Object Oriented Programming With C++

By Shailee Shah
Assistant professor
President Institute of Computer Application

What is c++ ?

- It is an object oriented programming language.
- It is developed by Bjrane stroustrup at AT&T bell laboratories in 1980's.
- C ++ is an extension of C with addition.
- C++ is a superset of C.
- C++ adds classes , inheritance ,function overloading ,operator overloading.

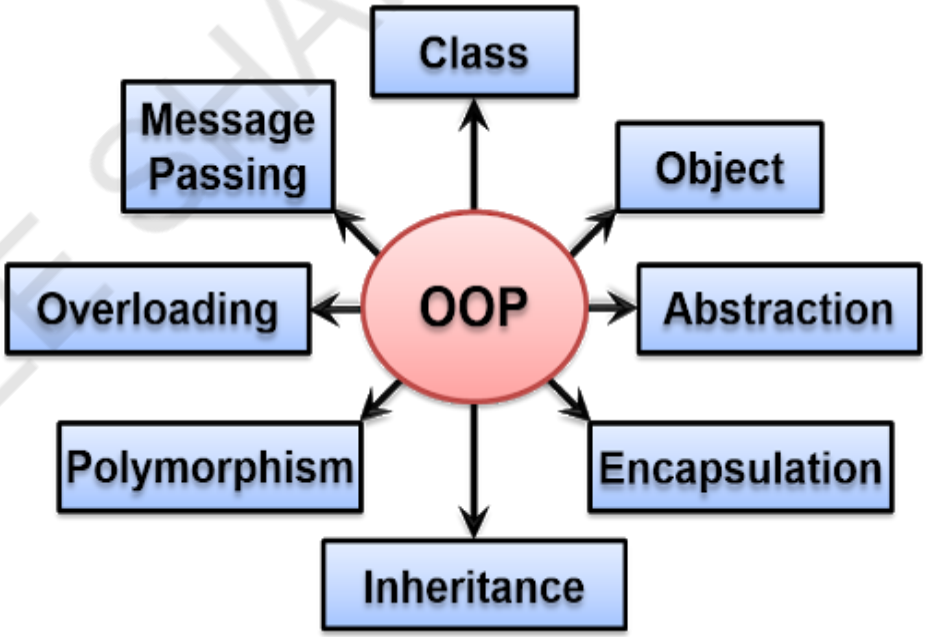
C VS C++

NO.	C	C++
1	C is a procedural programming language.	C++ is an **object oriented programming language**.
2	In C language, the solution is achieved through a sequence of procedures or steps. Therefore, C is a function driven language.	C++ can model the whole solution in terms of objects and that makes the solution better organized. C++ is an object driven language.
3	Concept of **virtual functions** is not present in C.	Concept of **virtual functions** is present in C++.
4	Operator overloading is not possible in C.	C++ allows **operator overloading**.
5	Data in C functions is not secured. Data can be easily accessed by other external functions.	All the data in C++ can be put inside objects. This provides better data security.
6	C is a *middle level language*.	C++ is a **high level language**.
7	C programs are divided into **modules and procedures**.	C++ programs are divided into **classes and functions**.
8	C programs use top-down approach.	C++ programs use bottom-up approach.

#	C	C++
9	In C, the ***main()* function** can be called from other functions.	In C++, the ***main()* function** can not be called from other functions. The program execution begins from *main()*function.
10	C language does not provide the feature of **namespace**.	Namespaces are available in C++.
11	Inheritance is not possible in C.	Inheritance is possible.
12	In C, all the variables must be declared at the beginning of a scope.	C++ allows declaring variables anywhere within the scope. This allows us to declare a variable when we use it for the first time.
13	In C, function overloading is not possible	**Function overloading** is possible in C++
14	Standard Input in C is received through *scanf()* function whereas standard output is given through*printf()* function.	C++ uses *cin>>* and *cout>>* as standard input and output functions respectively.
15	C programs are saved in files with extension *.c*	C++ programs are saved in files with extension *.cpp*
16	In C, **polymorphism** is not possible.	C++ offers polymorphism.

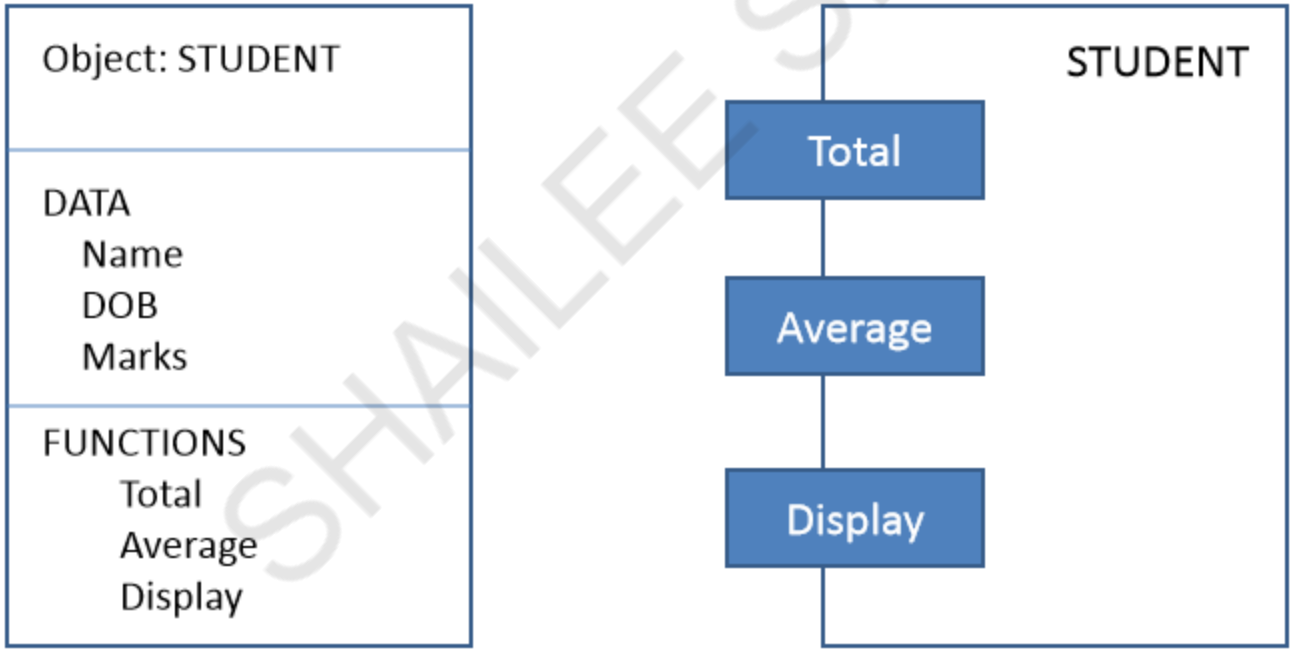
Features of Object Oriented Programming(OOP)

- Objects
- Classes
- Data Abstraction and Encapsulation
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing



❑ Objects:

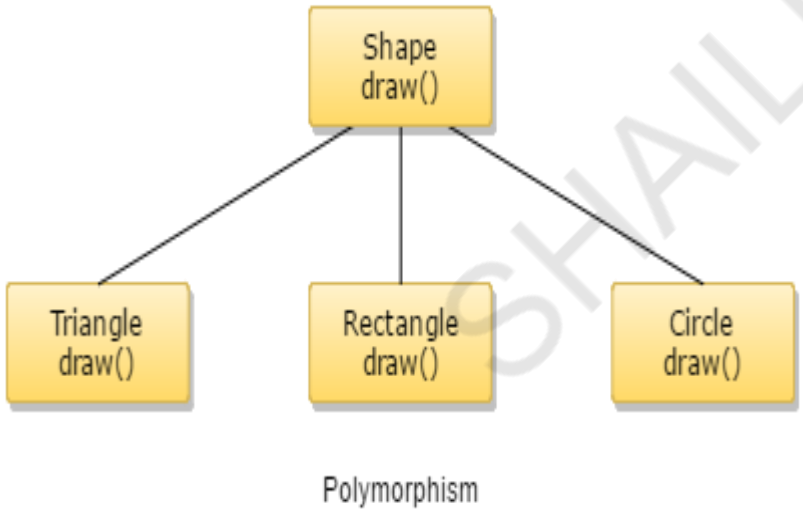
- This is the basic run-time entities in an object oriented programming.
- That is both data and function that operate on data are bundled as a unit called as object.
- Program objects should be chosen such that they match closely with the real- world objects.
- Objects take up space in the memory and have an associated address like structure in C.



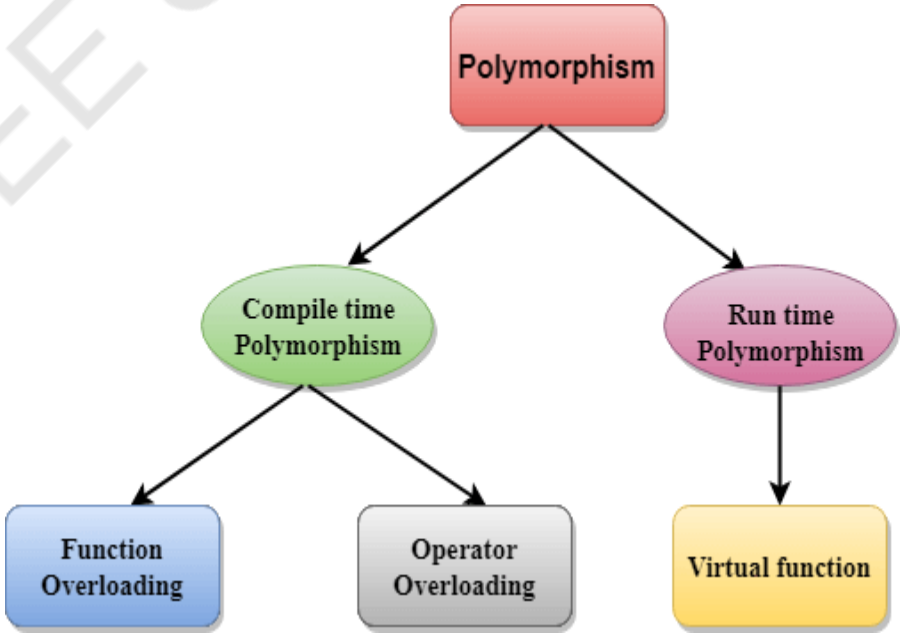
- Classes :
  - The entire set of data and code of an object can be made a userdefined data type with the help of a class.
  - Objects are variable of type class.
  - Once a class has been defined we can create any number of objects belonging to that class.
  - A class is thus a collection of objects of similar type.
  - Classes are user-defined data types and behave like the built-in types of a programming language.
  - e.g: Fruits mango , apple.

- Data Abstraction and Encapsulation :
  - The wrapping up of data and function into a single unit (called class) is known as encapsulation.
  - The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it.
  - It is called data hiding or information hiding.
  - Abstraction refers to the act of representing essential features without including the background details or explanation.
- Inheritance :
  - This is the process by which a class can be derived from a base class with all features of base class and some of its own.
  - This increases code reusability.

- Polymorphism :
  - Polymorphism is another important OOP concept .
  - Polymorphism means the ability to take more than one form.
  - The process of making an operator(like + ,- , * , /) to exhibit different behaviors in different instances is known as **operator overloading.**
  - Using a single function name to perform different type of task is known as **function overloading.**

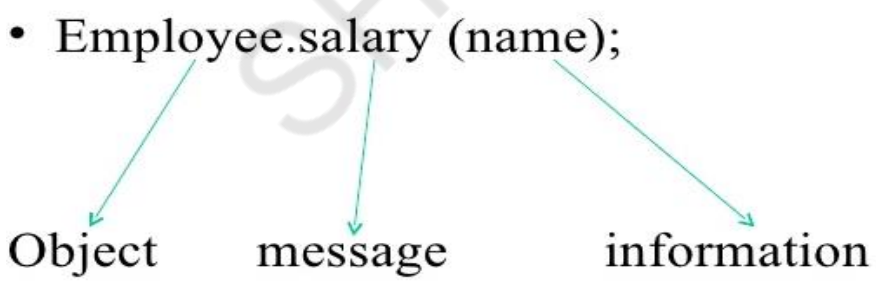


Polymorphism



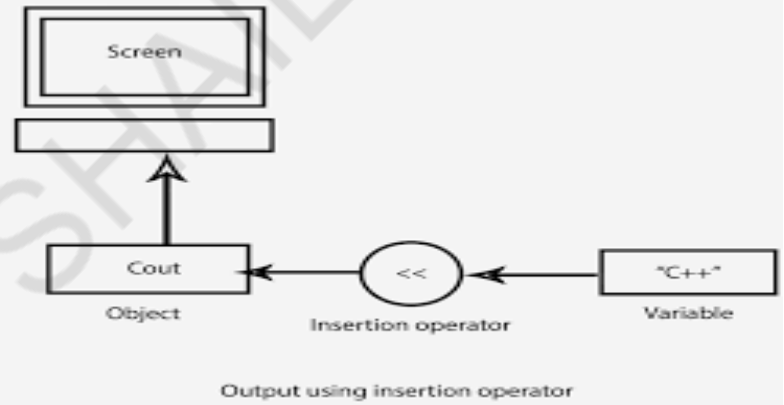
- Dynamic Binding :
  - Dynamic binding also known as late binding, means that the code associated with a given procedure call is not known until the time of the call at Run-Time.

- Message Passing
  - A message for an object is a request for execution of a procedure(FUNCTION) .
  - Invoke a function(procedure) in the receiving object that generates the desired result.
  - Message passing involves :
    1. Specifying name of the object.
    2. Name of the function(message).
    3. Information to be sent.



C++ output operator

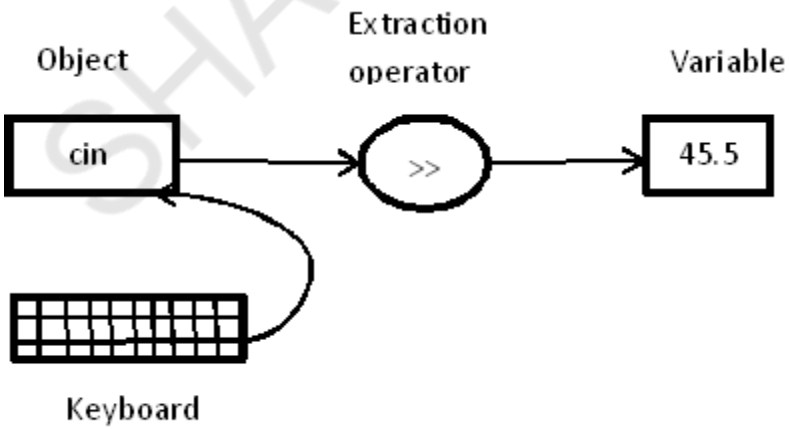
- It is an output statement.
- The identifier cout is a predefined object that represents the standard output stream in c++.
- The operator << is called the insertion /put to operator.
- It inserts the content of the variable.
- Cout is like printf in c.
- Syntax:
  - cout<<“HELLO,WELCOME TO SEM 3”;



Output using insertion operator

C++ input operator

- It is an input statement.
- The identifier cin is a predefined object that represents the standard input stream in c++.
- The operator >> is called the extraction / get from operator.
- It extracts the value from the keyboard and assigns it to a variable.
- Cin is like scanf in c.
- Syntax:
  - int number1,number2;
  - Cin>> number1;



C++ Standard Library Function

The C++ Standard Library provides a rich collection of functions for performing common mathematical calculations, string manipulations, character manipulations, input/output, error checking and many other useful operations. This makes the programmer's job easier, because these functions provide many of the capabilities programmers need. The C++ Standard Library functions are provided as part of the C++ programming environment.

> *Header file names ending in .h are "old-style" header files that have been superseded by the C++ Standard Library header files.*

C++ Standard Library header file	Explanation
<iostream>	Contains function prototypes for the C++ standard input and standard output functions. This header file replaces header file <iostream.h>.
<iomanip>	Contains function prototypes for stream manipulators that format streams of data. This header file replaces header file <iomanip.h>.
<cmath>	Contains function prototypes for math library functions. This header file replaces header file <math.h>.
<cstdlib>	Contains function prototypes for conversions of numbers to text, text to numbers, memory allocation, random numbers and various other utility functions. This header file replaces header file <stdlib.h>.

Comments in c++

- C++ uses // symbol for comments.
- Ex:
  - // this is demo of comment .
  - /*…*/

Simple c++ program

```
#include <iostream.h>
void main()
{
	int n1,n2,sum;

	cout<<“enter 2 numbers”<<“\n”;

	cin>>n1>>n2;

	sum=n1+n2;
	 cout<<“SUM IS = “<< sum<<“\n”;

}
```

Output:

Enter 2 numbers

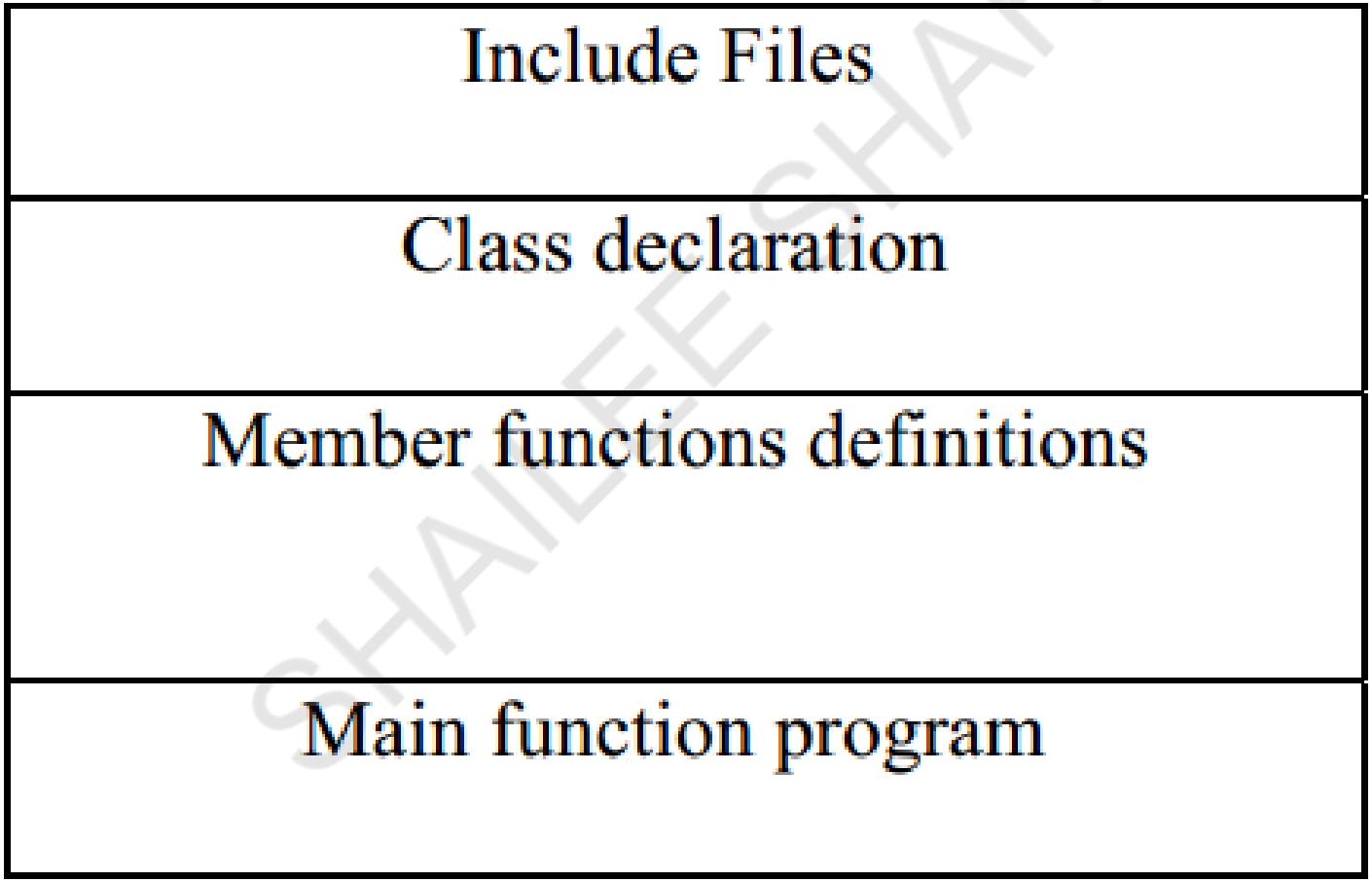
10
20

SUM IS=30

C++ KEYWORDS

asm	continue	float	new	signed	try
auto	default	for	operator	sizeof	typedef
break	delete	friend	private	static	union
case	do	goto	protected	struct	unsigned
catch	double	if	public	switch	virtual
char	Else	Inline	register	template	void
class	enum	int	return	this	volatile
const	extern	long	short	throw	while

Structure Of C++ Language



Variable

- C++ allows the declaration of a variable anywhere in the program.
- Before use we can declared variable ,not required to declare in the starting of the program.
- Ex:
- We can initialize variable dynamically

```
#include <iostream.h>
void main()
{
    int n1,n2;

    cout<<“enter 2 numbers”<<“\n”;
    cin>>n1>>n2;

    int sum;            // declared when needed
    sum=n1+n2;
     cout<<“SUM IS = “<< sum<<“\n”;

    int avg=sum/2;     // initialize variable dynamically
}
```

Reference Variable

- A reference variable can provide alias for a variable.
- If we give reference for sum variable as total ,then both the name can be used interchangeably.
- Syntax:

  Data-type & Reference-name = variable name

- Ex:

  int sum=10+20;

  Int & total=sum;

```
#include <iostream.h>
void main()
{
    int n1=5,n2=7,sum;

    sum=n1+n2;
     cout<<“SUM IS = “<< sum<<“\n”;

    int & total=sum    // reference variable

    cout<<“TOTAL IS = “<< total<<“\n”;
}
```

Output:

SUM IS=12

TOTAL IS=12

Functions in C++

```
void show();                    // Function declaration
main()
{
        .....
        show();                 // Function call
        .....
}
void show()                     // Function Definition
{
        ......
        ......                  // Function Body
}
```


- Function plays an important role in C++ program.
- To reduce the size of a program by calling and using them at different places in the program.

Main Function

- C does not specify any return type for the main() function which is the starting point for the program.
- In C++, the main() return a value of type int to the operating system.

Function Prototyping

- Function prototype describes the function interface to the compiler by giving details such as the number and type of argument and the type of return values.
- C also uses prototyping. But it was introduced first in C++ by stroustrup and the success of this feature inspired the ANSI C committee to adopt it.
- Function Prototype is a declaration statement in the calling program and is of the following form:

type function-name (argument-list);

float volume(int x, float y, float z);

OR

float volume(int, float, float);

Function Body

- After declaration of function we have to create the function body.

Exa.

```
float volume(int x, float y, float z)
{
        return (x*y*z);
}
```

Function call

- After creating the function body we can call this function at any point of time and any where in program.

  float ans = volume(a, b, c);

Inline Function

- Every time a function is called, it takes a lot of extra time in executing a series of instructions for tasks such as,

  jumping to the function , saving registers , pushing arguments into stack ,returning to the calling function.

- When a function is small, a large percentage of execution time may be spent in such overheads.
- C++ has a different solution to this problem.
- To eliminate the cost of call to small function, C++ proposes a new feature called ***inline*** function.
- An inline function is a function that is expanded in line when it is invoked.

Syntax :

```
inline function-header
{
        function body
}
```

example :

```
inline double cube(double a)
{
        return (a*a*a);
}
```

Usually, the function are made inline when they are small enough to be defined in one or two lines.

inline double cube(double a){ return (a*a*a); }

- Remember that the inline function keyword sends a request, not a command, to the compiler.
- The compiler may ignore this request if the function definition is too long or to complicated and compile the function as a normal function.
- Some situations when inline expansion may not work :
  - For function returning values, if a loop, a switch or a goto exists.
  - For function not returning values, if a return statement exists.
  - If function contain *static* variables.
  - If inline functions are recursive.

Default Arguments

- C++ allows us to call a function without specifying all its arguments.
- In such cases, the function assigns a *default values* to the parameter which does not have matching argument in the function call.

Ex:

```
float sum(int a, int b, float c=5.5);
```

Function call :

```
ans = sum(5,6);
```

Note : One important point to be note is that the trailing arguments can have default values and therefor we must add defaults from ***right to left.***

```
int sum (int a, int b=10, float c)          // not valid
int sum (int a, int b=10, float c=10.5)     // valid
```

Function Overloading

- C++ also permits overloading of functions. This means that we can use the same function name with different argument list.

Ex :

```
// Declarations
int add(int a, int b);              //prototype 1
int add(int a, int b, int c);       //prototype 2
double add(int a, double q);    //prototype 3


// Function calls
cout<<add(5, 10);               // uses prototype 1
cout<<add(5,10.0);              // uses prototype 3
cout<<add(5,10, 20);            // uses prototype 2
```

```
// function declaration

int area(int a);
int area(int b,int c);

int main()
{
      cout<<“AREA OF SQUARE IS :”<< area(5);
      cout<<“AREA OF RECTANGLE IS :”<< area(5,10);

      return 0;
};

//function definition

int area(int a)
{
      return (a*a);
}
```

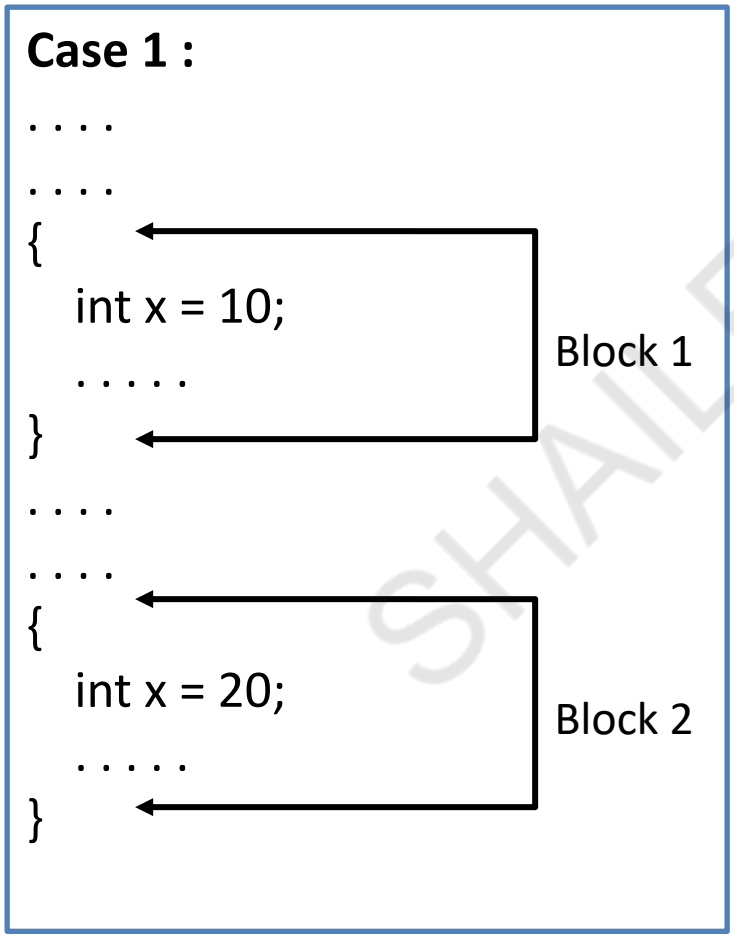
```
int area(int b,int c)
{
            return (b*c);
}
```

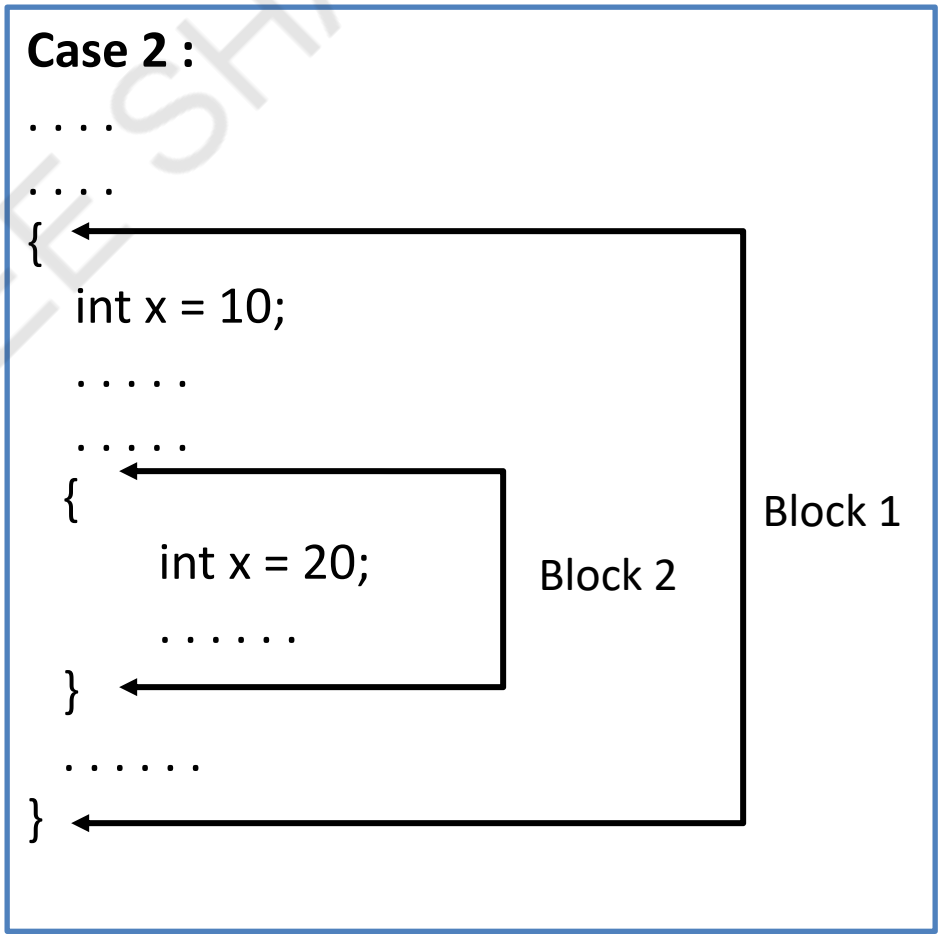
Output:

```
AREA OF SQUARE IS : 25
AREA OF RECTANGLE IS : 50
```

Scope resolution Operator

- Like C, C++ is also a block-structured language.
- Block and scope can be used in constructing programs.
- In different block we can have variable having same name.





- In C, The global version of a variable can't be accessed from within the inner block.
- C++ resolves this problem by introducing a new operator : : called the ***scope resolution operator***
- Syntax :

  : : variable-name

Note : :: m will always refers to the global m.

```
#include <iostream>
int m=10;
void main()
{
        int m=20;
        {
                int k = m;
                int m = 30;
                cout<<"K = "<< k <<"\n";          // k=20
                cout<<"M = "<< m <<"\n";          // m=30
                 cout<<"M = "<< ::m <<"\n";       // m=10
        }
        cout<<"M = "<< m <<"\n";                  // m=20
         cout<<"M = "<< ::m <<"\n";               // m=10
}
```

Object & Classes

Structure

- Unique feature of C language.

Ex :

```
struct student
{
        char name[20];
        int  rno;
        float per;
}

struct student s1;

strcpy(s1.name, “xyz”);
s1.rno = 101;
s1.per = 75.52;
```

Extensions to Structure

- C++ support all the features of structure as defined in C.
- But C++ has expanded its capabilities further to suit its OOP.
- In C++, a structure can have both variable and functions as members.
- It can also declare some of its members as private so that they can not be accessed directly by the external function.
- In C++, the keyword ‘struct’ is omitted in the declaration of structure variable.

```
student S1;        // C++ declaration
```

Class

- C++ incorporates all structure extensions in another user-defined type known as ***class.***
- There is very little syntactical difference between structure and classes in C++.
- The only difference between a structure and a class in C++ is that, by default, the members of a class are *private*, while, by default, the members of structure are *public*.

Specifying a class

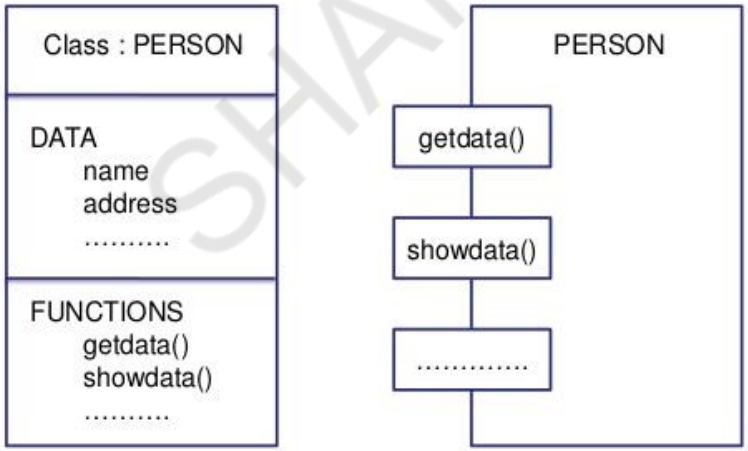
- A class is a way to bind the data and its associated function together.
- It allows the data and function to be hidden from out side world.
- When defining a class, we are creating a new *abstract data type*.
- A class contains two parts :
  - Class declaration
  - Class function definitions

Class

- Syntax:

```
class class_name
{
        private :
                variable declaration;
                function declaration;
        public :
                variable declaration;
                function declaration;
};
```

Visibility Label



Class

- The class members that have been declared as private can be accessed by only from within the class.
- Public member can accessed from outside that class also.
- Data hiding using private declaration, is the key feature of OOP.
- private is optional.
- By default, the member of class are private.
- Only the member function can have access to the private data member and private function.
- Public data member can be accessed from outside of a class.
- The binding of data and function together into a single class-type variable is referred to as *encapsulation*.

Thank you