OOPS

Ashhar Hasan

13BCS0015

OOPS LAB FILE

Table of Contents

[Count the number of vowels in a character array using pointer arithmetic 2](#_Toc451883064)

[Print a number in reverse order 4](#_Toc451883065)

[Use inline functions to implement addition, subtraction, multiplication, division and modulus 5](#_Toc451883066)

[Use static data member to keep a count of the number of objects of a class 8](#_Toc451883067)

[Use the ‘this’ pointer to print the caller of a function 10](#_Toc451883068)

[Create a base class polygon having a virtual function ‘area’ and override it in its derived classes 12](#_Toc451883069)

[Make a class bank and keep track on number of people in the bank 14](#_Toc451883070)

[Overload the ‘+’ and ‘-‘ operators for a class Company 16](#_Toc451883071)

[Overload the ‘+’ and ‘=’ operator for matrix addition 19](#_Toc451883072)

[Use a friend function to find the average of five numbers, two from class A, three from class B 23](#_Toc451883073)

[Find the average marks of a student from a Student class using a friend class 25](#_Toc451883074)

[Write a template function that finds minimum of two values of a basic data type 27](#_Toc451883075)

[Write a template class Max that finds maximum of two value of a basic data type initialized using a constructor that takes template type as argument 29](#_Toc451883076)

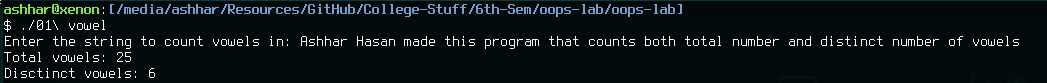
[Overload the stream operators to input and output details of a Student class 31](#_Toc451883077)

[Create a class 3D that inherits from a class 2D. Both should have a function to calculate distance between two points 33](#_Toc451883078)

[Create a calculator in Java which also implements M+ so that result of current operation may be saved into it 35](#_Toc451883079)

[Create a phonebook manager using file handling, with addition, removal and search of contacts 37](#_Toc451883080)

# Count the number of vowels in a character array using pointer arithmetic



#include <iostream>

int countVowels(char\* str)

{

char vowels[] = { 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' };

int numVowels = 10, count = 0;

for (int i = 0; \*(str + i) != '\0'; i++)

{

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

{

if (\*(str + i) == vowels[j])

count++;

}

}

return count;

}

int countDistinctVowels(char\* str)

{

char vowels[] = { 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U' };

int numVowels = 10, count = 0;

for (int i = 0; \*(str + i) != '\0'; i++)

{

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

{

if (\*(str + i) == vowels[j])

{

count++;

// Replace the visited vowels with '\0' to ensure they don't match again

vowels[j] = vowels[j + 5] = '\0';

}

}

}

return count;

}

int main()

{

char str[100];

std::cout << "Enter the string to count vowels in: ";

std::cin.getline(str, 100);

std::cout << "Total vowels: " << countVowels(str) << std::endl;

std::cout << "Disctinct vowels: " << countDistinctVowels(str) << std::endl;

return 0;

}

# Print a number in reverse order



#include <iostream>

int reverse(int num)

{

int n = 0;

while (num)

{

int digit = num % 10;

num /= 10;

n = (n \* 10) + digit;

}

return n;

}

int main()

{

int num;

std::cout << "Enter a number: ";

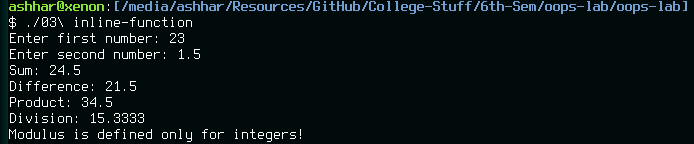
std::cin >> num;

std::cout << reverse(num) << std::endl;

return 0;

}

# Use inline functions to implement addition, subtraction, multiplication, division and modulus



#include <iostream>

class Math

{

public:

double getA()

{ return a; }

void setA(double);

double getB()

{ return b; }

void setB(double);

double add()

{ return (a + b); }

double subtract()

{ return (a - b); }

double multiply()

{ return (a \* b); }

double divide();

double modulus();

private:

double a, b;

};

void Math::setA(double num)

{

// Store as an integer if an integer was input

if (static\_cast<int>(num) == num)

a = static\_cast<int>(num);

else

a = num;

}

void Math::setB(double num)

{

// Store as an integer if an integer was input

if (static\_cast<int>(num) == num)

b = static\_cast<int>(num);

else

b = num;

}

double Math::divide()

{

if (b == 0)

{

std::cerr << "Division by 0 not allowed!" << std::endl;

// a / b equals 0 only when a = 0. We use this to detect error.

return 0;

}

return (a / b);

}

double Math::modulus()

{

// If the value casted to an integer is the same as the original value, the value is an integer.

if (static\_cast<int>(a) == a && static\_cast<int>(b) == b && static\_cast<int>(b) != 0)

return (static\_cast<int>(a) % static\_cast<int>(b));

std::cerr << "Modulus is defined only for integers!" << std::endl;

// Since a % b never equals b, returning b indicated error.

return b;

}

int main()

{

Math maths;

double num;

std::cout << "Enter first number: ";

std::cin >> num;

maths.setA(num);

std::cout << "Enter second number: ";

std::cin >> num;

maths.setB(num);

std::cout << "Sum: " << maths.add() << std::endl;

std::cout << "Difference: " << maths.subtract() << std::endl;

std::cout << "Product: " << maths.multiply() << std::endl;

if (maths.divide() != 0 && maths.getA() != 0)

std::cout << "Division: " << maths.divide() << std::endl;

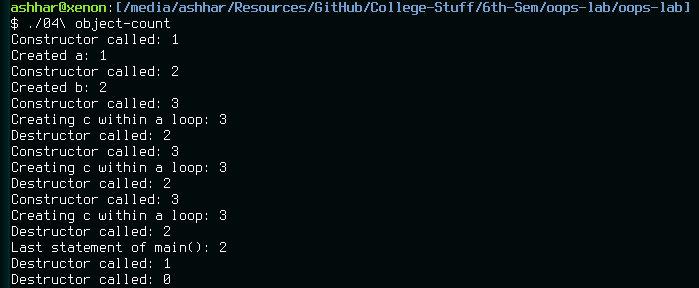
if (maths.modulus() != maths.getB())

std::cout << "Modulus: " << maths.modulus() << std::endl;

return 0;

}

# Use static data member to keep a count of the number of objects of a class



#include <iostream>

class SelfCount

{

public:

SelfCount()

{

count++;

std::cout << "Constructor called: " << count << std::endl;

}

~SelfCount()

{

count--;

std::cout << "Destructor called: " << count << std::endl;

}

static int count;

};

int SelfCount::count = 0;

int main()

{

SelfCount a;

std::cout << "Created a: " << a.count << std::endl;

SelfCount b;

std::cout << "Created b: " << b.count << std::endl;

for (int i = 0; i < 3; i++)

{

SelfCount c;

std::cout << "Creating c within a loop: " << c.count << std::endl;

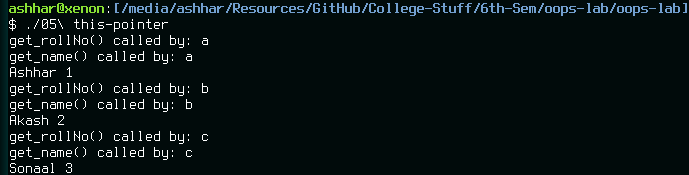
}

std::cout << "Last statement of main(): " << a.count << std::endl;

return 0;

}

# Use the ‘this’ pointer to print the caller of a function



#include <iostream>

#include <string>

#include <vector>

class Student

{

public:

Student(char \_id, std::string \_name, int \_rollNo) : objName(\_id),

name(\_name),

rollNo(\_rollNo) {}

int getRollNo();

void setRollNo(int roll)

{ rollNo = roll; }

std::string getName();

void setName(std::string str)

{ name.assign(str); }

char getObjName();

private:

char objName;

std::string name;

int rollNo;

};

int Student::getRollNo()

{

std::cerr << std::endl << "get\_rollNo() called by: " << this->objName << std::endl;

return rollNo;

}

std::string Student::getName()

{

std::cerr << std::endl << "get\_name() called by: " << this->objName << std::endl;

return name;

}

char Student::getObjName()

{

std::cerr << std::endl << "get\_objName() called by: " << this->objName << std::endl;

return objName;

}

int main()

{

char choice = 'y';

std::vector<Student> students;

while (tolower(choice) == 'y')

{

char \_id;

std::cout << "Enter a single character id for the object: ";

std::cin >> \_id;

std::string \_name;

std::cout << "Enter the name of the student: ";

// Ignore leftover characters

std::cin.ignore(1, '\n');

std::getline(std::cin, \_name);

int \_rollNo;

std::cout << "Enter the roll number of the student: ";

std::cin >> \_rollNo;

students.push\_back(Student(\_id, \_name, \_rollNo));

std::cout << "Do you want to create more objects? (y / n): ";

std::cin >> choice;

}

std::cout << std::endl;

for (unsigned int i = 0; i < students.size(); i++)

{

std::cout << "================" << std::endl << "Object " << i + 1 << std::endl << "================";

std::cout << "ID: " << students[i].getObjName() << std::endl;

std::cout << "Name: " << students[i].getName() << std::endl;

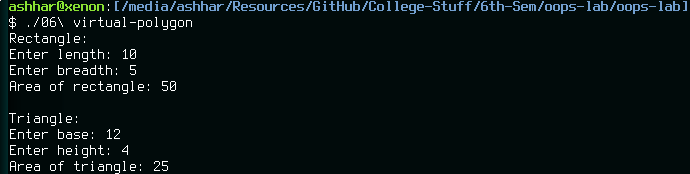
std::cout << "Roll No: " << students[i].getRollNo() << std::endl << std::endl;

}

return 0;

}

# Create a base class polygon having a virtual function ‘area’ and override it in its derived classes



#include <iostream>

class c\_polygon

{

public:

virtual double area()

{ return 0.0; }

};

class c\_rectangle : public c\_polygon

{

public:

double area()

{ return (length \* breadth); }

c\_rectangle(double l, double b) : length(l),

breadth(b) {}

private:

double length, breadth;

};

class c\_triangle : public c\_polygon

{

public:

double area()

{ return (0.5 \* base \* height); }

c\_triangle(double b, double h) : base(b),

height(h) {}

private:

double base, height;

};

int main()

{

double rl, rb, tb, th;

std::cout << "Rectangle" << std::endl;

std::cout << "Enter length: ";

std::cin >> rl;

std::cout << "Enter breadth: ";

std::cin >> rb;

c\_rectangle rectangle(rl, rb);

std::cout << "Area of rectangle: " << rectangle.area() << std::endl << std::endl;

std::cout << "Triangle" << std::endl;

std::cout << "Enter base: ";

std::cin >> tb;

std::cout << "Enter height: ";

std::cin >> th;

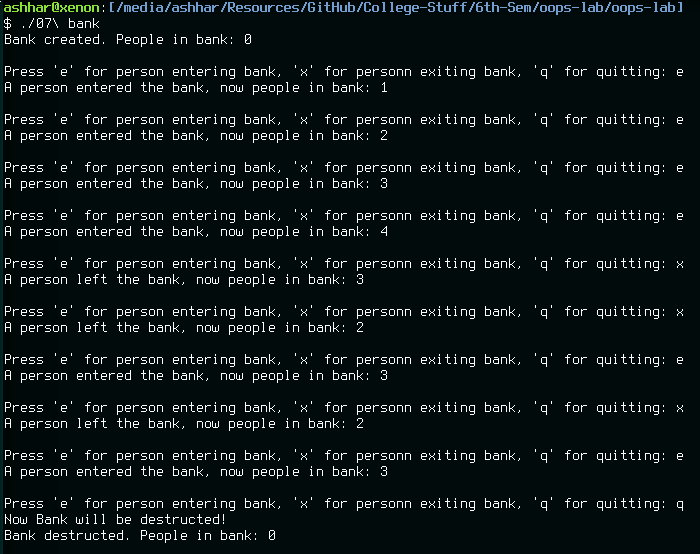
c\_triangle triangle(rl, rb);

std::cout << "Area of triangle: " << triangle.area() << std::endl;

return 0;

}

# Make a class bank and keep track on number of people in the bank



#include <iostream>

#include <cstdio>

class Bank

{

public:

Bank()

{

peopleInBank = 0;

std::cout << "Bank created. People in bank: " << peopleInBank << std::endl << std::endl;

}

~Bank()

{

peopleInBank = 0;

std::cout << "Bank destructed. People in bank: " << peopleInBank << std::endl << std::endl;

}

void enter()

{

peopleInBank++;

std::cout << "A person entered the bank, now people in bank: " << peopleInBank << std::endl << std::endl;

}

void exit()

{

peopleInBank--;

std::cout << "A person left the bank, now people in bank: " << peopleInBank << std::endl << std::endl;

}

private:

static int peopleInBank;

};

int Bank::peopleInBank = 0;

int main()

{

Bank bank;

char choice = 'e';

while (true)

{

std::cout << "Press 'e' for person entering bank, 'x' for personn exiting bank, 'q' for quitting: ";

std::cin >> choice;

if (choice == 'e')

bank.enter();

else if (choice == 'x')

bank.exit();

else if (choice == 'q')

break;

else

std::cout << "You entered an invalid operation. Try again.";

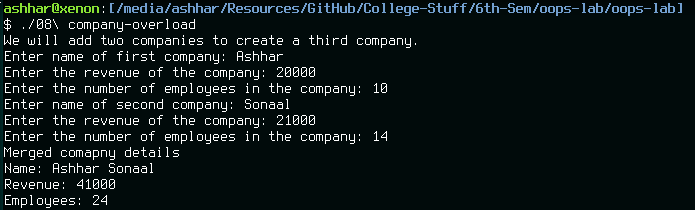
}

std::cout << "Now Bank will be destructed!" << std::endl;

return 0;

}

# Overload the ‘+’ and ‘-‘ operators for a class Company



#include <iostream>

#include <string>

class Company

{

public:

Company operator+(Company);

Company operator-(Company);

void setName(std::string \_name)

{ name = \_name; }

void setRevenue(int \_revenue)

{ revenue = \_revenue; }

void setEmployees(int \_employees)

{ employees = \_employees; }

std::string getName()

{ return name; }

int getRevenue()

{ return revenue; }

int getEmployees()

{ return employees; }

private:

std::string name;

int revenue;

int employees;

};

Company Company::operator+(Company a)

{

Company ret;

ret.name.assign(this->name);

ret.name += " ";

ret.name += a.name;

ret.revenue = this->revenue + a.revenue;

ret.employees = this->employees + a.employees;

return ret;

}

Company Company::operator-(Company)

{

Company ret;

ret.name.assign(this->name);

ret.revenue = this->revenue;

ret.employees = this->employees;

return ret;

}

int main()

{

Company a, b;

std::cout << "We will add two companies to create a third company." << std::endl;

std::string \_name;

int \_revenue, \_employees;

std::cout << "Enter name of first company: ";

getline(std::cin, \_name);

std::cout << "Enter the revenue of the company: ";

std::cin >> \_revenue;

std::cout << "Enter the number of employees in the company: ";

std::cin >> \_employees;

a.setName(\_name);

a.setRevenue(\_revenue);

a.setEmployees(\_employees);

std::cout << "Enter name of second company: ";

// Ignore leftover characters

std::cin.ignore(1, '\n');

getline(std::cin, \_name);

std::cout << "Enter the revenue of the company: ";

std::cin >> \_revenue;

std::cout << "Enter the number of employees in the company: ";

std::cin >> \_employees;

b.setName(\_name);

b.setRevenue(\_revenue);

b.setEmployees(\_employees);

Company c;

c = a + b;

std::cout << "Merged comapny details" << std::endl;

std::cout << "Name: " << c.getName() << std::endl;

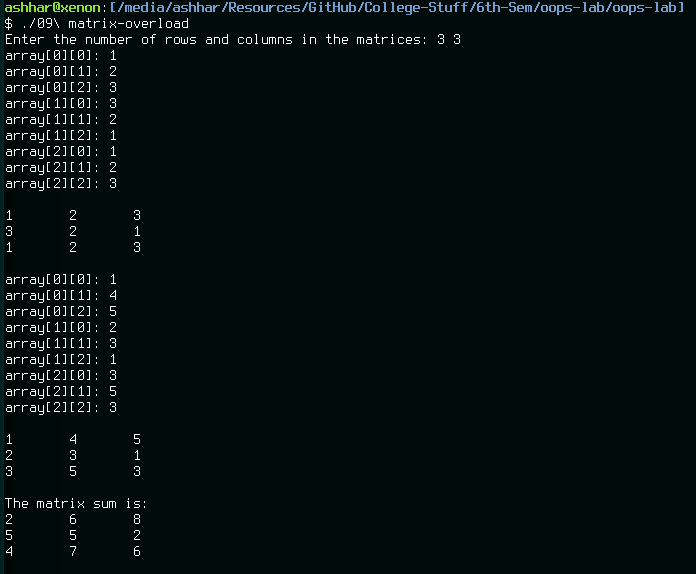
std::cout << "Revenue: " << c.getRevenue() << std::endl;

std::cout << "Employees: " << c.getEmployees() << std::endl;

return 0;

}

# Overload the ‘+’ and ‘=’ operator for matrix addition



#include <iostream>

#include <cstdlib>

class Matrix

{

public:

Matrix(int, int);

Matrix(const Matrix&);

~Matrix();

Matrix operator=(const Matrix&);

Matrix operator+(const Matrix&);

void print();

void getMatrix();

private:

int row;

int col;

int \*\*array;

void allocate();

void release();

};

Matrix::Matrix(int r, int c)

{

row = r;

col = c;

allocate();

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

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

array[i][j] = rand() % 10;

}

Matrix::Matrix(const Matrix& m)

{

row = m.row;

col = m.col;

allocate();

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

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

array[i][j] = m.array[i][j];

}

Matrix::~Matrix()

{ release(); }

void Matrix::allocate()

{

array = new int\*[static\_cast<unsigned long>(row)];

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

array[i] = new int[static\_cast<unsigned long>(col)];

}

void Matrix::release()

{

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

delete [] array[i];

delete [] array;

}

void Matrix::print()

{

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

{

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

std::cout << array[i][j] << "\t";

std::cout << std::endl;

}

}

void Matrix::getMatrix()

{

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

{

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

{

std::cout << "array[" << i << "][" << j << "]: ";

std::cin >> array[i][j];

}

}

}

Matrix Matrix::operator+(const Matrix &m)

{

if (row != m.row || col != m.col)

{

std::cerr << "Invalid Matrix addition!";

return \*this;

}

Matrix temp(row,col);

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

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

temp.array[i][j] = array[i][j] + m.array[i][j];

return temp;

}

Matrix Matrix::operator=(const Matrix &m)

{

for (int i = 0; i < m.row; i++)

for (int j = 0; j < m.col; j++)

array[i][j] = m.array[i][j];

return \*this;

}

int main()

{

int row, col;

std::cout << "Enter the number of rows and columns in the matrices: ";

std::cin >> row >> col;

Matrix a(row, col), b(row, col), c(row, col);

a.getMatrix();

std::cout << std::endl;

a.print();

std::cout << std::endl;

b.getMatrix();

std::cout << std::endl;

b.print();

std::cout << std::endl;

c = a + b;

std::cout << "The matrix sum is:" << std::endl;

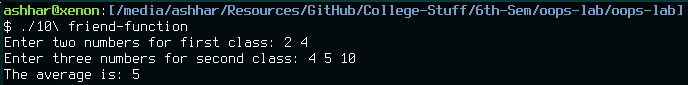
c.print();

std::cout << std::endl;

return 0;

}

# Use a friend function to find the average of five numbers, two from class A, three from class B



#include <iostream>

// Forward declaration for friend function in TwoNumbers

class ThreeNumbers;

class TwoNumbers

{

public:

TwoNumbers(int \_a, int \_b) : a(\_a),

b(\_b) {}

TwoNumbers(const TwoNumbers& t) : a(t.a),

b(t.b) {}

TwoNumbers operator=(const TwoNumbers&);

friend double computeAverage(TwoNumbers, ThreeNumbers);

private:

int a, b;

};

TwoNumbers TwoNumbers::operator=(const TwoNumbers& t)

{

a = t.a;

b = t.b;

return \*this;

}

class ThreeNumbers

{

public:

ThreeNumbers(int \_a, int \_b, int \_c) : a(\_a),

b(\_b),

c(\_c) {}

ThreeNumbers(const ThreeNumbers& t) : a(t.a),

b(t.b),

c(t.c) {}

ThreeNumbers operator=(const ThreeNumbers&);

friend double computeAverage(TwoNumbers, ThreeNumbers);

private:

int a, b, c;

};

ThreeNumbers ThreeNumbers::operator=(const ThreeNumbers& t)

{

a = t.a;

b = t.b;

c = t.c;

return \*this;

}

double computeAverage(TwoNumbers tw, ThreeNumbers th)

{

return ((tw.a + tw.b + th.a + th.b + th.c) / 5);

}

int main()

{

int a, b, c;

std::cout << "Enter two numbers for first class: ";

std::cin >> a >> b;

TwoNumbers tw(a, b);

std::cout << "Enter three numbers for second class: ";

std::cin >> a >> b >> c;

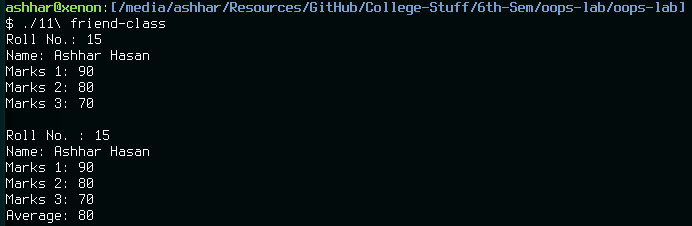
ThreeNumbers th(a, b, c);

std::cout << "The average is: " << computeAverage(tw, th) << std::endl;

return 0;

}

# Find the average marks of a student from a Student class using a friend class



#include <iostream>

#include <string>

class Student

{

public:

void getData();

void displayData();

friend class Average;

private:

int roll;

std::string name;

double m1, m2, m3;

};

class Average

{

public:

double computeAverage(Student& s)

{ return (s.m1 + s.m2 + s.m3) / 3; }

};

void Student::getData()

{

std::cout << "Roll No.: ";

std::cin >> roll;

std::cout << "Name: ";

// Ignore leftover characters

std::cin.ignore(1, '\n');

std::getline(std::cin, name);

std::cout << "Marks 1: ";

std::cin >> m1;

std::cout << "Marks 2: ";

std::cin >> m2;

std::cout << "Marks 3: ";

std::cin >> m3;

}

void Student::displayData()

{

Average a;

std::cout << std::endl << "Roll No. : " << roll << std::endl;

std::cout << "Name: " << name << std::endl;

std::cout << "Marks 1: " << m1 << std::endl;

std::cout << "Marks 2: " << m2 << std::endl;

std::cout << "Marks 3: " << m3 << std::endl;

std::cout << "Average: " << a.computeAverage(\*this) << std::endl;

}

int main()

{

Student s;

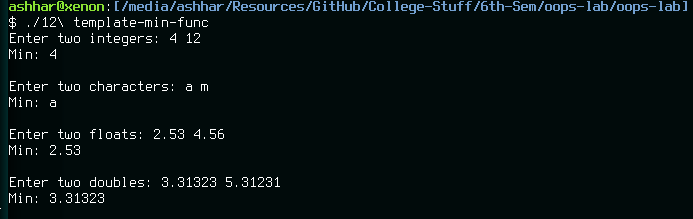
s.getData();

s.displayData();

return 0;

}

# Write a template function that finds minimum of two values of a basic data type



#include <iostream>

template <class T>

T min(T a, T b)

{

if (a < b)

return a;

else if (a > b)

return b;

else

return (a - b);

}

int main()

{

int ai, bi;

char ac, bc;

float af, bf;

double ad, bd;

std::cout << "Enter two integers: ";

std::cin >> ai >> bi;

std::cout << "Min: " << min(ai, bi) << std::endl << std::endl;

std::cout << "Enter two characters: ";

std::cin >> ac >> bc;

std::cout << "Min: " << min(ac, bc) << std::endl << std::endl;

std::cout << "Enter two floats: ";

std::cin >> af >> bf;

std::cout << "Min: " << min(af, bf) << std::endl << std::endl;

std::cout << "Enter two doubles: ";

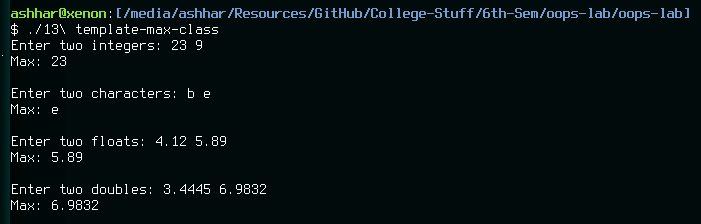
std::cin >> ad >> bd;

std::cout << "Min: " << min(ad, bd) << std::endl << std::endl;

return 0;

}

# Write a template class Max that finds maximum of two value of a basic data type initialized using a constructor that takes template type as argument



#include <iostream>

template <class T>

class Max

{

public:

Max(T \_a, T \_b) : a(\_a),

b(\_b) {}

T max()

{

if (a > b)

return a;

else if (a < b)

return b;

else

return (a - b);

}

private:

T a, b;

};

int main()

{

int ai, bi;

char ac, bc;

float af, bf;

double ad, bd;

std::cout << "Enter two integers: ";

std::cin >> ai >> bi;

Max<int> i(ai, bi);

std::cout << "Max: " << i.max() << std::endl << std::endl;

std::cout << "Enter two characters: ";

std::cin >> ac >> bc;

Max<char> c(ac, bc);

std::cout << "Max: " << c.max() << std::endl << std::endl;

std::cout << "Enter two floats: ";

std::cin >> af >> bf;

Max<float> f(af, bf);

std::cout << "Max: " << f.max() << std::endl << std::endl;

std::cout << "Enter two doubles: ";

std::cin >> ad >> bd;

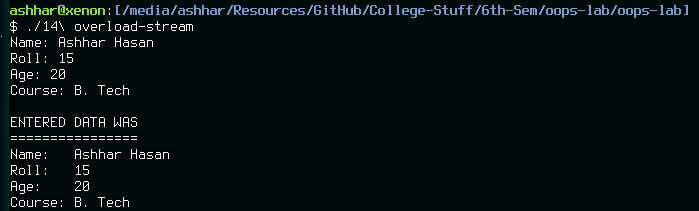
Max<double> d(ad, bd);

std::cout << "Max: " << d.max() << std::endl << std::endl;

return 0;

}

# Overload the stream operators to input and output details of a Student class



#include <iostream>

class Student

{

public:

Student(std::string \_name = "", int \_roll = 0, int \_age = 0, std::string \_course = "") : name(\_name),

roll(\_roll),

age(\_age),

course(\_course) {}

friend std::ostream& operator<<(std::ostream&, Student&);

friend std::istream& operator>>(std::istream&, Student&);

private:

std::string name;

int roll;

int age;

std::string course;

};

std::ostream& operator<<(std::ostream& os, Student& s)

{

os << "Name: " << s.name << std::endl;

os << "Roll: " << s.roll << std::endl;

os << "Age: " << s.age << std::endl;

os << "Course: " << s.course << std::endl;

return os;

}

std::istream& operator>>(std::istream& is, Student& s)

{

std::cout << "Name: ";

std::getline(is, s.name);

std::cout << "Roll: ";

is >> s.roll;

std::cout << "Age: ";

is >> s.age;

std::cout << "Course: ";

// Ignore leftover characters

std::cin.ignore(1, '\n');

std::getline(is, s.course);

return is;

}

int main()

{

Student s;

std::cin >> s;

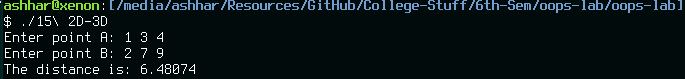
std::cout << std::endl << "ENTERED DATA WAS" << std::endl << "================" << std::endl;

std::cout << s;

return 0;

}

# Create a class 3D that inherits from a class 2D. Both should have a function to calculate distance between two points



#include <iostream>

#include <cmath>

class TwoD

{

public:

double distance(TwoD A)

{ return (sqrt(pow((A.x - this->x), 2) + pow((A.y - this->y), 2))); }

TwoD() : x(0),

y(0) {}

TwoD(int \_x, int \_y) : x(\_x),

y(\_y) {}

private:

int x, y;

};

class ThreeD : public TwoD

{

public:

double distance(ThreeD A)

{ return (sqrt(pow((A.x - this->x), 2) + pow((A.y - this->y), 2) + pow((A.z - this->z), 2))); }

ThreeD(int \_x, int \_y, int \_z) : TwoD(),

x(\_x),

y(\_y),

z(\_z) {}

private:

int x, y, z;

};

int main()

{

int x, y, z;

std::cout << "Enter point A: ";

std::cin >> x >> y >> z;

ThreeD A(x, y, z);

std::cout << "Enter point B: ";

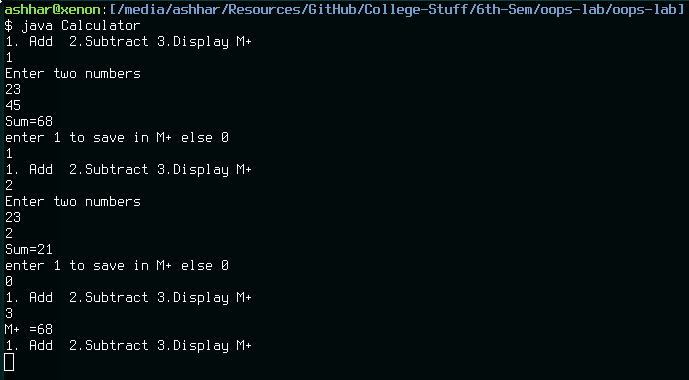
std::cin >> x >> y >> z;

ThreeD B(x, y, z);

std::cout << "The distance is: " << B.distance(A) << std::endl;

}

# Create a calculator in Java which also implements M+ so that result of current operation may be saved into it



import java.util.Scanner;

public class Calculator {

static int add(int a, int b) { return (a + b); }

static int subtract(int a, int b) { return (a - b); }

public static void main(String[] args) {

int m = 0, result;

Scanner sc = new Scanner(System.in);

int choice = 0;

while(choice <= 5)

{

System.out.println("1. Add\n2.Subtract\n3.Display M+\n4. Add M+\n5. Subtract M+\nAny other number to exit\n");

choice = sc.nextInt();

switch(choice) {

case 1:

System.out.println("Enter two numbers: ");

result = add(sc.nextInt(), sc.nextInt());

System.out.println("Result: " + result);

System.out.println("Enter 1 to save in M+, 0 to continue");

if (sc.nextInt() == 1)

m = result;

break;

case 2:

System.out.println("Enter two numbers: ");

result = subtract(sc.nextInt(), sc.nextInt());

System.out.println("Result: " + result);

System.out.println("Enter 1 to save in M+, 0 to continue");

if (sc.nextInt() == 1)

m = result;

break;

case 3:

System.out.println("M+: " + m);

break;

case 4:

System.out.println("Enter a number: ");

result = add(sc.nextInt(), m);

System.out.println("Result: " + result);

m = result;

break;

case 5:

System.out.println("Enter a number: ");

result = subtract(sc.nextInt(), m);

System.out.println("Result: " + result);

m = result;

break;

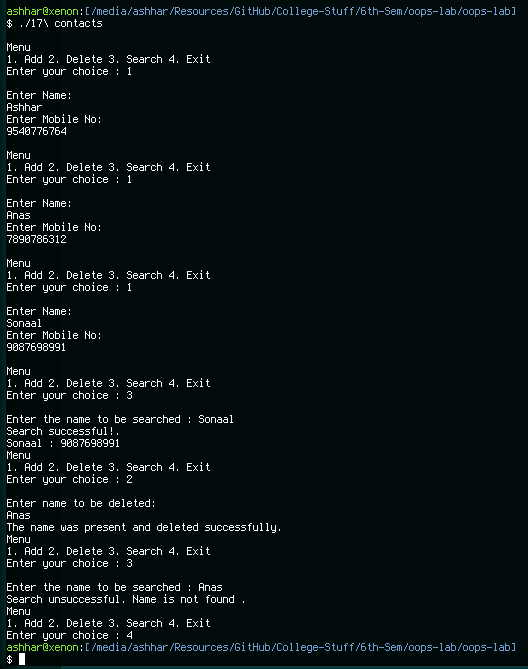
}

}

}

}

# Create a phonebook manager using file handling, with addition, removal and search of contacts



#include <fstream>

#include <iostream>

#include <cstdlib>

bool addContact()

{

std::string name;

std::string mobileNo;

std::cout << "Enter Name: ";

std::getline(std::cin, name);

std::cout << "Enter Mobile No: ";

// Ignore leftover characters

std::cin.ignore(1, '\n');

std::getline(std::cin, mobileNo);

std::ofstream contactFile("contacts.txt", std::ios::app | std::ios::out );

contactFile << name << "\n" << mobileNo << std::endl ;

return true;

}

bool deleteContact()

{

std::string name, line;

std::cout << "Enter name to be deleted: ";

std::getline(std::cin, name);

std::ifstream contactFile;

std::ofstream newFile;

contactFile.open("contacts.txt");

newFile.open("newFile.txt");

while (getline(contactFile, line))

{

if (line != name)

newFile << line << std::endl;

else

{

std::cout << "The name was present and deleted successfully.";

std::getline(contactFile, line);

std::getline(contactFile, line);

}

}

std::remove("contacts.txt");

std::rename("newFile.txt", "contacts.txt");

return true;

}

bool searchContact()

{

std::string name, line;

std::cout << "Enter the name to be searched : ";

std::getline(std::cin, name);

std::ifstream contactFile("contacts.txt");

while (std::getline(contactFile, line))

{

if (line == name)

{

std::cout << "Search successful!" << std::endl;

std::cout << line << " : ";

getline(contactFile, line);

std::cout << line;

return true;

}

}

std::cout << "Search unsuccessful. Contact not found.";

return false;

}

int main()

{

int choice;

while (true)

{

std::cout << "\nMenu" << std::endl;

std::cout << "1. Add\n2. Delete\n3. Search\n4. Exit" << std::endl;

std::cout << "Enter your choice: ";

std::cin >> choice;

switch(choice)

{

case 1:

addContact();

break;

case 2:

deleteContact();

break;

case 3:

searchContact();

break;

case 4:

exit(1);

default:

std::cout << "Invalid choice. Try Again";

break;

}

}

return 0;

}