Date : 30/7/2020
Program : 1

# INTRODUCTION TO OOP

## AIM

To familiarize with Object Oriented Programming.

## THEORY

Programming languages evolved from machine language to assembly and to high level language. The usability of high level language made it easier to access by more people. since the programming language changed according to programmers usability and complexity. Then the programs run structuredly or its called structured programming language. soon as task became complecated programmers need more efficient which they could relate easily to real life. Thus came the object oriented programming.

Object oriented programming as name suggest its based on objects. Object is actually a instance of a class, which is a group of data and its function.

Object oriented programming support some features like :-

1. It emphasis on doing thing not procedure.
2. Programs are divided into objects.
3. Data is hidden and can't be accessed by external functions.
4. communication between objects are possible. through functions.
5. Easy to maintain and change any changes in code.

Some of the characteristics of OOPs are:-

(i) Classes and Objects

Classes are like skelton or blueprint of the object we describe.

Object is a instance of this class. Classes are actually group of variable (data) and functions. Objects can communicate with each other with the help of function so the data is safe.

(ii) Abstraction

Abstraction is process of accessing the relevent and most likely behaviour of the data and hiding the rest. Its the act of representing the essential features without including the background details.

(iii) Encapsulation

Its a method of protecting the data from unwanted access or alteration by packaging it. Abstraction is the result of encapsulation. This is implemented in OOP by catergorising data and function into private, protected and public accessibility

(iv) Inheritance.

Its the ability of a class to access or inherit the characters or behaviour from the main class. The class from which data is inherited and used is called child or derived class and the main class from which data is derived is called base class or parent class. It increases readability, reusability and reusability.

(v) Polymorphism.

This is the characteristic of function or operator which works different according to the number of parameters passed to it. Here function could be used differently according to the situation

(vi) Dynamic Binding.

It is in which function call is resolved at runtime. The code is executed as a result of function call which happens

at run time. example for its application is virtual functions in multiple inheritance

vii) Message passing

Objects communicate using messages which is passed only through public functions which uses the data inside the class and returns the data to the object.

Sample program

```
#include<iostream>
using namespace std;
class book {        // private accessibility mode
    int book_no;
    char name[20];
  public :          // public accessibility mode
    getdata() {
        cin >> book_no >> name;
    }
    putdata() {
        cout << "Book no number : " << book_no;
        cout << "\n Book name : " << name;
    }
};      // closing class.
int main() {
    book b;     // creating object b of class
                // book.
```

```
    b.getdata();
    b.putdata();
    return 0;
}
```

3.

This program contains a class 'book' which contain book number stored in 'book_no' and book name in 'name'. Two functions which are in public visibility mode, "getdata()" that accepts book no and book name from user and the other function which displays the data to user ["putdata()"].

- Steps to code C++ in Ubuntu.

1.) Open text editor, write the code in it.
2) save the file as xxyz.cpp in documents or any folder.
3) Open terminal (press Ctl + Alt + T)
4) Move to file using /$ cd Documents/xxyz.cpp
5) type g++ xxyz.cpp.
6) If no errors, are found cursor will point to type again.
7) type If error displayed, correct it and run g++ again.
8) type ./a.out to run the code.
9) Output would be displayed below.

Date : 30/4/2020.
Programe : 2

# STRUCTURES : PHONE NUMBER.

## AIM

A phone number, such as (212) 767-8900 can be thought of as having three parts: the area code (212), the exchange (767), and the number (8900). write a C++ program that uses a structure to store these three parts of a phone number seperately. Call the structure phone. Create two structure variable of type phone. Initialize one, and have the user input number for the other one. Then display both numbers. The interchange might look like this:

Enter your area code, exchange and number:
415 555 1212
My number is (212) 767-8900
Your number is (415) 555-1212

## ALGORITHM.

Step 1 : start
Step 2 : Create a structure named phone with data area code, exchange, number of type int.
Step 3 : Initialize two object of structure namely S1, S2;

step 4 : Print "Enter your area code, exchange and number :"
step 5 : accept the variable data for S2 object.
step 6 : add S1 data directly to object.
step 7 : Display the data in object S1, S2 using the format (area_code) exchange-number.
step 8 : Stop.

RESULT

Succesfully applied the knowledge of structure to store and display the phone number.

CODE

```
#include <iostream>
using namespace std;
struct phone {
    int area_code, exchange, number;
} s1, s2;
int main() {
    s1 = {212, 767, 8900};
    cout << "Enter the area code, exchange and
number in respective order \n";
    cin >> s2.area_code >> s2.exchange >> s2.number;
    cout << '(' << s1.area_code << ')' << s1.exchange
<< '-' << s1.number << endl;
    cout << '(' << s2.area_code << ')' << s2.exchange
<< '-' << s2.number << endl;
    return 0;
}.
```

OUTPUT.

```
Enter the area code, exchange and number in
respective order
415 555 1212.
(212) 767 - 8900
(415) 555 - 1212.
```

Date : 30/7/2020
Program : 3

# CLASS : DATE VALIDATION

## AIM

Define a class date that contains details like year, month and year date. write a c++ program to check validity of the date that you enter and display the next date.

Sample 1 : 02/03/1990.
Valid, Next date : 03/03/1990.

Sample 2 : 29/02/1989.
Not valid.

## ALGORITHM

Step 1 : start
step 2 : create a class 'date'.
step 3 : declare variables date, month, year in private visibility.
step 4 : declare functions input data() which accepts date, month, year from user.
step 5 : declare function isLeapYear which takes year as argument.
step 5.1 : check if year % 4 == 0 and year % 100 != 0. or year % 400 == 0). is true then return true else false.
step 5.2 : declare function check_validity()

step 6.2 : if month < 1 or month > 12 then return false

step 6.3 : if date < 1 or date > 31 then return false

step 6.4 : if month == 2.

step 6.4.1 check if isLeapYear()

if date == 29

month++ ; date = 1 ;

else return (date <= 29).

else date == 28

month ++, date = 1

return (date <= 28).

step 6.5 : if month = 4 or 6 or 9 or 11 then return (date <= 30)

step 6.6 : return true as default.

step 7 : declare function nextDate()

step 7.1 : if date = 31 and month = 12 {

year = year + 1, date = 1, month = 1.

step 7.2 : else if date = 31

month = month + 1, date 1;

step 7.3 : else if month = 4 or 6 or 9 or 11

if date = 30

month ++, date = 1.

step 7.4 : else date = date + 1.

step 7.5 : print date, month, year

step 8 : declare function result()
step 8.1 : if (check_validity() return true
print nextDate()
else
print Not valid.
step 9 : declare main function
step 10 : declare object of class Date.
step 11 : call functions inputdate(), resultC

RESULT

The class phone is created and exceeuted with all its functionality.

CODE

```
#include <iostream>
using namespace std;
class Date {
  int date, month, year;
  public:
  void input_date() {
     cout<<"Enter date, month, year respectively
in this order \n";
     cin>>date>>month>>year;
  }
  bool isLeapYear(int year) {
     if(year%4==0 && year%100!=0 || year%400==0)
        return true;
     else
        return false;
  bool check_validity() {
     if(month<1 || month>12) return false;
     if(date<1 || date>31) return false;
     if(month==2){
        if(isLeapYear(year)) {
           if(date==29){
              month++;
              date=1;
           }
           return(date<=29);
        }
        else {
           if(date==28) {
              month++;
              date=1;
           }
           return(date<=28);
```

```
    }
  }
  if (month==4 || month==6 || month==9 || month
    return (date<=30);
  }
  return true;
}
void nextDate() {
  if (date==31 && month==12) {
      year++;
      date=1;
      month=1;
  }
  else if (date==31) {
      month++;
      date=1;
  }
  else if (month==4 || month==6 || month==9 ||
                month==11) {
    if (date==30) {
      month++;
      date=1;
    }
  }
  else  date++;
  cout<<date<<'/'<<month<<'/'<<year<<endl;
}
void result() {
  if (check_validity()) {
    cout<<"Valid, Next date :";
    nextDate();
  }
```

```
        else {
            cout << "Not valid" << endl;
        }
    }
};
int main() {
    d.input_date();
    d.result();
    return 0;
}
```

OUTPUT

Enter date, month, year respectively in this order

28 2 2020

Valid, next date : 29/2/2020.

Date : 30/7/2020

Program : 4

## CLASS : BANK

### AIM

Define a class to represent bank account. Include members like name of depositor account no. type of account, balance amount in the account. Write C++ program with member functions to a) Assign initial values b) To deposit c) To withdraw d) To display name and balance.

### ALGORITHM

Step1 : Start

Step 2 : create a class named bank with private members as variable and public members as functions.

step 3 : create an object of class bank and ask user for the choice.

step 4 : if choice is 1 accept the details of user

step 5 : if ~~user~~ choice is 2 accept deposit amount, add to balance.

step 6 : if choice is 3 accept withdraw amount, subtract balance.

step 7 : if choice is 4 display the account details.

step 8: if choice is 5 close the program.
step 9: if choice is anything else display wrong choice.
step 10: stop. Repeat step 4 to step 8 unit choice is 5.
step 11: stop.

RESUL

RESULT

Creating bank account details using classes and object was executed successfully.

CODE

```
#include<iostream>
using namespace std;
class bank{
  privale:
  int accNo;
  string firstName, lastName, type;
  float bal, handler;
  public:
  bank(){
    bal = 0.0;
    accNo = 0;
    firstName = '_';
    type = '_';
  }
  void accept(){
    cout<<"Enter your account number\n";
    cin>>accNo;
    cout<<"Enter your first name\n";
    cin>>firstName;
    cout<<"Enter your last name\n";
    cin>>lastName;
    cout<<"Enter type of your account\n";
    cin>>type;
  }
  void deporit(){
    cout<<"\nEnter the amount to deporit\n";
    cin>>handler;
    bal+=handler;
```

```
}
void withdraw(){
    cout<<"\n Your balance is :"<<bal<<endl;
    cout<<"Enter amount to withdraw\n";
    cin>>handler;
    if (handler >bal)
      cout<<"Insufficient balance";
    else
      bal -= handler;
}
void check_acc() {
    cout<<"\n=- Your account details --\n";
    cout<<"Account number \t:"<<accNo<<endl;
    cout<<"Name \t\t:"<<firstName + ' ' + lastName
    cout<<"\nType \t\t:"<< type<<endl;
    cout<<"Balance amount \t"<<bal<<endl<<endl;
}
};
int main() {
    bank b;
    int choice=0;
    do {
     cout<<"\n Welcome to Bank\n";
     cout<<"Enter your choice \n";
     cout<<"1. Create new account\n";
     cout<<" 2. Deposit\n";
     cout<<" 3. Withdraw\n";
     cout<<" 4. Balance\n";
     cout<<" 5. Exit\n";
     cin>>choice;
```

```
switch(choice){
    case 1 : b.accept();
             break;
    case 2 : b.deposit();
             break;
    case 3 : b.withdraw();
             break;
    case 4 : b.Balance();
             break;
    case 5 : exit(0);
             break;
    default : cout << "wrong choice\n";
              break;
  }
 }while(choice != 5);
 return 0;
}.
```

OUTPUT.

```
Welcome to My Bank
Enter your choice
1. create new account
2. Deposit
3. withdraw
4. Balance.
5. Exit.
```

5.

Date : 30/4/2020

program : 5

# CLASS : TRIANGLE

## AIM

Write a class which represent the shape triangle. The member functions should a) check the validity of the triangle b) display the sides c) find the area and display it.

## ALGORITHM

Step 1 : Start

Step 2 : create class triangle with private members as sides and private members as functio

Step 3 : create function accept() to accept sides of triangle.

Step 4 : Also validate() the triangle by checking sum of two side is less than or equal third side.

Step 5 : create display() to display the sides

Step 6 : create area() to find area of triangle using $\sqrt{s(s-a)(s-b)(s-c)}$

Step 7 : in main() create an object of class triangle check if the accept() return true then display the area, side.

Step 8 : else print Invalid triangle.

step 9 : Stop.

RESULT

The class triangle is created successfully and executed with all its functionality.

CODE

```
#include<iostream>
#include<math.h>
using namespace std;
class triangle {
   float a,b,c,s;
   public:
   void int accept() {
       cout<<"Enter the sides of the triangle\n";
       cin>>a>>b>>c;
       return validate();
   }
   bool validate(){
       if (a+c<=b || a+b<=c || b+c<=a) return fal
       else return true;
   }
   void display(){
      cout<<"The sides of the triangle are :"<<a
<<' '<<b<<' '<<c<<endl;
   }
   void area() {
       s=(a+b+c)/2;
       cout<<"Area of the triangle = "<< sqrt(
s*(s-a)*(s-b)*(s-c)<<"unit sq"<<endl;
       }
};
int main()
    triangle t;
    if (t.accept()) {
        t.display();
```

```
        t.area();
    }
    else
        cout<<"Invalid triangle sides"<<end
    }
    return 0;
}
```

OUTPUT:

Enter the side of triangle
6 8 10
The sides of the triangle are: 6. 8 10
Area of the triangle = 24 sq unit.