//1.1 ascii

#include<iostream>

int main() {

char letter;

std::cout << "Insert a letter." << std::endl;

std::cin >> letter; //read the letter

//show the letter converted to ascii code

std::cout << "The ASCII code for the letter "

<< letter

<< " is "

<< (int)letter

<< std::endl;

system("pause");

return 0;

}

//1.2

#include<iostream>

int main() {

int a, b, c;

double media, a\_media, b\_media, c\_media;

std::cout << "A? ";

std::cin >> a; //read the value of a first number A

std::cout << "B? ";

std::cin >> b; //read the value of a second number B

std::cout << "C? ";

std::cin >> c; //read the value of a third number C

// calculate the average and the difference between it and each number

media = (a + b + c) / 3.0;

a\_media = a - media;

b\_media = b - media;

c\_media = c - media;

//show the results

std::cout << "media = " << media

<< std::endl << "A\_media = " << a\_media

<< std::endl << "B\_media = " << b\_media

<< std::endl << "C\_media = " << c\_media

<< std::endl;

system("pause");

return 0;

}

//1.3

#include<iostream>

#include<cmath>

int main() {

double m, p, r, PI = acos(-1); //define the value of pi

std::cout << "What is the specific weight of the material in kg/m^3 ? ";

std::cin >> p; // read the value of p has the weight of the material

std::cout << "What is the radius of the sphere in meters? ";

std::cin >> r; // read the value of r as the radius of the sphere

m = (4 / 3) \* p \* PI \* pow(r, 3); //calculate the weight of the sphere

std::cout << "The weight of the sphere is " << m << " kg." << std::endl; //show the result

system("pause");

return 0;

}

//1.4

#include<iostream>

int main()

{

float a, b, c, d, e, f, x, y;

std::cout << "If you have two equations: a\*x + b\*y = c and d\*x + b\*y =c, and you mean to figure the values of x and y, insert the values of the constants."

<< std::endl;

std::cout << "a ? ";

std::cin >> a;

std::cout << "b ? ";

std::cin >> b;

std::cout << "c ? ";

std::cin >> c;

std::cout << "d ? ";

std::cin >> d;

std::cout << "e ? ";

std::cin >> e;

std::cout << "f ? ";

std::cin >> f;

x = (c \* e - b \* f) / (a \* e - b \* d); //calculate variable x

y = (a \* f - c \* d) / (a \* e - b \* d); //calculate variable y

//show the values of x and y

std::cout << "x = " << x << std::endl

<< "y = " << y << std::endl;

system("pause");

return 0;

}

//1.5

//1.6

#include<iostream>

#include<cmath>

int main()

{

float x1, y1, x2, y2, x3, y3, area, a, b, c, s;

std::cout << "Insert the coordinates of the first vertex (x y): ";

std::cin >> x1 >> y1;

std::cout << "Insert the coordinates of the second vertex (x y): ";

std::cin >> x2 >> y2;

std::cout << "Insert the coordinates of the third vertex (x y): ";

std::cin >> x3 >> y3;

//calculate the length of each side of the triangule

a = sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));

b = sqrt(pow(x3 - x1, 2) + pow(y3 - y1, 2));

c = sqrt(pow(x3 - x2, 2) + pow(y3 - y2, 2));

s = (a + b + c) / 2; //calculate the semi-perimeter

//calculate the area

area = sqrt(s\*(s - a)\*(s - b)\*(s - c));

//show results

std::cout << "Area = " << area << std::endl;

system("pause");

return 0;

}

//2.1

#include<iostream>

int main()

{

float a, b, c, d, e, f, x, y;

std::cout << "If you have two equations: a\*x + b\*y = c and d\*x + b\*y =c, and you mean to figure the values of x and y, insert the values of the constants."

<< std::endl;

std::cout << "a ? ";

std::cin >> a;

std::cout << "b ? ";

std::cin >> b;

std::cout << "c ? ";

std::cin >> c;

std::cout << "d ? ";

std::cin >> d;

std::cout << "e ? ";

std::cin >> e;

std::cout << "f ? ";

std::cin >> f;

if (a \* e - b \* d == 0 && (c \* e - b \* f == 0 || a \* f - c \* d == 0))

{

std::cout << "sistema indeterminado" << std::endl;

}

else if (a \* e - b \* d == 0)

{

std::cout << "sistema impossivel" << std::endl;

}

else

{

x = (c \* e - b \* f) / (a \* e - b \* d); //calculate variable x

y = (a \* f - c \* d) / (a \* e - b \* d); //calculate variable y

//show the values of x and y

std::cout << "x = " << x << std::endl

<< "y = " << y << std::endl;

}

system("pause");

return 0;

}

//2.2 a

#include<iostream>

int main()

{

int num1, num2, num3, major, minor;

std::cin >> num1 >> num2 >> num3;

//let's assume that the major is num1

major = num1;

//find the major

if (major >= num2 && major >= num3) //num1 is really the major

{

if (num2 <= num3) //let's find out what is the minor in this case

{

minor = num2;

}

else { minor = num3; }

}

else if (num2 >= num1 && num2 >= num3)

{

major = num2; //num2 is the real major

if (num1 <= num3) //let's find out what is the minor in this case

{

minor = num1;

}

else { minor = num3; }

}

else

{

major = num3; //num3 is the major

if (num1 <= num2) //let's find out what is the minor in this case

{

minor = num1;

}

else { minor = num2; }

}

//show results

std::cout << "Maior: " << major << std::endl

<< "Menor: " << minor << std::endl;

system("pause");

return 0;

}

//2.2 b

#include<iostream>

int main()

{

int num1, num2, num3, major, minor, intermediate;

std::cin >> num1 >> num2 >> num3;

//let's assume that the major is num1

major = num1;

//find the major

if (major >= num2 && major >= num3) //num1 is really the major

{

if (num2 <= num3) //let's find out what is the minor in this case

{

minor = num2;

}

else { minor = num3; }

}

else if (num2 >= num1 && num2 >= num3)

{

major = num2; //num2 is the real major

if (num1 <= num3) //let's find out what is the minor in this case

{

minor = num1;

}

else { minor = num3; }

}

else

{

major = num3; //num3 is the major

if (num1 <= num2) //let's find out what is the minor in this case

{

minor = num1;

}

else { minor = num2; }

}

intermediate = (num1 + num2 + num3) - (major + minor);

//show results

std::cout << major << " " << intermediate << " " << minor << std::endl;

system("pause");

return 0;

}

//2.2 c

#include<iostream>

int main()

{

double num1, num2, num3, major, sum\_minors;

std::cin >> num1 >> num2 >> num3;

//find the major side

if (num1 >= num2 && num1 >= num3) //in case the biggest is num1

{

major = num1;

sum\_minors = num2 + num3;

}

else if (num2 >= num1 && num2 >= num3) //in case the biggest is num2

{

major = num2;

sum\_minors = num1 + num3;

}

else

{

major = num3; //in case the biggest is num3

sum\_minors = num1 + num2;

}

if (sum\_minors > major)

{

std::cout << "Podem representar as medidas do lados de um triângulo." << std::endl;

}

else std::cout << "Não podem representar as medidas do lados de um triângulo." << std::endl;

system("pause");

return 0;

}

//2.5

#include<iostream>

#include<iomanip>

int main()

{

float a, b, c, raiz1, raiz2;

std::cout << "Introduza os coeficientes (a b c): ";

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

if (pow(b, 2) - 4 \* a\*c > 0) //in case the roots are different and real

{

raiz1 = (-b - sqrt(pow(b, 2) - 4 \* a\*c)) / (2 \* a);

raiz2 = (-b + sqrt(pow(b, 2) - 4 \* a\*c)) / (2 \* a);

std::cout << "A equação tem 2 raízes reais diferentes: " << std::fixed << std::setprecision(3) << raiz1 << " e " << raiz2 << std::endl;

}

else if (pow(b, 2) - 4 \* a\*c == 0) //in case the roots are the same

{

raiz1 = -b / (2 \* a);

std::cout << "A equação tem 2 raízes reais iguais: " << std::fixed << std::setprecision(3) << raiz1 << std::endl;

}

else //in case the roots are complex numbers

{

std::cout << "A equação tem 2 raízes complexas conjugadas: " << std::fixed << std::setprecision(3)

<< -b / (2 \* a) << "+" << (sqrt(abs(pow(b, 2) - 4 \* a\*c))) / (2 \* a) << "i e "

<< -b / (2 \* a) << "-" << (sqrt(abs(pow(b, 2) - 4 \* a\*c))) / (2 \* a) << "i" << std::endl;

}

system("pause");

return 0;

}

//2.6 a

#include<iostream>

#include<cmath>

int main()

{

int number, i = 2;

bool prime = true;

std::cout << "Give me a number. ";

std::cin >> number;

while (i <= sqrt(number)) //we only have to test until the square root of the number

{

if (number % i == 0) //if the number divided by i is

{

prime = false; //if found, it's not a prime

break; //so we can leave the cicle

}

else i++; //if not, we try the next integer

}

//show results

if (prime == false) //if its not a prime

{

std::cout << number << " is not a prime." << std::endl;

}

else //if it is a prime

{

std::cout << number << " is a prime." << std::endl;

}

system("pause");

return 0;

}

//2.6 b

#include<iostream>

#include<cmath>

int is\_it\_prime(int number) //this function tests if a given number is a prime

{

int i = 2;

bool prime = true;

while (i <= sqrt(number)) //we only have to test until the square root of the number

{

if (number % i == 0) //if the number divided by i is

{

prime = false; //if found, it's not a prime

break; //so we can leave the cicle

}

else i++; //if not, we try the next integer

}

return prime;

}

int main()

{

std::cout << "1" << std::endl; //the first prime number

for (int i = 2; i <= 100; i++) //we only have to test until the square root of the number

{

bool testnumber = is\_it\_prime(i); //it calls the function is\_it\_prime, and receives a boolean valor

if (testnumber == 1) //if it is a prime number, it writes it

{

std::cout << i << std::endl;

}

//if not, it proceeds to test the next integer

}

system("pause");

return 0;

}

//2.6 c

#include<iostream>

#include<cmath>

int is\_it\_prime(int number) //this function tests if a given number is a prime

{

int i = 2;

bool prime = true;

while (i <= sqrt(number)) //we only have to test until the square root of the number

{

if (number % i == 0) //if the number divided by i is

{

prime = false; //if found, it's not a prime

break; //so we can leave the cicle

}

else i++; //if not, we try the next integer

}

return prime;

}

int main()

{

std::cout << "1" << std::endl; //the first prime number

for (int i = 2; i <= 10000; i++) //we only have to test until the square root of the number

{

bool testnumber = is\_it\_prime(i); //it calls the function is\_it\_prime, and receives a boolean valor

if (testnumber == 1) //if it is a prime number, it writes it

{

std::cout << i << std::endl;

}

//if not, it proceeds to test the next integer

}

system("pause");

return 0;

}

// 2.7 a

#include <iostream>

#include<cmath>

#include<iomanip>

int main()

{

const float PI = acos(-1);

float maximum, minimum, increment;

std::cout << "Insert the maximum, the minimum and the increment (by this order). ";

std::cin >> minimum >> maximum >> increment;

std::cout << "ang sen cos tan" << std::endl;

for (float i = minimum; i <= maximum;)

{

float angle = (PI \* i) / 180;

if (i == 90)

{

std::cout << std::fixed << std::setprecision(2) << i << " "

<< std::fixed << std::setprecision(6)

<< sin(angle) << " " << cos(angle) << " " << "infinito" << std::endl;

}

else

{

std::cout << std::fixed << std::setprecision(2) << i << " "

<< std::fixed << std::setprecision(6)

<< sin(angle) << " " << cos(angle) << " " << tan(angle) << std::endl;;

}

i = i + increment;

}

system("pause");

return 0;

}

//2.9 a

#include<iostream>

int main()

{

int minimum = 0, maximum = 0, length = 0, value, sum = 0;

float mean;

while (1)

{

std::cin >> value;

if (value == 0) { break; }

if (length == 0)

{

minimum = value;

maximum = value;

}

if (value > maximum) { maximum = value; }

if (value < minimum) { minimum = value; }

sum = sum + value;

length = length + 1;

}

mean = sum / length;

std::cout << "Minímo: " << minimum << std::endl

<< "Máximo: " << maximum << std::endl

<< "Soma: " << sum << std::endl

<< "Média: " << mean << std::endl;

system("pause");

return 0;

}

//2.9 c

#include<iostream>

int main()

{

int minimum, maximum, length = 0, value, sum = 0;

float mean;

while (!std::cin.eof()) //only stops when you insert the control for end of file

{

std::cin >> value;

if (length == 0) //if it is the first value to be introduced we make it the maximum and the minimum until a bigger or smallest number is insert

{

maximum = value;

minimum = value;

}

if (value > maximum) { maximum = value; } //a new maximum has been found

if (value < minimum) { minimum = value; } //a new minimum has been found

sum = sum + value;

length++; //increment the count of numbers insert

}

mean = sum / length;

//show results

std::cout << "Minimum: " << minimum << std::endl

<< "Maximum: " << maximum << std::endl

<< "Sum: " << sum << std::endl

<< "Mean: " << mean << std::endl;

system("pause");

return 0;

}

//3.5

#include<iostream>

using namespace std;

bool readFracc(int &numerator, int &denominator) //validate if the fraction is acceptable

{

char separator;

std::cin >> numerator >> separator >> denominator;

if (!cin.fail() && separator == '/') //if the values introduced are, by this order, an integer, a "/" and another integer

{

return true; //the value has been introduced correctly

}

else //if one of them wasn't correct

{

cin.clear(); //we clear the inputs

numerator = 0; //assume numerator and denominator are 0 for valid inputs

denominator = 0;

return false;

}

}

void writeFracc(int numerator, int denominator) //shows the fraction as it was written in the first place

{

std::cout << numerator << "/" << denominator << std::endl;

}

void reduceFracc(int &numerator, int &denominator) //this function returns the fraction in it's reduced form

{

int mdc, resto, a = numerator, b = denominator;

if (a == 0 || b == 0) { mdc = 1; } //euclid's algorithm

else

{

resto = a % b;

while (resto > 0)

{

a = b;

b = resto;

resto = a % b;

}

mdc = b; //find the greatest commun divisor

}

//obtain the simplified fraction

numerator = numerator / b;

denominator = denominator / b;

}

void add(int num1, int den1, int num2, int den2) //adds two fractions

{

int numerator, denominator;

//multiply each fraction by the other one's denominator, because we can only add two fractions with the same denominator

numerator = (num1 \* den2) + (num2 \* den1);

denominator = den1 \* den2;

reduceFracc(numerator, denominator); //reduce the fraction

//show results

writeFracc(numerator, denominator);

}

void sub(int num1, int den1, int num2, int den2) //using a similar process, this function subtracts the two fractions

{

int numerator, denominator;

numerator = (num1 \* den2) - (num2 \* den1);

denominator = den1 \* den2;

reduceFracc(numerator, denominator);

writeFracc(numerator, denominator);

}

void multiply(int num1, int den1, int num2, int den2) //multiplies two fractions

{

int numerator, denominator;

numerator = num1 \* num2; //it multiplies both numerators

denominator = den1 \* den2;//and both denominators

reduceFracc(numerator, denominator); //obtains the reduced form

//shows results

writeFracc(numerator, denominator);

}

void divide(int num1, int den1, int num2, int den2) //divides two fractions

{

int numerator, denominator;

numerator = num1 \* den2; //the division of two fractions is obtained by multiplying the first one's numerator by the second one's denominator

denominator = den1 \* num2;//and the first one's denominator by the second's numerator

reduceFracc(numerator, denominator); //obtain the reduced for of the resulting fraction

//show results

writeFracc(numerator, denominator);

}

int main()

{

int num1, den1, num2, den2, operation;

std::cout << "Insert the first fraction. (n/d) ";

readFracc(num1, den1); //reads the first fraction, validating it

reduceFracc(num1, den1); //simplifies it's form

std::cout << "Insert the second fraction. (n/d) ";

readFracc(num2, den2); //reads and validates the second fraction

reduceFracc(num2, den2); //simplifies it's form

while (true)

{

//simple operations using fractions

std::cout << "What would you like to do?" << std::endl

<< "1. Add 2. Subtract 3. Multiply 4. Divide 5. New fractions Ctrl+Z to exit"

<< std::endl;

std::cin >> operation; //reads the chosen operation from the above

if (cin.eof()) //tests if the user wants to leave by using the end of file

{

return 0;

}

if (cin.fail()) //if there is an error with the inputs

{

cin.clear();

cin.ignore(100000000, '\n');

}

switch (operation) //calls the respective function to operate the fractions

{

case 1: add(num1, den1, num2, den2);

break;

case 2: sub(num1, den1, num2, den2);

break;

case 3: multiply(num1, den1, num2, den2);

break;

case 4: divide(num1, den1, num2, den2);

break;

case 5: break;

}

}

system("pause");

}

//4.3

#include<iostream>

#include<string>

#include<stdio.h>

void decompose(std::string compound)

{

int i = 0;

std::cout << "The compound is formed by ";

while (i < compound.length())

{

if(isupper(compound[i]))

{

if(isupper(compound[i+1]))

{

std::cout << compound[i] << " and ";

}

else std::cout << compound[i];

}

if (islower(compound[i]))

{

if (isupper(compound[i + 1]))

{

std::cout << compound[i] << " and ";

}

else std::cout << compound[i];

}

if (isdigit(compound[i]) && isupper(compound[i + 1]))

{

std::cout << " and ";

}

if(i == (compound.length() - 1))

{

std::cout << std::endl;

}

i++;

}

}

int main()

{

std::string compound;

while (!std::cin.eof())

{

std::cin >> compound;

decompose(compound);

}

system("pause");

return 0;

}

//4.6 c

#include<iostream>

void readIntArray(int a[], int nElem)

{

int i = 0;

while (i < nElem)

{

std::cout << "índice " << i << ": ";

std::cin >> a[i];

std::cout << std::endl;

i++;

}

}

int searchValueInIntArray(const int a[], int nElem, int value)

{

int i = 0, indice = -1;

while (i < nElem)

{

if (a[i] == value)

{

indice = i;

break;

}

i++;

}

return indice;

}

int main()

{

const int nElem = 10;

int a[nElem], value, indice;

readIntArray(a, nElem);

while (!std::cin.eof())

{

std::cout << "What value do you want to search on the array? ";

std::cin >> value;

indice = searchValueInIntArray(a, nElem, value);

if (indice == -1)

{

std::cout << "The value is not found in the array." << std::endl;

}

else

{

std::cout << "The value is found in indice " << indice << " of the array." << std::endl;

}

}

system("pause");

return 0;

}