

C++ Programming

Trainer : Akshita Chanchlani

Email: akshita.chanchlani@sunbeaminfo.com



Functions / User Defined Functions

- It is a set of instructions written to gather as a block to complete specific functionality.
- Function can be reused.
- It is a subprogram written to reduce complexity of source code
- Function may or may not return value.
- Function may or may not take argument
- Function can return only one value at time
- Function is building block of good top-down, structured code function as a "black box"
- **Writing function helps to**
 - improve readability of source code
 - helps to reuse code
 - reduces complexity
- **Types of Functions**
 - Library Functions
 - User Defined Functions



User Defined Functions

- **Function declaration / Prototype / Function Signature**

<return type> <functionName> ([<arg type>...]);

- **Function Definition**

<return type> < functionName > ([<arg type> <identifier>...])

{

}

- **Function Call**

<location> = < functionName >(<arg value/address>);



Inline Function

- C++ provides a keyword *inline* that makes the function as inline function.
- Inline functions get replaced by compiler at its call statement. It ensures faster execution of function just like macros.
- Advantage of inline functions over macros: inline functions are type-safe.
- Inline is a request made to compiler.
- If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time.

When to use Inline function?

- We can use Inline function as per our needs.
- We can use the inline function when performance is needed.
- We can use the inline function over macros.
- We prefer to use the inline keyword outside the class with the function definition to hide implementation details of the function.



Default Arguments

- In C++, functions may have arguments with the default values. Passing these arguments while calling a function is optional.
- A default argument is a default value provided for a function parameter/argument.
- If the user does not supply an explicit argument for a parameter with a default argument, the default value will be used.
- If such argument is not passed, then its default value is considered. Otherwise arguments are treated as normal arguments.
- Default arguments should be given in right to left order.

```
int sum (int a, int b, int c=0, int d=0) {  
    return a + b + c + d;  
}  
• The above function may be called as  
  • Res=sum(10,20);  
  • Res=sum(10,20,40);  
  • Res=sum(10,30,40,50);
```



Function Overloading

- Functions with same name and different signature are called as overloaded functions.
- Return type is not considered for function overloading.
- Function call is resolved according to types of arguments passed.
- Function overloading is possible due to name mangling done by the C++ compiler (Name mangling process , mangled name)
- Differ in number of input arguments
- Differ in data type of input arguments
- Differ at least in the sequence of the input arguments
- Example :
 - `int sum(int a, int b) { return a+b; }`
 - `float sum(float a, float b) { return a+b; }`
 - `int sum(int a, int b, int c) { return a+b+c;;`



Access Specifier

- - If we want to control visibility of members of structure/class then we should use access Specifier.
- Defines the accessibility of data member and member functions
- **Access specifiers in C++**
 1. private(-)
 2. protected(#)
 3. public(+)
- 1. Private - Can access inside the same struct/class in which it is declared Generally data members should declared as private. (data security)
- 2. public - Can access inside the same struct/class in which it is declared as well as inside out side function(like main()). Generally member functions should declared as public.



Scope Resolution Operator (::)

- :: operator is used to bind a member with some class or namespace.
- It can be used to define members outside class.
- Also used to resolve ambiguity.
- It can also be used to access global members.
 - Example :- ::a =10; access global var.
- Scope resolution Operator is used to :
 - to call global functions
 - to define member functions of class outside the class
 - to access members of namespaces



Types of Member Functions within class

- Constructor : object initialization
- Destructor : used to release the resources
- Mutators/setter : modify state of object
- inspector/getter : do not change the state of the object
- facilitator



Constructor

- It is a member function of a class which is used to initialize object.
- Constructor has same name as that of class and don't have any return type.
- Constructor get automatically called when object is created i.e. memory is allocated to object.
- If we don't write any constructor, compiler provides a default constructor.
- Due to following reasons, constructor is considered as special function of the class:
 1. Its name is same as class name.
 2. It doesn't have any return type.
 3. It is designed to call implicitly.
 4. In the life time of the object , it gets called only once per object and according to order of its declaration.
- We can not call constructor on object, pointer or reference explicitly. It is designed to call implicitly.
- We can not declare constructor static, constant, volatile or virtual. We can declare constructor only inline.
- Constructor overloading means inside a class more than one constructor is defined.
- We can have constructors with
 - No argument : initialize data member to default values
 - One or more arguments : initialize data member to values passed to it
 - Argument of type of object : initialize object by using the values of the data members of the passed object. It



Types of Constructor

- Parameterless constructor
 - also called zero argument constructor or user defined default constructor
 - If we create object without passing argument then parameterless constructor gets called
 - Constructor do not take any parameter
- Parameterized constructor
 - If constructor take parameter then it is called parameterized constructor
 - If we create object, by passing argument then parameterized constructor gets called
- Default constructor
 - If we do not define constructor inside class then compiler generates default constructor for the class.
 - Compiler generated default constructor is parameterless.



Thank You

