C++ Programming

Trainer : Akshita Chanchlani

Email: akshita.chanchlani@sunbeaminfo.com

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.

Header Guard

```
#ifndef HEADER_FILE_NAME_H_
#define HEADER_FILE_NAME_H_
//TODO : Type declaration here
#endif
```

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++
- Generally const keyword is used with the argument of function to ensure that the variable cannot be modified within that function.

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;
- 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 exisiting 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
- **Using typedef we can create alias for class whereas using reference we can create alias for object.**

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;
        //int *const num2 = &num1;
        cout<<"Num2 : "<<num2<<endl;
        //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

- If variable is passed by reference, then any change made in variable within function is reflected in caller function.
- Reference can be argument or return type of any function

Copy Constructor

- Copy constructor is a single parameter constructor hence it is considered as parameterized constructor
- Example:
  - **.Complex sum(const Complex &c2)**

Thank You