Agenda

- Access Specifier in structure
- Inline Function
- Function Overloading
- Default Argument Function
- Namespace
- Console I/O
- Class
- Object
- Naming Convention
- Menu Driven Code
- this pointer

Access Specifier in Structure

- 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

Inline Function

- Managing function activation record is a job of compiler.
- If we give call to the function then compiler need to create Stack Frame and push it into stack. Upon returning control back to the calling function it needs to destroy stackframe from stack. In other words giving call the function is overhead to the compiler.
- If we want to reduce compilers overhead then we should use inline function.
- C++ provides a keyword inline that makes the function as inline function.
- If we declare function inline then compiler do not call function rather it replaces function call by function body.
- As Inline functions get replaced by compiler at its call statement. It ensures faster execution of function just like macros.
- Inline is request to the compiler.
- In following cases, function is not considered as inline:
  1. If we use loop(for/while) inside function
  2. If we implement function using recursion
  3. If we use jump statement inside function
- In case of modular approach, we can use inline keyword in either declaration, definition or both places.
- We can not declare main function static, constant, virtual or inline.
- Advantage of inline functions over macros: inline functions are type-safe.
- If we define member function inside class then function are by default considered as inline.
- If we want to make member function inline whose definition is global then we must explicitly use inline keyword.
- We can not divide inline function code in multiple files.

Naming Convention

- The naming convention used for software development are

1. Camel Case Convention

- In this case, except word, first Character of each word must be in upper case.
- Consider following example.
    - main()
    - parseInt()
    - showInputDialog
    - addNumberOfDays(int days)
- We should use this convention for
    - Data member
    - Member function
    - Function Parameter
    - Local and global variable

2. Pascal Case Convention

- In this case, including first word, first character of each word must be in upper case.
- Consider following example
    - System
    - StringBuilder
    - NullPointerException
    - IndexOutOfBoundsException
- We should use this convention for
    - Union Name
    - Structure Name
    - Class Name
    - Enum Name

3. Convention For macro and constant

- Name of constant, enum constant and macro should be in upper case.

```
#define NULL 0
#define EOF -1
#define SIZE 5

const float PI = 3.142;

enum ShapeType
{
    EXIT, LINE, RECT, OVAL
};
```

4. Naming Convention for namespace

- Name of the namespace should be in lowercase.

```
namespace collection
{
    class Stack
    {
    };
}
```

Class

- Class is a logical entity.
- It is a collection of data member and member function.
- Structure and behavior of an object depends on class hence class is considered as a template/model/blueprint for an object.
- Class represents encpasulation.
- Example: Mobile Phone,Laptop,Car

Object

- It is physical entity.
- Object is a variable/instance of a class.
- An entity, which get space inside memory is called object.
- With the help of instantiation we achive abstraction.
- Example: Nokia 1100, MacBook Pro, sMaruti 800

Characteristics of object

- Object defines 3 things

1. State

- Value stored inside object is called state of the object.
- Value of data member represent state of the object.

2. Behavior

- Set of operation that we perform on object is called behavior of an object.
- Member function of class represent behavior of the object.

3. Identity

- Value of any data member, which is used to identify object uniquly is called its identity.
- If state of object is same the its address can be considered as its identity.

Object Size

- If we create object of the class then only non static data members get space inside object.
- Hence size of object is depends on size of all the non static data members declared inside class.
- Member function do not get space inside object.
- Data members get space once per object according to the order of data member declaration.
- 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.
- Object of an empty class is 1 byte.

Access Specifiers

- If we want to control visibility of members of structure/class then we should use access specifier.
- Access specifiers in C++

1. private(-)
    - visiblity only within the class
2. protected(#)
    - visibility in the derived classes
3. public(+)
    - visibility every where on structure/class object

- In C++, structure members are by default considered as public and class members are by default considered as private

Namespace

- If we want to access value of global variable then we should use scope resolution operator(::)

```
int num1 = 10;
int main()
{
    int num1 = 10;
    printf("value of local num1 = %d\n", num1);
    printf("value of global num1 = %d\n", ::num1);
    return 0;
}
```

- Namespace in C++ language is used:

1. To avoid name clashing/collision/ambiguity.
2. To group functionally equivalent/related types together.

- Namespace can contain:

1. Variable
2. Function
3. Types[structure/union/class]
4. Enum

5. Nested Namespace

- Namespaces are used to organize code into logical groups and to prevent name collisions that can occur especially when your code base includes multiple libraries.s
- We can not instantiate namespace.
- If we want to define namespace then we should use namespace keyword.
- namespaces can only be defined in global or namespace scope. In other words, we can not define namespace inside function/class.
- If we want to access members of namespace then we should use namespace name and scope resolution operator.
- If name of the namespaces are diffrent then we can give same/diffrent name to the members of namespace.
- If name of the namespaces are same then name of members must be different.
- We can define namespace inside another namespace. It is called nested namespace.
- If we define member without namespace then it is considered as member of global namespace.
- If we want to access members of namespace frequently then we should use using directive.

```
namespace na
{
int num1 = 10;
}

int main( void )
{
printf("Num1 : %d\n",na::num1);
return 0;
}
```

Console Input and OutPut Operation

- C++ provides an easier way for input and output.
- Console Input -> Keyborad
- Console Output -> Monitor
- Console = Keyboard + Monitor
- iostream is standard header file of C++.

1. cout

- cout is external object of ostream class.
- cout is member of std namespace and std namespace is declared in iostream header file.
- cout represents monitor.
- An insertion operator(<<) is designed to use with cout.

2. cin

- cin is an external object of istream class.
- cin is a member of std namespace and std namespace is declared in iostream header file.
- Extraction operator(>>) is designed to use with cin object.

- cin represents keyboard.

Menu driven code

- When we want to execute the code in continious manner and want to execute the code based on users choice then we can write a menu driven code.

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

Default Argument Function

- 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.

this pointer

- 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 implitly 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.
- General type of this pointer is:
  - Classname * const this;