C++ Programming

Trainer : Pradnyaa S Dindorkar

Email: pradnya@sunbeaminfo.com

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

Namespace

- To prevent name conflicts/ collision / ambiguity in large projects
- to group/ organize functionally equivalent / related types together.
- We can not instantiate namespace.
- It is designed to avoid name ambiguity and grouping related types.
- We can not define namespace inside function/class.
- We can not define main function inside namespace.
- Namespace can contain:
  1. Variable
  2. Function
  3. Types[structure/union/class]
  4. Enum
  5. Nested Namespace

Note :
•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.

cin and cout

- C++ provides an easier way for input and output.
- Console Output : Monitor
  - iostream is the standard header file of C++ for using cin and cout.
  - cout is external object of ostream class.
  - cout is member of std namespace and std namespace is declared in iostream header file.
  - cout uses insertion operator(<<)
- Console Input : Keyboard
  - cin is an external object of istream class.
  - cin is a member of std namespace and std namespace is declared in iostream header file.
  - cin uses Extraction operator(>>)
- The output:
  - cout << “Hello C++”;
- The input:
  - cin >> var;

Complex class :- Ex = 5+j7

Data member = real , imaginary

member functions = complex()
complex(int r,int i)
acceptComplexNumber()
printComplexNumber()
~complex()

Example Scope Resolution

```
class complex {
int real, imag;
public: complex();
void show();
};
```

complex.h

```
complex::complex() {
            real = imag = 0;
}
void complex::show() {
            cout<<real<<imag;
}
```

complex.cpp

```
main()
{
            complex obj;
            obj.show();
}
```

Program.cpp

Modular Approach

- "/usr/include" directory is called standard directory for header files.
- It contains all the standard header files of C/C++
- If we include header file in angular bracket (e.g #include<filename.h>) then preprocessor try to locate and load header file from standard directory only(/usr/include).
- If we include header file in double quotes (e.g #include“filename.h") then preprocessor try to locate and load header file first from current project directory if not found then it try to locate and load from standard directory.

Constant in C++

- We can declare a constant variable that cannot be modified in the app.
- If we do not want to modify value of the variable then const keyword is used.
- constant variable is also called as read only variable.
- The value of such variable should be known at compile time.
- In C++ , Initializing constant variable is mandatory
- const int i=3; //VALID
- Const int val; //Not ok in c++

Constant data member

- Once initialized, if we do not want to modify state of the data member inside any member function of the class including constructor body then we should declare data member constant.
- If we declare data member constant then it is mandatory to initialize it using constructors member initializer list.

```
class Test
{
private:
		const int num1;
public:
		Test( void ) : num1( 10 ) //OK
		{
				//this->num1 = 10; //Not OK
		}
};
```

Const member function

- The member function can declared as const. In that case object invoking the function cannot be modified within that member function.
- We can not declare global function constant but we can declare member function constant.
- If we do not want to modify state of current object inside member function then we should declare member function as constant.
- void display() const; , void printData() const;
- Even though normal members cannot be modified in const function, but *mutable* data members are allowed to modify.
- In constant member function, if we want to modify state of non constant data member then we should use **mutable keyword.**
- We can not declare following function constant:
  1. Global Function
  2. Static Member Function
  3. Constructor
  4. Destructor

Reference

- Reference is derived data type.
- It alias or another name given to the existing memory location / object.
  - Example : int a=10; int &r = a;
  - In above example a is referent variable and r is reference variable.
  - It is mandatory to initialize reference.
- Reference is alias to a variable and cannot be reinitialized to other variable
- When ‘&’ operator is used with reference, it gives address of variable to which it refers.
- Reference can be used as data member of any class

Reference

- We can not create reference to constant value.
  - int &num2 = 10; //can not create reference to constant value
- Reference is internally considered as constant pointer hence referent of reference must be variable/object.

```
int main( void )
{
        int num1 = 10;
        int &num2 = num1;
        cout<<"Num2 : "<<num2<<endl;
        return 0;
}
```

pass arguments to function, by value, by address or by reference.

- **In C++, we can pass argument to the function using 3 ways:**

1. By Value
2. By Address
3. By Reference

Difference between Pointers and reference

Pointer

- It is a variable that points to another variable.
- To access the value of a variable with the help of a pointer, we need to do dereferencing explicitly.
- We can create a pointer to pointer
- We can create a pointer without initialization. Create a NULL pointer.

Eg int n=5; int* ptr=&n;

Reference

- It is an alias / secondary name to an already existing memory.
- No need of dereferencing to access a value of a variable with ref.
- We can't create a reference to reference.
- We can't create a ref without initialization NULL ref can't be created.

Eg : int n=5; int& ref=n;

Sum function and Copy Constructor

- Copy constructor is a single parameter constructor hence it is considered as parameterized constructor
- Example: sum of two complex number

  Complex sum(const Complex &c2)

Copy constructor

Complex c2(c1)

or

Complex c2=c1

$$\begin{tikzcd}[column sep=small]
C1 \arrow[r] & 7 \arrow[d] & + & j & 6 \arrow[d] \\
C2 \arrow[r] & 7 & + & j & 6
\end{tikzcd}$$

Write a function in complex class to add 2 complex numbers

C1 → 7+j6

+

C2 → 3+j2

____

C3 → 10+j8

Dynamic Memory Allocation

To allocate memory dynamically then we should use **new** operator

To deallocate that memory we should use **delete** operator.

Dangling pointer :- The pointer which contains, address of deallocated memory.

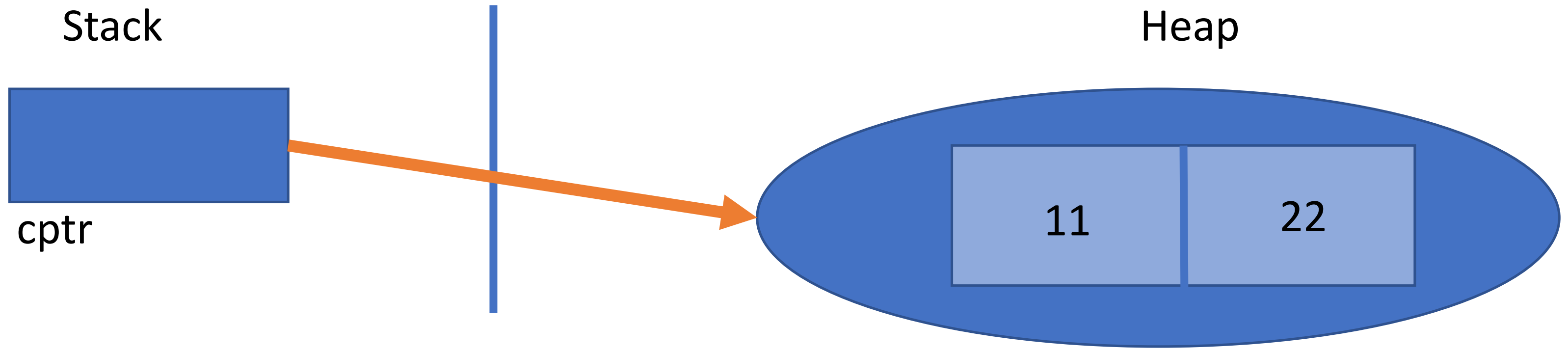
Memory leakage :- When we allocate space in memory, and if we loose pointer to reach to that memory then such wastage of memory is called memory leakage.

```
int main()
{
        int *ptr = new int;                 // allocate memory
        *ptr = 125;                         //Dereferencing
        cout<<"Value:"<<*ptr<<endl;  //Dereferencing
        delete ptr;                         // deallocate memory
        ptr = NULL;
        return 0;
}
```

Heap Based object

```
Complex *cptr=new Complex (11,22) ;
delete cptr;
```

- By using new we are allocating dynamic memory for complex class object .
- Object get created on heap section hence this object is call Heap based or dynamic object.
- Cptr is complex type pointer which is holding the address of that dynamic object.



Difference between malloc and new

new	malloc
new is an operator.	malloc is a function.
new returns a typecasted operator, so no need to do explicit typecasting.	malloc returns void pointer,malloc returns void pointer,cast it explicitly, before use.
We must mention the datatype while allocating the memory with new.	malloc accepts only the exact no. of bytes required, so no need to mention datatype.
When memory is allocated with new, constructor gets called for the object.	When memory gets allocated by malloc function, constructor function does not gets called.

Difference between malloc and new

new	malloc
Memory allocated by new is released by the operator delete.	Memory allocated by malloc is released by function free.
Destructor is called when memory is released with delete.	Destructor is NOT called when the memory is released with free.
To release memory for array syntax is delete[ ]	To release memory for array syntax is free(ptr);
In case new fails to allocate memory, it raises a Run time exception called as bad_alloc.	In case malloc fails to allocate memory it returns a NULL.

Thank You