The multiplication is %d", res);
        return 0;}
```

Argument	Parameter
When a function is called, the values that are passed during the call are called as arguments.	The values which are defined at the time of the function prototype or definition of the function are called as parameters.
These are used in function call statement to send value from the calling function to the receiving function.	These are used in function header of the called function to receive the value from the arguments.
During the time of call each argument is always assigned to the parameter in the function definition.	Parameters are local variables which are assigned value of the arguments when the function is called.
They are also called Actual Parameters	They are also called Formal Parameters

```
int num = 20;
Call(num)
// num is argument
```

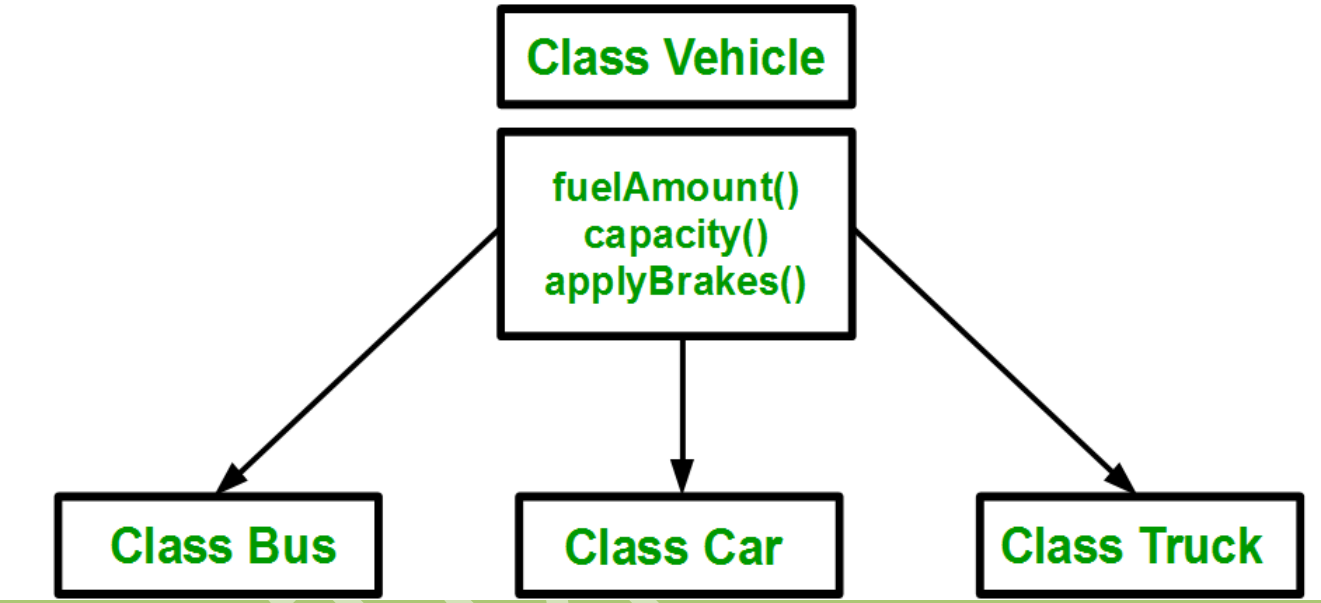
```
int Call(int rnum)
{
  printf("the num is %d", rnum);
}

// rnum is parameter
```

17. Inheritance:

Is a relationship

Inheritance is **a feature or a process in which, new classes are created from the existing classes**. The new class created is called "derived class" or "child class" and the existing class is known as the "base class" or "parent class". The derived class now is said to be inherited from the base class.



18. Aggregation:

In C++, aggregation is **a process in which one class defines another class as any entity reference**. It is another way to reuse the class. It is a form of association that represents HAS-A relationship.

```
#include <iostream>
using namespace std;
class Address {
public:
string addressLine, city, state;
Address(string addressLine, string city, string state)
{
```

```
this->addressLine = addressLine;
this->city = city;
this->state = state;
}
};
class Employee
{
private:
Address* address;  //Employee HAS-A Address
public:
int id;
string name;
Employee(int id, string name, Address* address)
{
this->id = id;
this->name = name;
this->address = address;
}
void display()
{
cout<<id <<" "<<name<< " "<<
address->addressLine<< " "<< address->city<< " "<<address-
>state<<endl;
}
};
int main(void) {
Address a1= Address("C-146, Sec-15","Noida","UP");
Employee e1 = Employee(101,"Nakul",&a1);
e1.display();
return 0;
}
```

Output: 101 Nakul C-146, Sec-15 Noida UP

19. Composition:

composition in C++ is also referred to as **object composition**. The object composition concept work on the model of has-a relationship among two different objects. Complex objects are often referred to as parent components while objects which are simpler or smaller are often referred to as child components.

- Composition in C++ is also referred to as object composition.
- The object composition concept work on the model of **has-a** relationship among two different objects.
- Complex objects are often referred to as parent components while objects which are simpler or smaller are often referred to as child components.

20. Polymorphism:

A student of C++ needs to know that polymorphism refers to **the ability of a C++ function or object to perform in different ways, depending on how the function or object is used**. Polymorphism is one of the main features of object-oriented programming.

21. Function overloading:

C++ lets you specify more than one function of the same name in the same scope. These functions are called overloaded functions, or overloads. Overloaded functions enable you to supply different semantics for a function, depending on the types and number of its arguments.