**M. Ali. Arslan**

**19F-0348**

**Lab # 08**

**Program # 1:**

#include<iostream>

using namespace std;

class Data

{

private:

double balance, interest, interestRate;

int transactions;

static int count;

public:

Data();

void inccount();

void setinterestrate(double);

void makeDeposite(double);

void withdraw(double);

void calint();

int getcount() const;

double getintersetrate() const;

double getBalance() const;

double getInterest() const;

int gettransactions() const;

};

int Data::count = 0;

int main()

{

char choice;

long double amount;

Data Acc;

do

{

system("CLS");

cout << "\t\MENU" << endl;

cout << "---------------------------------" << endl;

cout << "A) Display the account balance " << endl;

cout << "B) Display the number of transaction " << endl;

cout << "C) Display the intrest earned for this period " << endl;

cout << "D) Make the deposite " << endl;

cout << "E) Make a withdrawal " << endl;

cout << "F) Add interest for this period " << endl;

cout << "G) Exit the program" << endl;

cout << "Number of times program has taken choice: " << Acc.getcount() << endl;

cout << "Enter your choice: ";

cin >> choice;

if (choice == 'a')

{

cout << "The current balance is $" << Acc.getBalance() << endl;

Acc.inccount();

system("Pause");

}

else if (choice == 'b')

{

cout << "There have been " << Acc.gettransactions() << " transactions!" << endl;

Acc.inccount();

system("Pause");

}

else if (choice == 'c')

{

cout << "The interest earned for this period is: " << Acc.getInterest() << endl;

Acc.inccount();

system("Pause");

}

else if (choice == 'd')

{

cout << "Enter the amount to be Deposited: ";

cin >> amount;

Acc.makeDeposite(amount);

Acc.inccount();

system("Pause");

}

else if (choice == 'e')

{

cout << "Enter the Amount of Withdrawl: ";

cin >> amount;

Acc.withdraw(amount);

Acc.inccount();

system("Pause");

}

else if (choice == 'f')

{

cout << "Intrest added!" << endl;

Acc.calint();

Acc.inccount();

system("Pause");

}

else if (choice != 'g')

{

cout << "Invalid Entry!" << endl;

system("Pause");

}

} while (choice != 'g');

system("Pause");

return 0;

}

Data::Data()

{

balance = 0;

interest = 0;

interestRate = 0.045;

transactions = 0;

};

void Data::inccount()

{

count++;

};

void Data::setinterestrate(double intrate)

{

interestRate = intrate;

}

void Data::makeDeposite(double amount)

{

balance = balance + amount;

transactions++;

}

void Data::withdraw(double amount)

{

if (amount <= balance)

{

balance = balance - amount;

transactions++;

}

else

cout << "WithDrawl Amount is too Large!" << endl;

}

void Data::calint()

{

interest = balance \* interestRate;

balance = balance + interest;

}

int Data::getcount() const

{

return count;

};

double Data::getintersetrate() const

{

return interestRate;

};

double Data::getBalance() const

{

return balance;

};

double Data::getInterest() const

{

return interest;

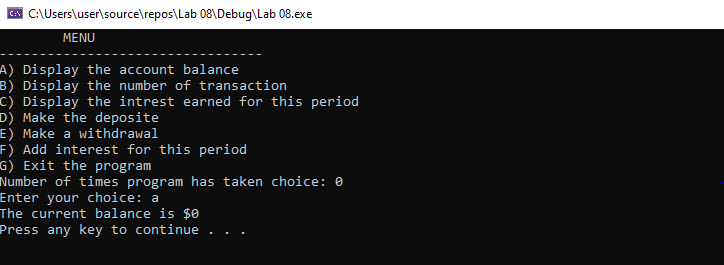
};

int Data::gettransactions() const

{

return transactions;

};



**Program # 2:**

#include<iostream>

using namespace std;

class matrix

{

int\*\* ptr;

int rows;

int cols;

public:

void add(matrix, matrix);

void subtract(matrix, matrix);

void printmatrix();

void input();

matrix()

{

rows = 2;

cols = 2;

}

matrix(int r, int c)

{

rows = r;

cols = c;

}

};

void matrix::input()

{

ptr = new int\* [rows];

for (int i = 0; i < rows; i++)

{

ptr[i] = new int[cols];

}

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

cin >> ptr[i][j];

}

}

}

void matrix::printmatrix()

{

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

cout << ptr[i][j] << " ";

}

cout << endl;

}

}

void matrix::add(matrix A, matrix B)

{

if (rows == cols)

{

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

ptr[i][j] = A.ptr[i][j] + B.ptr[i][j];

}

}

}

else

cout << "Ivalid Entry!" << endl;

}

void matrix::subtract(matrix A, matrix B)

{

if (rows == cols)

{

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

ptr[i][j] = A.ptr[i][j] - B.ptr[i][j];

}

}

}

else

cout << "invalid Entry!" << endl;

}

int main()

{

matrix A(3, 3);

cout << "Enter value of matrix 1: " << endl;

A.input();

cout << "Matrix 1" << endl;

A.printmatrix();

matrix B(3, 3);

cout << "Enter value of matrix 2: " << endl;

B.input();

cout << "Matrix 2" << endl;

B.printmatrix();

matrix C(3, 3);

cout << "After adding matrix:";

C.add(A, B);

C.printmatrix();

matrix d(3, 3);

cout << "After sutracting matrix:";

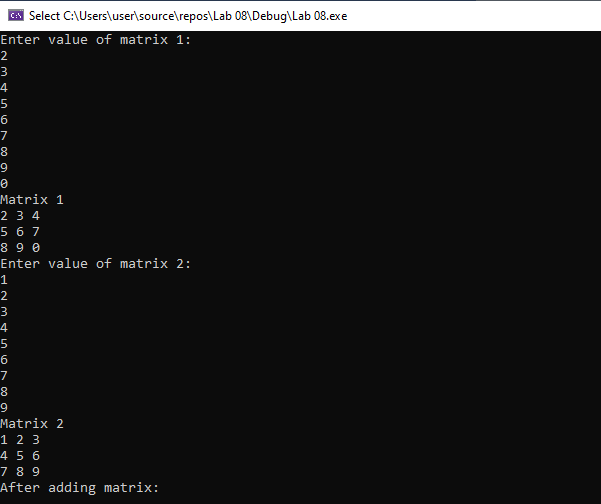
d.subtract(A, B);

d.printmatrix();

system("pause");

return 0;

}



**Program # 3:**

#include<iostream>

using namespace std;

class Area

{

private:

int length, width, area;

static int count;

public:

Area(int, int);

static int times();

void Dispaly();

};

int main()

{

int num1, num2;

cout << "Enter Length: ";

cin >> num1;

cout << "Enter Width: ";

cin >> num2;

Area cal(num1, num2);

system("pause");

return 0;

}

int Area::times()

{

count++;

return count;

};

void Area::Dispaly()

{

cout << "The Area is: " << area << endl;

cout << "The Number of times the area calculated is: " << count << endl;

};

int Area::count = 0;

Area::Area(int length, int width)

{

Area::length = length;

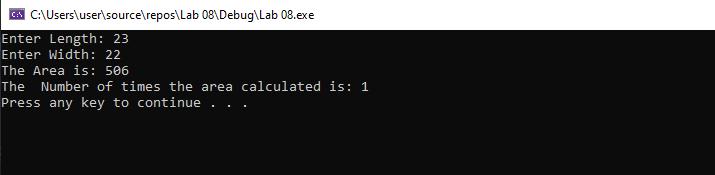
Area::width = width;

area = Area::length \* Area::width;

times();

Dispaly();

};



**Program # 4:**

#include<iostream>

#include<string>

using namespace std;

class person

{

private:

string Name, favoritesports;

int\* Age;

bool Employeestatus;

public:

person();

person(const person&);

void setnamesport();

void Display()const;

};

int main()

{

person p1;

p1.setnamesport();

person p2 = p1;

cout << "Object p2" << endl;

p2.Display();

cout << "New object p1" << endl;

p1.setnamesport();

cout << "Object p2" << endl;

p2.Display();

system("pause");

return 0;

};

person::person()

{

Name = "null";

Age = new int;

\*Age = 0;

Employeestatus = 1 / true;

favoritesports = "Swimming";

}

person::person(const person& object)

{

Name = object.Name;

Age = new int;

\*Age = \*(object.Age);

Employeestatus = object.Employeestatus;

favoritesports = object.favoritesports;

};

void person::setnamesport()

{

cout << "Enter Name: ";

cin >> Name;

cout << "Enter Age: ";

cin >> \*Age;

cout << "Enter Favorite sport: ";

cin >> favoritesports;

cout << "Is the person Employed?: ";

cin >> Employeestatus;

}

void person::Display()const

{

cout << "The Name is: " << this->Name << endl;

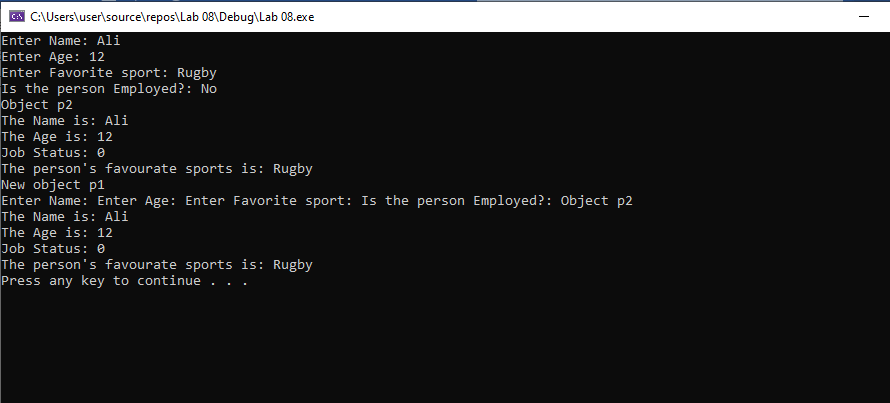
cin.ignore();

cout << "The Age is: " << \*(this->Age) << endl;

cout << "Job Status: " << this->Employeestatus << endl;

cout << "The person's favourate sports is: " << this->favoritesports << endl;

}



**Program # 5:**

#include<iostream>

using namespace std;

int main()

{

cout << "1. Const Variables" << endl;

cout << "2. Constant Function Parameters" << endl;

cout << "3. Constant Return type" << endl;

cout << "4. Const Pointer" << endl;

cout << "5. Pointer to Const Variable" << endl;

cout << "6. Constant Data Members of Class" << endl;

cout << "7. Constant Member Function of Class" << endl;

cout << "select option" << endl;

int option;

cin >> option;

if (option >= 0)

{

switch (option)

{

case 1:

{

cout << "If you make any variable as constant, using const keyword with variable name, you cannot change its value " << endl;

cout << "eg: const datatype a; a is the Constant Variable " << endl;

break;

}

case 2:

{

cout << "A constant parameter is declared when it is necessary that the value of the transferred object remains constnt in the function" << endl;

cout << "eg = void SetValue(const datatype b) " << endl;

break;

}

case 3:

{

cout << "If a function returns a class object by value as a const it cannot be assigned to or otherwise modified" << endl;

cout << "eg = const datatype a(const datatype x) {return x \* x; }" << endl;

break;

}

case 4:

{

cout << "If we make a pointer const, we cannot change the pointer. This means that the pointer will always point tothe same address but we can change the value of that address." << endl;

cout << "eg = int \*const a = & num;" << endl;

break;

}

case 5:

{

cout << "pointer that point constant value the variable which is point by pointer cannot change" << endl;

cout << "const int value = 5; int\* ptr = &value" << endl;

break;

}

case 6:

{

cout << " const data members are not assigned values during its declaration. Const data members are assigned values in the constructor." << endl;

break;

}

case 7:

{

cout << "A member function will not modify the object and it cannot call any non-const member functions" << endl;

break;

}

default:

{

cout << "Invalid Entry!" << endl;

}

}

}

system("pause");

return 0;

}

