Classes

# Introduction

The most important feature of C++ is the “class”. It’s significance is highlighted by the fact that Stroustroup initially gave the name “C with classes” to his new language. A class is an extension of the idea of structures used in C.

There are some limitations to the Structures used in C language. The structures can’t be treated like built in data types.

Struct class

{

Int a;

Float b;

}obj1, obj2, obj3;

The values of obj1, obj2, on=bj3 can be easily assigned, but arithmetic operations such as obj3 = obj2 + obj1 ; are not allowed.

Another important aspect is that they do not permit data hiding ie. The variables declared inside a structure can be called by any function anywhere in their scope, that means that the variables declared inside the structures are by default public and can be called by any function.

# Data Hiding

In classes the concept of structures has been further extended to suit the OOP concept developed later on. Data members inside classes and functions can be declared as private, public or protected. These are 3 different visibility modes, where the public data members can be called by any function outside the scope of the class, the private and protected members can only be called by the member functions inside the scope of the class. This uses the OOP concept of encapsulation.

# Objects

Multiple objects can be declared for a class very similar to as they were in C.

Class school

{

Private: //visibility mode

Float rollno;

Protected: //visibility mode

Char name[1000];

Public: //visibility mode

Int I,j;

}

Anish Sachdeva

DTU/2K16/B6/1041

Encapsulation

The OOP concept was built after the production of the C language and as incorporated in the making of the C++ language. There are 4 main OOP concepts which are; Data abstraction, encapsulation, inheritance and polymorphism.

Encapsulation as seen in real life is how we encapsulate or cover something up. The capsule that we take for medicinal purposes contains the medicine inside it, but it is enclosed by the capsule which we ingest and the capsule breaks only at the target site. This is a real world example of encapsulation.

In Computers (or C++ specifically), the wrapping up of functions into a single unit (called class) is known as encapsulation. The data is not accessible to the outside world and only those functions which are allowed to access the data which are wrapped in the class along with the data members. This insulation of the data from direct access by the program is called data hiding or information hiding.

This encapsulation feature plays a very important role in the programs that we use day to day, such as our productivity applications, games and even windows. They contain many classes and data types that the program runs on, but the consumer or the user can’t change, but can inly utilize the program when it’s open. This type of encapsulation of important data from the users keeps it safe from the general public and also from the malicious hackers that try to exploit the vulnerabilities of the code.

Inheritance

Inheritance is the property by which objects of one class acquire the property of another class. It supports the concept of hierarchical classification. For example the bird “robin” is a part of the class “flying bird” which is again a part of the class “bird”. The principle behind this sort of division is that each derived class shares the characteristic of the base class (class from which it was derived).

In OOP, the concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it. This is possible by deriving the new class from the existing one. The new class will have the combined feature of both the classes.

Polymorphism

Polymorphism is another important OOP concept. Polymorphism is a Greek term which means ‘the ability to take more than one form’. An operation may exhibit different behaviours in different instances. Here the program can decide or interpret the code in more than one way at runtime depending on the conditions of the program, e.g.

# Functions

If more than one function is declared with the same name, but different arguments. The program can decide depending on the arguments the user passes which function to call despite them having the same name at call time.

int func (int, int);

float func(int, float);

char func(char, char);

In this case the function to be called will be decided at runtime, depending on the arguments that will be passed.

Overloading

There are 2 types of overloading in C++ language where both utilize the polymorphism concept. They are function overloading and constructor overloading.

# Function Overloading

The concept of function overloading is that the many different functions are created with the same name, but varying arguments and return types. The program during runtime depending on the return type and the arguments passed can select which function to call despite them having the same name (as explained above).

# Constructor overloading

Constructors are declared within a class to initialize a class on runtime of the program, before the declaration of objects. Multiple constructors can be declared within a class with different parameters, so that we can create an object of the class with varying parameters depending o how we want to initialize the variables for that particular object of the class. These different constructors have the same name as that of the class they belong to, so when an object is created with it’s definite set of arguments, the program at runtime decides which constructor to load to declare that object. This concept is called constructor overloading.

Constructor

# Introduction

A constructor is a part of a class that is declare in the public accessibility mode, so that it can be accessed by the functions outside it’s scope. The task of a constructor is to declare the variables and the functions of the class at runtime, as soon as the class is processed. Generally the data members and the function members of the class are declared when the object of the class is created, but if the class includes a constructor, then the data members and the functions members are given memory at the initialization of the program.

# Initialization

class school

{

publc:

school();

int a, b, c;

}

# Parameterized constructors

One van declare more than one constructor inside the class and they will all share the same name i.e. the name of the class itself, but can contain different arguments that can be used to assign values to the data members at initialization or any other purpose. The user while creating the object for this class can pass the required no. of arguments and the program at runtime will create the object while passing the arguments into the specific constructor. This is called constructor overloading.

Destructor

# Introduction

A destructor is declared inside the body of the class and it’s function is to remove all the data occupied by the member functions and the data members of the class once the class goes out of scope, i.e. the class is no longer being used, and all the objects have gone out of scope.

The destructor inside the body of the class can also contain a body of it’s own, i.e. carry code and can perform certain functions before the class is about to go out of scope.

# Initialization

class school

{

public:

school(); //constructor

int roll\_no;

~school() //destructor

{

printf(“The class is going out of scope”);

}

}