

# OOP using C++ Programming

Trainer : Pradnyaa S. Dindorkar

Email: [pradnya@sunbeaminfo.com](mailto:pradnya@sunbeaminfo.com)

CCAT Exam : 9 Questions concept based



# What we are going to cover in this module?

- 1 : Introduction to C++
- 2 : Function features
- 3 : class and object
- 4 : cin ,cout
- 5 : static, const, friend, reference
- 6 : memory management [new- delete]
- 7 : Object oriented concept
- 8 : Association
- 9 : Inheritance
- 10: virtual function
- 11: Advance C++ feature



# Limitations of C Programming

- C is said to be process oriented, structured programming language.
- When program becomes complex, understating and maintaining such programs is very difficult.
- Language don't provide security for data.
- Using functions we can achieve code reusability, but reusability is limited. The programs are not extendible.



# Few Real Time Applications of C++

- Games
- GUI Based Application (Adobe)
- Database Software (MySQL Server)
- OS (Apple OS)
- Browser( Mozilla)
- Google Applications(Google File System and Chrome browser)
- Banking Applications
- Compilers
- Embedded Systems(smart watches, MP3 players, GPS systems)



# Characteristics of Language

1. It has own syntax
2. It has its own rule( semantics )
3. It contain tokens:
  - Identifier
  - Keyword
  - Constant/literal
  - Operator
  - Separator / punctuators
4. It contains built in features.
5. We use language to develop application( CUI, GUI, Library )



# Classification of high level Languages

## 1. Procedure Oriented Programming language( POP )

- ALGOL, FORTRAN, PASCAL, BASIC, C etc.
- "FORTRAN" is considered as first high level POP language.
- All POP languages follows "TOP Down" approach

## 2. Object oriented programming languages( OOP )

- Simula, Smalltalk, Java, C++, C#, Python, R, PHP, Visual Basic.NET, Ruby, Perl, Objective-C, Swift, Scala, Kotlin
- "Simula" is considered as first high level OOP language.
- more than 2000 lang. are OO.
- All OOP languages follows "Bottom UP" approach

## 3. Object based programming languages -

Object based languages supports the usage of object.  
e.g- VB, Javascript



# History of C++

- @1960 ← OOP designed by Alen Kay American computer scientist these are concept, theory not any programing language. It is a process / programming methodology which is used to solve real world problems.
- @1960 ← simula is first oop language designed by Ole-Johan Dahl and Kristen Nygaard. Problem with simula is that performance wise it is very slow.
- @1972 ← C is developed by Dennis Ritchie at AT&T bell laboratories.
- @1979 ← Bjarne Stroustrup develop "C with Classes"
- @1983 ← C++ (rename 'C with classes') with added features. New features were added, including virtual functions, operator overloading ,references
- @1998 ← C++ is standardized by the ANSI  
(American National Standards Institute)



# OOPS(Object Oriented Programming Language)

---

- It is a programming methodology to organize complex program in to simple program in terms of classes and object such methodology is called oops.
- It is a programming methodology to organized complex program into simple program by using concept of abstraction , encapsulation , polymorphism and inheritance.
- Languages which support abstraction , encapsulation polymorphism and inheritance are called oop language.





# Main Function

- main should be entry point function of C/C++
- Calling/invoking main function is responsibility of operating system.
- Hence it is also called as Callback function.

[ In cpp extension of file is -> .cpp  
compiler name is -> g++]

Helloworld.cpp



# 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
  
- **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

A function is a group of statements that together perform a task.

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

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

- **Function Call**

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

- **Function Definition**

```
<return type> < functionName > ([<arg type> <identifier>...])  
    {  
        //function body  
    }
```



# 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.
- 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 the inline function when performance is needed.



# 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 :

```
void print ( int n1) { ----- }  
void print (char a) { ----- }  
void print (int a, int b) { ----- }  
void print (char a, int b) { ----- }  
void print (int i, char c) { ----- }
```



# 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.



# Bool and wchar\_t

- **e.g:** `bool val=true;`
- **wchar\_t:** Wide Character. This should be avoided because its size is implementation defined and not reliable.
- Char can take 256 values which corresponds to entries in the ASCII
- Wide char can take on 65536 values which corresponds to UNICODE values
- This data type occupies 2 or 4 bytes depending on the compiler being used.
- Mostly the wchar\_t datatype is used when international languages like Japanese are used.
- L is the prefix for wide character literals and wide-character string literals which tells the compiler that that the char or string is of type wide-char.
- w is prefixed in operations like scanning (**wcin**) or printing (**wcout**) while operating wide-char type.





# 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



- ✓ Introduction and History of C++
- ✓ OOPS (Object Oriented Programming Language)
- ✓ Function overloading
- ✓ Default Argument
- ✓ Inline function
- ✓ Bool and wchar\_t
- ✓ Struct in C



Cpp is \_\_\_\_\_ language.

- a. Procedure Oriented Programming
- b. Object Oriented Programming
- c. Object based Programming
- d. Tag based Programming

Cpp is \_\_\_\_\_ language.

- a. Procedure Oriented Programming
- b. Object Oriented Programming**
- c. Object based Programming
- d. Tag based Programming



\_\_\_\_\_ is the first Object Oriented Programming language.

- a. Ada
- b. COBOL
- c. Pascal
- d. Simula

\_\_\_\_\_ is the first Object Oriented Programming language.

- a. Ada
- b. COBOL
- c. Pascal
- d. Simula**

Which of the following data type is added in CPP?

a.char\_t

b.wchar

c.uchar

d.wchar\_t



Which of the following data type is added in CPP?

a.char\_t

b.wchar

c.Uchar

**d.wchar\_t**





Overloaded functions are \_\_\_\_\_.

- a. Very long functions that can hardly run
- b. One function containing another one or more functions inside it
- c. Two or more functions with the same name but different number of parameters or type
- d. Functions having different return data type .



Overloaded functions are \_\_\_\_\_.

- a. Very long functions that can hardly run
- b. One function containing another one or more functions inside it
- c. Two or more functions with the same name but different number of parameters or type**
- d. Functions having different return data type



Which value will it take when both user and default values are given?

- a) user value
- b) default value
- c) garbage value
- d) defined value



Which value will it take when both user and default values are given?

- a) **user value**
- b) default value
- c) garbage value
- d) defined value



Why would you want to use inline functions?

- A. To decrease the size of the resulting program
- B. To increase the speed of the resulting program
- C. To simplify the source code file
- D. To remove unnecessary functions



Why would you want to use inline functions?

- A. To decrease the size of the resulting program
- B. To increase the speed of the resulting program**
- C. To simplify the source code file
- D. To remove unnecessary functions



Which of the following is a valid inline for function foo?

- A. inline void foo() {.....}
- B. void foo() inline {.....}
- C. inline:void foo() {.....}
- D. void foo() {.....}inline;



Which of the following is a valid inline for function foo?

**A. inline void foo() {.....}**

B. void foo() inline {.....}

C. inline:void foo() {.....}

D. void foo() {.....}inline;





---

# Thank You

