OOP Lab

Created	@August 27, 2023 1:19 AM
Last edited time	@August 27, 2023 1:30 AM
Created by	Ⓑ Borhan
Tags	CSTE OOP Year 2 Term 1

- **Write a functions power() to raise a number m to a power n. The function takes a double value for m and int value for n, and returns the result correctly. Use a default value of 2 for n to take the function to calculate squares when this argument is omitted. Write a program where main function gets the values of m and n from the user to test the function.**

```
#include<bits/stdc++.h>
using namespace std;

double power(double m, int n=2){
  return pow(m, n);
}

int main(){
  int t;  cin >> t;
  while(t--){
    double m; int n;  cin >> m >> n;
    cout << power(m, n) << "\n";
  }
  return 0;
}
```

- **Write a program to find the largest of three numbers using inline function.**

```
#include<bits/stdc++.h>
using namespace std;

inline int large(int a, int b, int c){
  return max({a, b, c});
}

int main(){
  int a, b, c;  cin >> a >> b >> c;
  cout << large(a, b, c) << "\n";

  return 0;
}
```

- **Consider a shopping list of items for which you place an order with a dealer every month. The list includes details such as the code number and price of each item. You will perform the operations such as adding an item to the list, deleting an item from the list and printing the total value of the order. Write a program to implement these operations using a class with arrays as data members.**

```
#include<bits/stdc++.h>
using namespace std;

const int MAX = 1e5+5;

class Order{
  pair<int, double> arr[MAX];
  int n;

  public:
  Order(){
    n=0;
  }
```

```
  void addOrder(int code, double price){
    arr[n] = make_pair(code, price);
    n++;
  }

  void deleteOrder(int code){
    int ind=-1;
    for(int i=0; i<n; i++){
      if(arr[i].first == code){
        ind = i;  break;
      }
    }

    if(ind == -1){
      cout << "Code not found\n";
      return;
    }

    for(int i=ind; i<n; i++){
      arr[i]=arr[i+1];
    }
    n--;
  }

  void display(){
    cout << n << "\n";
    for(int i=0; i<n; i++){
      int code  = arr[i].first;
      int price = arr[i].second;
      cout << code << " " << price << "\n";
    }
  }

};


int main(){
  cout << "First Input the test Cases\n1 - Adding Items\n2 - Deleting Item \n3 - Display\n";
  int t;  cin >> t;
  Order order;
  while(t--){
    int operation;  cin >> operation;
    int code;   double price;

    if(operation == 1){
      cin >> code >> price;
      order.addOrder(code, price);
    } else if(operation == 2){
      cin >> code;
      order.deleteOrder(code);
    } else{
      order.display();
    }
  }
}
```

- **Write a program to display names, roll number, and grades of 3 students who have appeared in the examination. Declare the class of name, roll number and grade. Create an array of class objects. Read and display the contents of the array.**

```
#include<bits/stdc++.h>
using namespace std;

class Student{
  string name; int roll; double grade;

  public:
  void get(string name, int roll, double grade){
    this->name = name;
    this->roll = roll;
    this->grade = grade;
  }

  void display(){
    cout << name << " " << roll << " " << grade << "\n";
  }
};

int main(){
  Student students[3];
  string name; int roll; double grade;
```

```
  for(int i=0; i<3; i++){
    cin >> name >> roll >> grade;
    students[i].get(name, roll, grade);
  }

  for(int i=0; i<3; i++){
    students[i].display();
  }
}
```

- **Write a program to perform the addition of time in the hour and minutes format. Use a class time and a function sum( ) that takes two objects as arguments.**

```
#include<bits/stdc++.h>
using namespace std;

class Time{
  int hour, minute;

  public:
  Time(){
    hour=minute=0;
  }

  void get(){
    cin >> hour >> minute;
  }

  void sum(Time t1, Time t2){
    int totalHour = t1.hour + t2.hour;
    int totalMinute = t1.minute + t2.minute;
    this->hour = totalHour + (totalMinute/60);
    this->minute = totalMinute%60;
  }

  void display(){
    cout << hour << " hours " << minute << " minutes\n";
  }
};

int main(){
  Time t1, t2, t3;

  t1.get();
  t2.get();
  t3.sum(t1, t2);

  t1.display();
  t2.display();
  t3.display();

  return 0;
}
```

- **Define a class to represent a bank account. Include the following members:**
  Data members
  i. Name of the depositor
  ii. Account number
  iii. Type of account
  iv. Balance amount in the account
  Member functions
  i. To assign initial values
  ii. To deposit an amount
  iii. To withdraw an amount after checking the balance
  iv. To display name and balance
  Write a program to implement these operations.

```
#include<bits/stdc++.h>
using namespace std;

class BankAccount{
  string name, type, account_number;
  int balance;

  public:

  void get(string n, string t, string num, int balance){
    name = n;
    type = t;
    account_number = num;
    this->balance = balance;
  }

  void deposit(int amount){
    balance  += amount;
    cout << "The deposit of " << amount << " taka has been executed successfully.\n";
  }

  void withdraw(int amount){
    if(amount > balance){
      cout << "You have not sufficient balacne to withdraw " << amount << " \n";
      return;
    }
    balance -= amount;
    cout << "A withdrawal of "  << amount << " taka was completed successfully.\n";
  }

  void display(){
    cout << "Name : " << name << "\n";
    cout << "Balance : " << balance << "\n";
  }
};


int main(){
  string name, acc_number, type; int balance;
  cin >> name >> type >> acc_number >> balance;

  BankAccount b;
  b.get(name, type, acc_number, balance);

  b.display();

  b.deposit(200);
  b.display();

  b.withdraw(299);
  b.display();

  b.withdraw(100);

  return 0;
}
```

- **Define a class to represent a bank account. Include the following members:**
  Data members
  i. Name of the depositor
  ii. Account number
  iii. Type of account
  iv. Balance amount in the account
  Member functions
  i. To assign initial values
  ii. To deposit an amount
  iii. To withdraw an amount after checking the balance
  iv. To display name and balance
  Write a program to implement these operations.

```
#include<bits/stdc++.h>
using namespace std;
```

```
const int MAX = 1e4+5;

class BankAccount{
  string name[MAX], type[MAX], account_number[MAX];
  int balance[MAX];
  int n;
  public:
  BankAccount(){
    n=0;
  }

  void get(string nm, string t, string num, int balance){
    name[n] = nm;
    type[n] = t;
    account_number[n] = num;
    this->balance[n] = balance;
    n++;
  }

  int check(string acc_num){
    int ind = -1;
    for(int i=0; i<n; i++){
      if(account_number[i]==acc_num){
        ind=i; break;
      }
    }

    if(ind == -1) {
      cout << "Acount is not found.\n";
    }
    return ind;
  }

  void deposit(string acc_num, int amount){
    int ind = check(acc_num);
    if(ind == -1) return;

    balance[ind]  += amount;
    cout << "The deposit of " << amount << " taka in " << acc_num << " has been executed successfully.\n";
  }

  void withdraw(string acc_num, int amount){
    int ind = check(acc_num);
    if(ind == -1) return;

    if(amount > balance[ind]){
      cout << "You have not sufficient balacne to withdraw " << amount << " \n";
      return;
    }
    balance[ind] -= amount;
    cout << "A withdrawal of "  << amount << " taka was completed successfully.\n";
  }


  void display(string acc_num){
    int ind = check(acc_num);
    if(ind == -1) return;

    cout << "Name : " << name[ind] << "\n";
    cout << "Balance : " << balance[ind] << "\n";
  }


};


int main(){
  string name, acc_number, type; int balance;

  BankAccount b;
  for(int i=0; i<10; i++){
    cin >> name >> type >> acc_number >> balance;
    b.get(name, type, acc_number, balance);
  }

  int q;  cin >> q;
  cout << "1. Display \n2. Deposit\n3. Withdraw\n";

  while(q--){
    int op; cin >> op;

    if(op==1){
      cin >> acc_number;
      b.display(acc_number);
    } else if(op==2){
      cin >> acc_number >> balance;
      b.deposit(acc_number, balance);
```

```
    } else if(op == 3){
      cin >> acc_number >> balance;
      b.withdraw(acc_number, balance);
    }
  }

  return 0;
}
```

- **Write a class to represent a vector (a series of float values). Include member functions to perform the following tasks:**
  i. To create the vector
  ii. To modify the value of a given element
  iii. To multiply by a scalar value
  iv. To display the vector in the form (10, 20, 30, …)
  Write a program to implement these operations.

```
#include<bits/stdc++.h>
using namespace std;

class Vector{
  int size;
  float *p;

  public:

  Vector(){
    cin >> size;
    p = new float [size];

    for(int i=0; i<size; i++)
      cin >> p[i];
  }

  void modify(int n, int value){
    p[n-1]=value;
  }

  void multiply(int n){
    for(int i=0; i<size; i++){
      p[i] *= n;
    }
  }

  void display(){
    for(int i=0; i<size; i++){
      cout << p[i] << " ";
    }
    cout << "\n";
  }

  ~Vector(){
    delete []p;
  }
};


int main(){
  Vector v;

  v.display();

  v.multiply(10);
  v.display();

  v.modify(2, 6);
  v.display();
  return 0;
}
```

- **Modify the class and program of the problem 6 such that the program would be able to add two vectors and display the resultant vector. (You can pass objects as function arguments).**

```
#include<bits/stdc++.h>
using namespace std;

class Vector{
  int size;
  float *p;

  public:
  Vector(){
    size = 1;
    p = new float[1];
    p[0]=1;
  }
  Vector(int size){
    this->size = size;
    p = new float [size];

    for(int i=0; i<size; i++)
      cin >> p[i];
  }

  void modify(int n, int value){
    p[n-1]=value;
  }

  void multiply(int n){
    for(int i=0; i<size; i++){
      p[i] *= n;
    }
  }

  void display(){
    for(int i=0; i<size; i++){
      cout << p[i] << " ";
    }
    cout << "\n";
  }

  void add(Vector &v1, Vector &v2){

    int sz1 = v1.size;
    int sz2 = v2.size;

    size = max(sz1, sz2);

    int mn = min(sz1, sz2);
    p = new float[max(sz1, sz2)];

    for(int i=0; i < mn; i++){
      p[i] = v1.p[i]+v2.p[i];
    }
    for(int i = min(sz1, sz2); i<size; i++){
      if(sz1 > sz2){
        p[i] = v1.p[i];
      } else p[i]=v2.p[i];
    }
  }

  ~Vector(){
    delete []p;
  }
};


int main(){
  Vector v1(3), v2(2);
  v2.display();

  Vector v3;
  v3.add(v1, v2);
  v3.display();

  return 0;
}
```

- **Create two classes DM and DB which store the value of distances. DM stores distances in metres and centimeters and DB in feet and inches. Write a program that can read values for the class objects and add one objects of DM with another object of DB.**

Use a friend function to carry out the addition operation. The object that stores the results may be a DM object or DB object, depending on the units in which the results are required. The display should be in the format of feet and inches or metres and centimeters depending on the objects on display.

```
#include<bits/stdc++.h>
using namespace std;

class DB;

class DM {
  double m, cm;

  public:
  DM(){
    m=0; cm=0;
  }
  void get(double x, double y){
    m=x; cm=y;
  }
  double fttom(double x){
    return x*0.3048;
  }
  double inchtocm(double x){
    return x*2.54;
  }
  void display(){
    cout << m << " m " << cm << " cm\n";
  }
  friend void add(DM &dm, DB &db);
};

class DB {
  double ft, inch;

  public:
  DB(){
    ft=0; inch=0;
  }
  void get(double x, double y){
    ft=x; inch=y;
  }
  double mtoft(double x){
    return x*3.28;
  }
  double cmtoinch(double x){
    return x/2.54;
  }
  void display(){
    cout << ft << " feet " << inch << " inch\n";
  }
  friend void add(DM &dm, DB &db);
};

void add(DM &dm, DB &db){
  DM d1;
  d1.m = dm.m + d1.fttom(db.ft);
  d1.cm = dm.cm + d1.inchtocm(db.inch);

  DB d2;
  d2.ft = db.ft + d2.mtoft(dm.m);
  d2.inch = db.inch + d2.cmtoinch(dm.cm);

  d1.display();
  d2.display();
}

int main(){
  DM dm; DB db;

  double x, y;
  cout << "Input m and cm: \n";
  cin >> x >> y; dm.get(x, y);
  cout << "Input feet and inch: \n";
  cin >> x >> y; db.get(x, y);

  add(dm, db);
}
```

- **Consider the long term deposit schemes working in the commercial banks. The banks provide different interest rates for different schemes as well as for different periods of investment. Write a program that contains the class variables (Principal amount, period of investment, Interest rate and Return value of amount) for holding account details and display the principal amount and return value.**

```
#include<bits/stdc++.h>
using namespace std;

class Bank{
  double principle, interest_rate, return_value, year;

  public:
  Bank(){
    cout << "Input Principle amount, interest rate and year of investment\n";
    cin >> principle >> interest_rate >> year;
    return_value = principle*pow(1+(interest_rate/100), year);
  }

  void display(){
    cout << "Principle Amount : " << principle << "\n";
    cout << "Return value: " << return_value << "\n";
  }
};

int main(){
  Bank bn;
  bn.display();

  return 0;
}
```

- **A book shop maintains the inventory of books that are being sold at the shop. The list includes details such as author, title, price, publisher and stock position. Whenever a customer wants a book, the sales person inputs the title and author and the system searches the list and display whether it is available or not. If it is not, an appropriate message is displayed. If it is, then the system displays the book details and requests for the number of copies required. If the requested copies are available, the total cost of the requested copies is displayed; otherwise the message “Required copies not in stock” is displayed.**
  Write a program using a class called books with suitable member functions and constructors. Use new operator in constructors to allocate memory space required.

```
#include<bits/stdc++.h>
using namespace std;

class Books{
  char *title, *author, *publisher;
  float price;
  int stock;

  public:
  Books();
  void get();
  bool search(string t, string a);
  void buy();
  void display(){
    cout << title << " " << author << "\n";
  }
};

Books::Books(){
  title = new char[150];
  author = new char[150];
  publisher = new char[150];
}
void Books::get(){
```

```
  cin >> title >> author >> publisher ;
  cin >> price >> stock;
}
bool Books::search(string t, string a){
  if(title == t && a==author){
    cout << "Title : " << title << "\n";
    cout << "Author : " << author << "\n";
    cout << "Publisher : " << publisher << "\n";
    cout << "Price : " << price << "\n";
    cout << "Stock : " << stock << "\n";
    return true;

  } else{
    return false;
  }
}
void Books::buy(){
  cout << "Input for required copies << " << "\n";
  int x;  cin >> x;

  if(x <= stock){
    cout << x << " copies are available.\n";
    cout << "Total Cost : " << x*price << "\n";
  } else{
    cout << "Required copies not in stock\n";
    return;
  }
}


int main(){
  int n;
  cout << "Total Number of books : \n";
  cin >> n;
  Books books[n];
  for(int i=0; i<n; i++){
    books[i].get();
  }

  char *title = new char[150];
  char *author = new char[150];

  cin >> title >> author;
  bool found = false;
  for(int i=0; i<n; i++){
    if(books[i].search(title, author)){
      books[i].buy();
      found = true;
      break;
    }
  }

  if(!found){
    cout << "Books not found.";
  }

}
```

- **Write a C++ program to overload +, - and = operators.**

```
#include<bits/stdc++.h>
using namespace std;

class Complex{
  double i,j;

  public:
  Complex(){
    i=0; j=0;
  }

  void get(){
    cout << "Input real and imaginary part:\n";
    cin >> i >> j;
  }

  Complex operator+(const Complex &a){
    Complex x;
    x.i = i + a.i;
    x.j = j+a.j;
```

```
    return x;
  }

  void operator=(const Complex &b){
    i=b.i;
    j=b.j;
  }

  Complex operator-(const Complex &a){
    Complex x;
    x.i = i - a.i;
    x.j = j - a.j;
    return x;
  }

  void display(){
    cout << i << (j >= 0 ? "+" : "-") << abs(j) << "i" << "\n";
  }
};

int main(){
  Complex a, b;
  a.get(); b.get();

  Complex c;
  c = a+b;
  c.display();

  c = a-b;
  c.display();

  c=a;
  c.display();
}
```

- **Write a program to manipulate complex numbers using operator overloading.**

```
#include<bits/stdc++.h>
using namespace std;

class Complex{
  double i,j;

  public:
  Complex(){
    i=0; j=0;
  }

  void get(){
    cout << "Input real and imaginary part:\n";
    cin >> i >> j;
  }

  Complex operator+(const Complex &a){
    Complex x;
    x.i = i + a.i;
    x.j = j+a.j;
    return x;
  }

  void operator=(const Complex &b){
    i=b.i;
    j=b.j;
  }

  Complex operator-(const Complex &a){
    Complex x;
    x.i = i - a.i;
    x.j = j - a.j;
    return x;
  }

  Complex operator*(const Complex &a){
    Complex x;
    x.i = (i*a.i - j*a.j);
    x.j = (i*a.j + j*a.i);

    return x;
```

```
  }

  Complex operator/(const Complex &y){
    Complex x;
    double a=y.i; double b=y.j;
    double c=i; double d=j;
    x.i = (c*a + b*d)/(a*a + b*b);
    x.j = (a*d - c*b)/(a*a - c*b);

    return x;
  }


  void display(){
    cout << i << (j >= 0 ? "+" : "-") << abs(j) << "i" << "\n";
  }
};

int main(){
  Complex a, b;
  a.get(); b.get();

  Complex c;
  c = a+b;
  c.display();

  c = a*b;
  c.display();

  c = a/b;
  c.display();

  c=a;
  c.display();
}
```

- **Assume that the test results of a batch of students are stored in three different classes. Class student stores the roll-number, class test stores the marks obtained in two subjects and class result contains the total marks obtained in the test. The class result can inherit the details of the marks obtained in the test and the roll-number of students through multilevel inheritance.**

```
#include<bits/stdc++.h>
using namespace std;

class student{
  protected:
    int roll;

  public:
  student(){
    cout << "Input Roll : \n";
    cin >> roll;
  }
};

class test{
  protected:
  int num1, num2;

  public:
  test(){
    cout << "Input subject 1 and 2 marks respectively\n";
    cin >> num1 >> num2;
  };
};

class result : public student, public test{
  int totalMarks;

  public:
  result() {
    totalMarks = num1+num2;
  }
  void display(){
    cout << "\nRoll : " << roll << "\n";
```

```
    cout << "Marks in two subject : " << num1 << ", " << num2 << "\n";
    cout << "Total Marks : " << totalMarks << "\n";
  }
};

int main(){

  result res;
  res.display();
}
```

- **Assume that a bank maintains two kinds of accounts for customers, one called as savings account and the other as current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.**
  Create a class account that stores customer name, account number and type of account.
  From this derive the classes cur_acct and sav_acct to make them more specific to their requirements. Include necessary member functions in order to achieve the following tasks:
  i. Accept deposit from a customer and update the balance.
  ii. Display the balance.
  iii. Compute and deposit interest.
  iv. Permit withdrawal and update the balance.
  v. Check for the minimum balance, impose penalty and update the balance.

```
#include<bits/stdc++.h>
using namespace std;

class account{
  protected:
  string name, acc_num, acc_type;
  int balance;

  public:
  account(string name, string acc_num, string acc_type, int balance){
    this->name = name;
    this->acc_num = acc_num;
    this->acc_type = acc_type;
    this->balance = balance;
  }

  void deposit(int amount){
    balance += amount;
    cout << "A deposit to " << acc_num << " executed successfully\n";
  }

  void display(){
    cout << "Balance : " << balance << "\n";
  }


};

class cur_acct : public account{
  int interest_rate, min_balance, penalty;
  public:
  cur_acct(string name, string acc_num, string acc_type, int balance, int rate, int min_balance, int penalty) : account(name, acc_num,
    interest_rate=rate;
    this->min_balance = min_balance;
    this->penalty = penalty;
  }

  void checkInterest(){
    int interest = (interest_rate/100)*balance;
    balance += interest;
    cout << "New balance : " << balance << "\n";
  }

  void checkPenalty(){
    if(balance < min_balance){
      balance -= penalty;
      cout << "Penalty for having less than minimum balance.\n";
```

```
    } else{
      cout << "No penalty\n";
    }
  }

  void withdraw(int amount){
    if(amount > balance){
      cout << "Your balance is insufficient.\n";
    } else{
      balance -= amount;
      cout << "Withdraw successfull\n";
    }
  }

};

class sav_acct : public account{

  public:
  sav_acct(string name, string acc_num, string acc_type, int balance) : account(name, acc_num, acc_type, balance){

  }

  void withdraw(int amount){
    if(amount > balance){
      cout << "Balance is insufficient to withdraw\n";
    } else{
      balance -= amount;
      cout << "Withdraw Successfully\n";
    }
  }
};

int main(){
  cout << "Saving account \n";
  sav_acct acc1("Borhan", "008", "Saving", 1000);
  acc1.display();
  acc1.deposit(200);
  acc1.display();
  acc1.withdraw(500);
  acc1.display();
  acc1.withdraw(701);

  cout << "\n\nCurrent account \n";
  cur_acct acc2("Borhan", "0100", "Current", 500, 40, 300, 20);
  acc2.display();
  acc2.checkPenalty();
  acc2.deposit(400);
  acc2.withdraw(200);
  acc2.display();
  acc2.withdraw(405);
  acc2.display();
  acc2.checkPenalty();
  acc2.display();

}
```