

# C++ Programming

Trainer : Pradnya S. Dindorkar  
Email: pradnya@sunbeaminfo.com

DESD Aug 2025



# Data Types in C++

- It describes 3 things about variable / object
  - 1. Memory : How much memory is required to store the data.
  - 2. Nature : Which type of data memory can store
  - 3. Operation : Which operations are allowed to perform on data stored inside memory.
- Fundamental Data Types (int,char,float,double)
- Derived Data Types ( Array, Function, Pointer, Union ,Structure)

Two more additional data types that c++ supports are

1. **bool** :- it can take *true* or *false* value. It takes one byte in memory.
2. **wchar\_t** :- it can store 16 bit character. It takes 2 bytes in memory.



# Structure

- Structure is a collection of similar or dissimilar data. It is used to bind logically related data into a single unit.
- This data can be modified by any function to which the structure is passed.
- Thus there is no security provided for the data within a structure.
- This concept is modified by C++ to bind data as well as functions.

## Access Specifier

- By default all members in structure are accessible everywhere in the program by dot(.) or arrow(→) operators.
- But such access can be restricted by applying access specifiers.
  - private: Accessible only within the struct
  - public: Accessible within & outside struct



```
struct Time
{
    int hr;
    int min;
    int sec;
    void printTime() {--}
    void acceptTime() {--}
    void incrementTimeByOneSec() {--}
}
```



Data member - also called field, property, attribute

*struct Time*

{

*int hr;*  
*int min;*  
*int sec;*

Member function-methods , operations, behaviour, message

*void printTime() {--}*  
*void acceptTime() {--}*  
*void incrTime() {--}*

}



# 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.



# Difference - Structure in C & C++

<b>struct in c</b>	<b>struct in c ++</b>
we can include only variables into the structure.	we can include the variables as well as the functions in structure.
We need to pass a structure variable by value or by address to the functions.	We don't pass the structure variable to the functions to accept it / display it. The functions inside the struct are called with the variable and DOT operator.
By default all the variables of structure are accessible outside the structure. ( using structure variable name)	By default all the members are accessible outside the structure, but we can restrict their access by applying the keywords private /public/ protected.
struct Time t1;	struct Time t1;
AcceptTime(struct Time &t1);	t1.AcceptTime(); //function call



# Time class in CPP

Class is collection of logically related data member and member function.



# **Difference between class and struct in cpp**

- all the members of class is by default private and all the members of structure in cpp is by default public.



# Class

- Building block that binds together data & code.
- Program is divided into different classes
- Class is collection of data member and member function.
- Class represents set/group of such objects which is having common structure and common behaviour.
- Class is logical entity.
- Class has
  - Variables (data members)
  - Functions (member functions or methods)
- By default class members are private( not accessible outside class scope)
- Classes are stand-alone components & can be distributed in form of libraries
- Class is blue-print of an object.



# Object

- Object is an instance of class.
- Entity that has physical existence, can store data.
- An entity, which get space inside memory is called object.
- Object is used to access data members and member function of the class
- Process of creating object from a class is called instantiation.
- **Object has**
  - ***State of object (Data members)***
    - Value stored inside object is called state of the object.
    - Value of data member represent state of the object.
  - ***Behavior of object (Member function)***
    - Set of operation that we perform on object is called behaviour of an object.
    - Member function of class represent behaviour of the object.
  - ***Identity of object (Unique address)***
    - Value of any data member, which is used to identify object uniquely is called its identity.
    - State of object may be same But its address can be **Unique** considered as its identity.



## Few Points to note

- Member function do not get space inside object.
- If we create object of the class then only data members get space inside object. Hence size of object is depends on size of all the data members declared inside class.
- Data members get space once per object according to the order of data member declaration.
- Structure of the object is depends on data members declared inside class.
- Member function do not get space per object rather it gets space on code segment and all the objects of same class share single copy of it.
- Member function's of the class defines behaviour of the object.





# **Empty Class :-**

A class which do not have any Data member or member function is called as the Empty class.

- We can create an object of a empty class.
- Size of an object of a empty class is 1 byte.



# this pointer

## Structure in C

```
struct time {  
    int hr, min, sec;  
};  
void accept( struct time *p) {  
    scanf("%d:%d:%d", &p->hr,  
        &p->min, &p->sec);  
}  
Main()  
{  
    struct time t;  
    accept(&t);  
}
```

## class in C++

```
class time {  
    int hr, min, sec;  
    void accept(){  
        scanf("%d:%d:%d",&hr, &min,  
            &sec);  
    }  
};  
Main()  
{  
    time t;  
    t.accept();  
}
```



# this pointer

- To process state of the object we should call member function on object. Hence we must define member function inside class.
- If we call member function on object then compiler implicitly pass address of that object as a argument to the function implicitly.
- To store address of object compiler implicitly declare one pointer as a parameter inside member function. Such parameter is called *this* pointer.
- *this* is a keyword. "*this*" pointer is a constant pointer.
- *this* is used to store address of current object or calling object.
- The invoking object is passed as implicit argument to the function.
- *this* pointer points to current object i.e. object invoking the member function.
- Thus every member function receives *this* pointer.
- Following functions do not get *this* pointer:
  1. Global Function
  2. Static Member function
  3. Friend Function.



# Types of Member Functions within class

- Constructor : object initialization
- Destructor : used to release the resources
- Mutators / setter : modify state of object
- inspector/getter : read the data member but do not change the state of the object
- Facilitator : Provide extra facility to work with object

Sunbeam



# 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 is called as copy constructor.



# 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.



# Destructor

- It is a member function of a class which is used to release the resources.
- It is considered as special function of the class
  - Its name is same as class name and always precedes with tilde operator( ~ )
  - It doesn't have return type or doesn't take parameter.
  - It is designed to call implicitly.
- Destructor calling sequence is exactly opposite of constructor calling sequence.
- Destructor is designed to call implicitly.
- If we do not define destructor inside class then compiler generates default destructor for the class.
- Default destructor do not release resources allocated by the programmer. If we want to release it then we should define destructor inside class.



# Thank You

Sunbeam

