2:30 Tuesday July 23

Intro to object oriented programming

object oriented programming or OOP refers to languages that uses objects in programming

OOP aims to implement real world entities like inheritance, data binding, polymorphism etc. in programming

the main aim of OOP is to bind together the data and the function that operate on them so that no other part of the code can access that data except that function

OOPs makes it easier to make modular, re-usable and maintainable software.

as building software rose in complexity, object oriented programming has become the dominant paradigm in software development for developing scalable products.

Features of Object oriented programming are as follows →

1) class
2) objects
3) data abstraction

4) incapsulation
5) inheritance
6) polymorphism
7) dynamic binding
8) message passing.

points
1) program
2) programming language
3) software
4) procedure oriented approach and object oriented approach.

Class -> a class is a collection of functions and variables. the functions which are present in the class are called member function. the variables which are present in the class are called member variables.
a class is created using a 'class' keyword. Members of the class can be either public or private. a class is a logical entity.

the private members of the class are only

accessible to other members of the class (private or public members).

public members are accessible by other members of the class and as well as outside the class.

11:30 Wednesday July 24

Objects in OOP

in order to use a class, we have to create an object of a class. An object is an instance of a class. An object is also called a copy of a class. It is the basic unit of Object oriented programming. When a class is defined, no memory is alocated. When an object of a class is created, memory is then allocated.

an object is used to access the public members of the class. We cannot acess the private members of the class through an object. an object is a physical entity.

Structure of C++ Program consists of four parts

1 -> include header files
2 -> Class declaration
3 -> Class definition
4 -> main program

Incapsulation in OOP

Wrapping up of data and functions in one logical Unit called class is called incapsulation

10:30 Thursday 25 July

Data Abstraction is one of the most essential and important feature of object oriented programming. Data Abstraction refers to providing only the essential information to the outside world and hiding the background details of implementation. in short, Data abstraction means hiding the complexity from the user

Data hiding → in a class, we can have a private section in which we place all members which cannot be accessed outside the class. Some members are placed in the public section which are accessible outside the class using objects this concept of using private and public section in a class implements data hiding.

Inheritance → It is the process by virtue of which one class acquires the features of other class. in this process one class becomes the base class and the other class

becomes the derived class. the derived class acquires the features of base class. inheritance is of following types ->

1) Simple inheritance
2) Multi-level inheritance
3) Multiple inheritance
4) hierarchical inheritance
5) hybrid inheritance

30 July 2024 Tuesday

2:30

Poly morphism -> poly means many and morphism means roles. so polymorphism means when the same thing plays more than one role.

function overloading ->

operator overloading ->

Dynamic binding is in dynamic binding, the code to be executed in response to a function call is decided at runtime.

Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run time.

Message passing → It is a form of communication used in object oriented programming. Objects communicate with one another by sending & recieving information to each other is known as message passing

11:30 31 July Thursday

Control Statements

Control Statements are used to control the flow of execution of the program. They are of the following types

1) if statement
2) if else statement
3) if else ladder
4) nested if else statement
5) switch case statement

— decision control statements

6) for loop
7) While loop
8) do while loop

— loop control statements

(i)→ input a number from a user and find whether the number is prime or not

(ii) find all armstrong no. upto 1000.

(iii) use switch case statement to develop a simple calculator.

1 →
```
#include <iostream>
using namespace std;
int main() {
    int n, f = 1;
    cout << "Enter number :";
    cin >> n;
    for (int i = 2 ; i < n; i++) {
        if ( n % i == 0) {
            cout << "number is not prime!";
            flag = 0;
        }
    }
    if (f) {
        cout << "number is prime!";
    }
}
```

(i)
```
#include <iostream>
#include <math.h>
using namespace std;
int armstrong ( int n) {
    int d, s, t1 = n, t2 = n
    while (t1) {
        d++;
        t1 /= 10 ;
    }
```

```
    while (t2) {
        s += pow(t2 % 10, d);
        t2 /= 10;
    }
    if (s == n) {
        cout << n << " is armstrong." ;
    } else {
        cout << n << " is not armstrong";
    }
}
int main() {
    for(int i=1; i<=1000; i++) {
        armstrong(i);
    }
}
```

Variable -> a variable is an entity in a program whose value can vary during the course of a program

Constants -> a constant is a entity in a program whose value is fixed and doesn't change during the course of a program
a constant is declared by using a keyword 'const'

Operators -> operators are used to perform operations on variables and values. In C++, the following are types of operators ->

1) Arithmetic -> +, -, *, /, %, ++, --
2) logical -> ||, &&, !
3) Assignment -> =, +=, -=, *=, /=, %=, &=, !=, ^=
4) Comparision/Relational -> ==, !=, >, <, >=, <=
5) Bitwise operators -> &, !, ~, ^

* Increment operator -> prefix ++a,
postfix a++

10:30 Tuesday 6 August 2024

Identifier → identifier is the name given to a variable, constant or any entity

Ascii → American Standard code for information interchange

Data types → It specifies the size and type of information the variable will store. a variable or constant in C++ must be of specified datatype

Continue Statement →

Continue statement in C++ is used to take the control to the beginning of the loop

Break

Break keyword will take the control out of the loop in which it is defined.

Missing notes for Monday 12 Aug.

1) Scope resolution operator (::)
2) nesting of member functions

Tuesday Aug 13, 2024

Constructor:- It is used to create a object i.e. It allocates memory to the object. It is used to initialize the values of the variables in the class. It is a special member function. It is different from a function in two aspects.

1) A constructor has the same name as that of class name
2) It has no return type.

If we don't create a constructor of class, then a default constructor with empty Body is automatically created. A constructor is automatically called when the object of a class created.

member variable declaration

```
int a,b;  ✓  Syntax error
int a=10;  ✗  "  "
int b=20;  ✗  "  "
int a;  ✓
int b;  ✓
```

```
#include <iostream>
class Demo
{
    private:
        int a;
        int b;
    public:
        Demo(int x, int y);
        void Display();
};
Demo::Demo(int x, int y)
{
    a = x;
    b = y;
}
void Demo::display()
{
    cout << a << "  " << b;
}
```

```
int main() {
    Demo d1(10,20);
    d1.display();
}
```

Types of Constructors

1) default constructor
2) parameterised constructor
3) Copy constructor

Wednesday 14 August 2024

Copy constructor is a type of constructor that initializes an object using another object of the same class. In simple terms, a constructor which creates an object by initialising it with an object of the same class

```
Class  Demo
{
      private:
        int a;
        int b;
      public:
          Demo(int x, int y);
          Demo ( Demo  &d);
          void  Display();
Demo :: Demo (int x ,int y) {
          a= x;
          b= y;
 }
Demo :: Demo ( Demo  &d) {
          a = d.a ;
          b=  d. y ;
    }
void  Demo :: display ()
  {
      Cout  << a << "   " << b;
  }
int main() {
    Demo d1( 10,20) ;
     d1. display ();
    Demo  d2 ( d1) ;
```

```
    d2.display();
}
```

Destructor → a destructor is reverse of a constructor, it is used to deallocate the memory assigned to the object. It has the same name as that of class name preceded by tilde sign (~)

It doesn't take any parameters and it doesn't have any return type. It is automatically called when the object is deleted. If we don't create a destructor of a class, then a default destructor with empty Body is automatically created. a class can have only one destructor

```
class Demo
{
    private:
        int a;
        int b;
    public:
        Demo(int x, int y);
        ~Demo();    //
            .   .
};
```

```
Demo :: Demo (int x, int y) {
        a = x;
        b = y;
}
Demo :: Demo ( ) {

        Cout << " Object destroyed ";
    }
int main ( ) {
    Demo d1( 10,20) ;
```

Tuesday 20 Aug

1→ Array → an array is a collection of elements of similar datatype. all the elements of the array are stored at contiguous memory allocation. arrays are also called dense or static list because insertion and deletion in case of arrays is difficult. indexing in arrays starts from 0.

Arrays can be of the following types.

1) One dimensional array
2) multi dimensional array

Wednesday 21 Aug 2024

A pointer is a variable that holds the address of another variable of same data type

1. Call by Value → In this method, arguments are passed by copying the value of actual parameters, ensuring the original values remain unchanged. the values is copied to the formal parameters. One is the original copy and the other is the function copy. Any changes made to the parameters within the function don't change the original values outside the function.

Arguments are the values which are assigned to parameters.

Call by Reference -> On this method, the memory address ie reference address of the actual parameters is passed to the function allowing direct access and modification of the original values. The actual and the formal parameters point to the same memory address. any change made to the parameters within the function are reflected in the original values outside the function.

Tuesday 27 Aug

1. Inline function:

C++ provides inline functions to reduce the function call overhead. an inline function is a function that is expanded in line when it is called. When the inline function is called, the whole code of the inline function gets inserted or substituted at the point of inline function call. This substitution is performed by the C++ compiler at compile time. An Inline

function may increase efficiency if it is small.

if we want to make a outside function inline, we have to use the keyword 'inline'.

2. friend function → a friend function can be granted special access to private and protected members of a class in C++. they are not the member function of a class but can access and manipulate the private and protected members of the class for which they are declared as friend.

friend function can be a global function or a member function of another class

make a member function of another class a friend

friend void another class :: func (first & f1)

Borrow notes for Sch 3 Tuesday

Static member variables and functions →

String handling

- -> String
- -> String header < string >
- -> String concatenation + , .append()
- -> String length , .length()
- -> Accessing string .at(), []
- -> special characters
- -> User input strings -> getline (cin, var)

Empty class -> An empty class is a class without data members and member functions. The size of empty class is 1 byte

Nested class ->

A nested class is a class that is declared in another enclosing class. Nested class is a member and as such has the same access rights as any other members. The members of the enclosing class have no special access to members of a nested class.

Borrow notes for 23 September 2024 monday

Polymorphism

Function overloading and Constructor overloading.

Operator overloading → C++ has the ability to provide the operators with a special meaning for a datatype. This ability is known as operator overloading.

We can overload unary and binary operators

```
Class complex {
    public:
        int real;
        int img;
        complex ( int x , int y) {
            real = x;
            img = y;
        }
        Complex ( ) {
        }
        void display ( ) {
            cout << real << "+ " << img << "i";
        }
        complex operator + ( complex &obj )
        {
            complex temp;
            temp.real = real + obj.real;
            temp.img = img + obj.img;
        }
};
```

```
Void main() {
    Complex c1(10, 20);
    c1.display();
    Complex c2(1, 2);
    c2.display();
    Complex c3;
    c3 = c1 + c2;
    c3.display();
}
```