**Programação**

Índice

[TP1-a 5](#_Toc478545681)

[TP1-b 6](#_Toc478545682)

[1. Tipos de dados. Expressões aritméticas. Entrada e saída. 8](#_Toc478545683)

[1.1 8](#_Toc478545684)

[1.2 9](#_Toc478545685)

[1.3 10](#_Toc478545686)

[1.4 11](#_Toc478545687)

[1.5 12](#_Toc478545688)

[1.6 13](#_Toc478545689)

[2. Estruturas de selecção e de repetição 14](#_Toc478545690)

[2.1 14](#_Toc478545691)

[2.2 15](#_Toc478545692)

[2.2-a) 15](#_Toc478545693)

[2.2-b) 16](#_Toc478545694)

[2.2-c) 17](#_Toc478545695)

[2.3 18](#_Toc478545696)

[2.4 19](#_Toc478545697)

[2.6 20](#_Toc478545698)

[2.6-a) 20](#_Toc478545699)

[2.6-b) 20](#_Toc478545700)

[2.6-c) 22](#_Toc478545701)

[2.7 23](#_Toc478545702)

[2.7-a) 23](#_Toc478545703)

[2.7-b) 23](#_Toc478545704)

[2.8 24](#_Toc478545705)

[2.9 25](#_Toc478545706)

[2.9-a) 26](#_Toc478545707)

[2.9-b) 27](#_Toc478545708)

[2.9-c) 28](#_Toc478545709)

[2.10 29](#_Toc478545710)

[2.10-a) 29](#_Toc478545711)

[2.10-b) 30](#_Toc478545712)

[2.11 31](#_Toc478545713)

[2.11-a) 31](#_Toc478545714)

[2.11-b) 32](#_Toc478545715)

[2.11-c) 33](#_Toc478545716)

[2.12 34](#_Toc478545717)

[2.12-a) 34](#_Toc478545718)

[2.12-b) 36](#_Toc478545719)

[2.12-c) 37](#_Toc478545720)

[2.13 39](#_Toc478545721)

[2.14 40](#_Toc478545722)

[2.14-a) 41](#_Toc478545723)

[2.14-b) 42](#_Toc478545724)

[2.15 43](#_Toc478545725)

[3. Funções 45](#_Toc478545726)

[3.1 45](#_Toc478545727)

[3.2 46](#_Toc478545728)

[3.3 46](#_Toc478545729)

[3.3-a) 47](#_Toc478545730)

[3.3-b) 47](#_Toc478545731)

[3.4 49](#_Toc478545732)

[3.5 50](#_Toc478545733)

[3.5-a) 51](#_Toc478545734)

[3.5-b) 52](#_Toc478545735)

[3.5-c) 53](#_Toc478545736)

[3.5-d) 54](#_Toc478545737)

[3.5-e) 55](#_Toc478545738)

[3.6 57](#_Toc478545739)

[3.6-a) 58](#_Toc478545740)

[3.6-b) 59](#_Toc478545741)

[3.6-c) 60](#_Toc478545742)

[3.6-d) 62](#_Toc478545743)

[3.6-e) 65](#_Toc478545744)

[3.7 69](#_Toc478545745)

[3.8 70](#_Toc478545746)

[3.9 71](#_Toc478545747)

[4. Arrays, Vectors, Strings, Structs e Files 73](#_Toc478545748)

[4.1 73](#_Toc478545749)

[4.1-a) 73](#_Toc478545750)

[4.1-b) 74](#_Toc478545751)

[4.2 75](#_Toc478545752)

[4.3 76](#_Toc478545753)

[4.4 77](#_Toc478545754)

[4.5 78](#_Toc478545755)

[4.6 82](#_Toc478545756)

[4.6-a) 83](#_Toc478545757)

[4.6-b) 84](#_Toc478545758)

[4.6-c) 85](#_Toc478545759)

[4.6-d) 86](#_Toc478545760)

[4.6-e) 87](#_Toc478545761)

[4.7 88](#_Toc478545762)

[4.7-a) 89](#_Toc478545763)

[4.7-b) 91](#_Toc478545764)

[4.8 92](#_Toc478545765)

[4.8-a) 93](#_Toc478545766)

[4.8-b) 94](#_Toc478545767)

[4.13 95](#_Toc478545768)

# TP1-a

#include <iostream>

#include <ctime>

using namespace std;

int main(void)

{

// variable declarations and initialization

int operand1, operand2, result, answer;

char operators[4] = { '+','-','\*','/' }; char operation; // why not 'operator' ...?

// initialize random number generator

srand((unsigned int)time(NULL));

// randomly generate operands and operator

operand1 = rand() % 10 + 1;

operand2 = rand() % 10 + 1;

operation = operators[rand() % 4];

// calculate the correct result

switch (operation)

{

case '+':

result = operand1 + operand2;

break;

case '-':

result = operand1 - operand2;

break;

case '\*':

result = operand1 \* operand2;

break;

case '/':

result = operand1 / operand2;

break;

}

// ask the answer from the user

cout << operand1 << " " << operation << " " << operand2 << " ? ";

cin >> answer;

// verify if the answer of the user is correct

if (answer == result)

cout << "Correct answer. Congratulations\n";

else

cout << "Wrong answer ...\n";

return 0;

}



# TP1-b

#include <iostream>

#include <ctime>

using namespace std;

int main(void)

{

// variable declarations and initialization

int operand1, operand2, result, answer, numCorrectAnswers;

char operators[4] = { '+','-','\*','/' };

char operation; // why not 'operator' ...?

const int MAX\_OPERAND\_VALUE = 10;

const int NUM\_OPERATIONS = 10;

int maxOperandValue;

int numOperations;

numCorrectAnswers = 0;

// initialize random number generator

srand((unsigned int)time(NULL));

cout << "Maximum operand value? "; cin >> maxOperandValue;

cout << "Number of operations ? "; cin >> numOperations;

// randomly generate operands and operator

for (int i = 1; i <= numOperations; i++)

{

operand1 = rand() % maxOperandValue + 1;

operand2 = rand() % maxOperandValue + 1;

operation = operators[rand() % 4];

// calculate the correct result

switch (operation)

{

case '+':

result = operand1 + operand2;

break;

case '-':

result = operand1 - operand2;

break;

case '\*':

result = operand1 \* operand2;

break;

case '/':

result = operand1 / operand2;

break;

}

// ask the answer from the user

cout << operand1 << " " << operation << " " << operand2 << " ? ";

cin >> answer;

// verify if the answer of the user is correct

if (answer == result)

{

cout << "Correct answer. Congratulations\n";

numCorrectAnswers++;

}

else

cout << "Wrong answer ...\n";

}

// show final result

cout << "Number of correct answers = " << numCorrectAnswers << endl;

// classify results

if (numCorrectAnswers <= (int)(0.3 \* numOperations))

cout << "VERY BAD ......\n";

else

if (numCorrectAnswers <= (int)(0.5 \* numOperations))

cout << "POOR...\n";

else

if (numCorrectAnswers <= (int)(0.7 \* numOperations))

cout << "FAIR\n";

else

if (numCorrectAnswers <= (int)(0.9 \* numOperations))

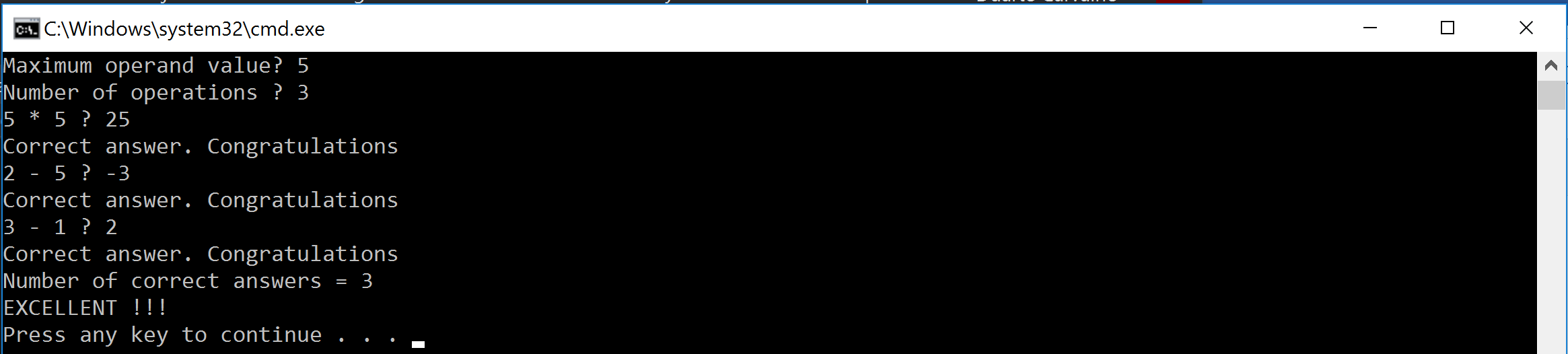
cout << "GOOD !\n";

else

cout << "EXCELLENT !!!\n";

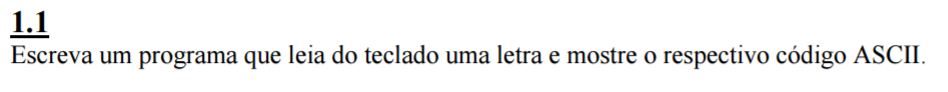
return 0;

}



# Tipos de dados. Expressões aritméticas. Entrada e saída.

## 1.1



#include <iostream>

#include <ctime>

using namespace std;

int main(void)

{

char input;

cout << "Insira letras\n";

while (1)

{

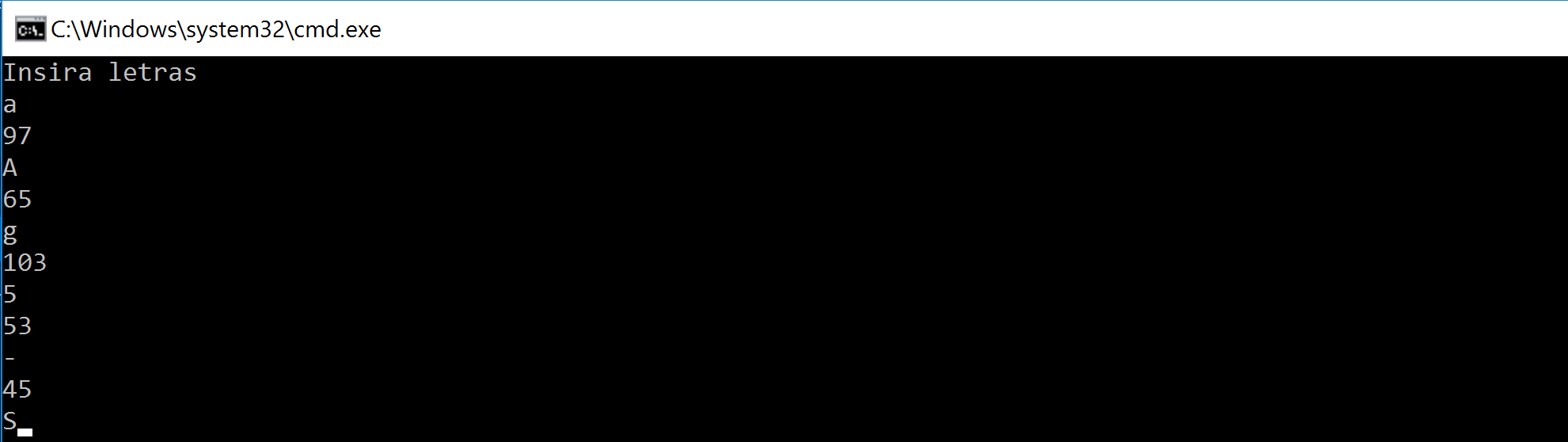
cin >> input;

cout << (int)input << endl;

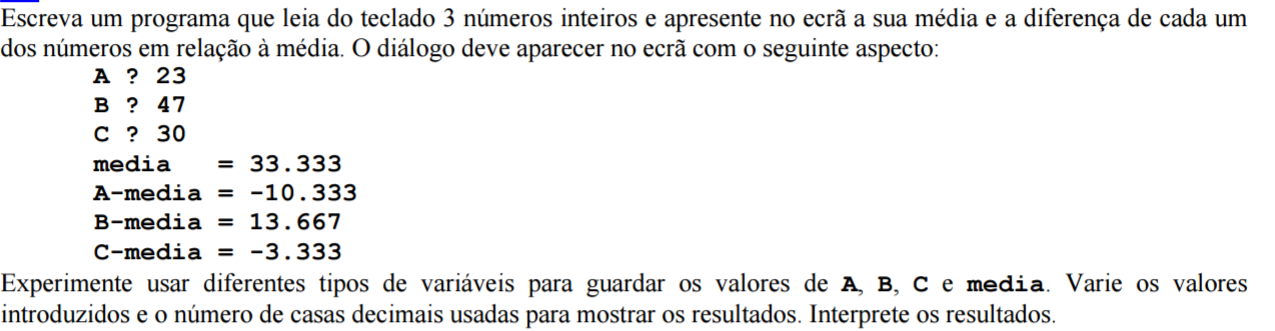
}

return 0;

}



## 1.2



#include <iostream>

#include <ctime>

using namespace std;

int main(void)

{

int x1, x2, x3;

float media = 0;

cout << "Insira 3 valores!\n";

cout << "A?\n";

cin >> x1;

cout << "B?\n";

cin >> x2;

cout << "C?\n";

cin >> x3;

media = (x1 + x2 + x3) / 3;

cout << "Media = " << media << endl;

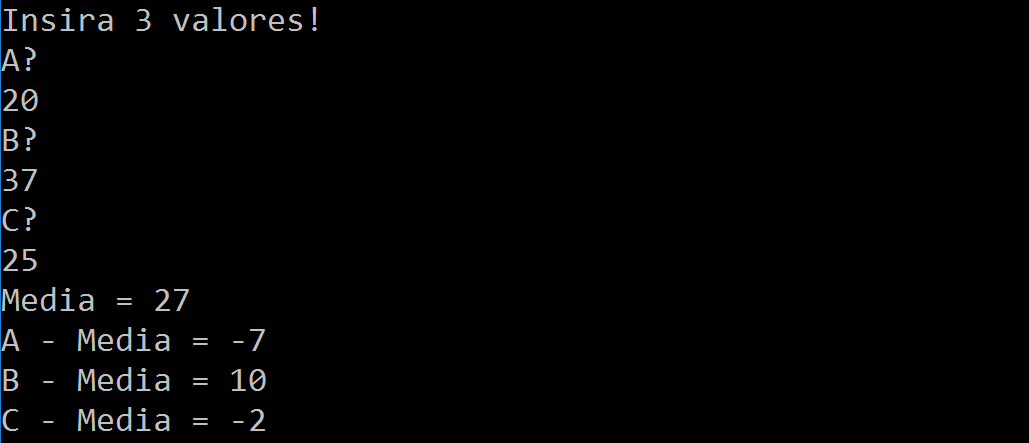
cout << "A - Media = " << x1 - media << endl;

cout << "B - Media = " << x2 - media << endl;

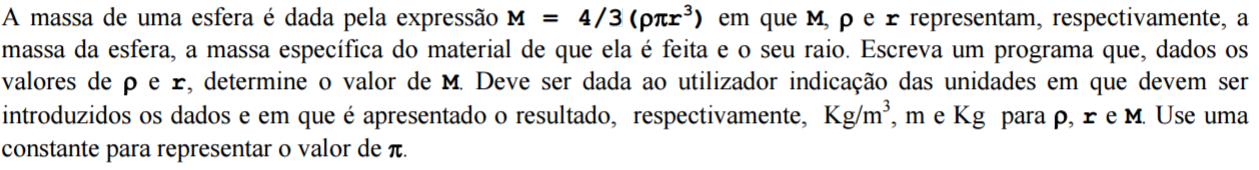
cout << "C - Media = " << x3 - media << endl;

return 0;

}



## 1.3



#include <iostream>

#include <ctime>

using namespace std;

int main(void)

{

float p, raio;

const double pi = 3.1415;

cout << "Indique o valor da massa do material! (Kg/m^3)\n";

cin >> p;

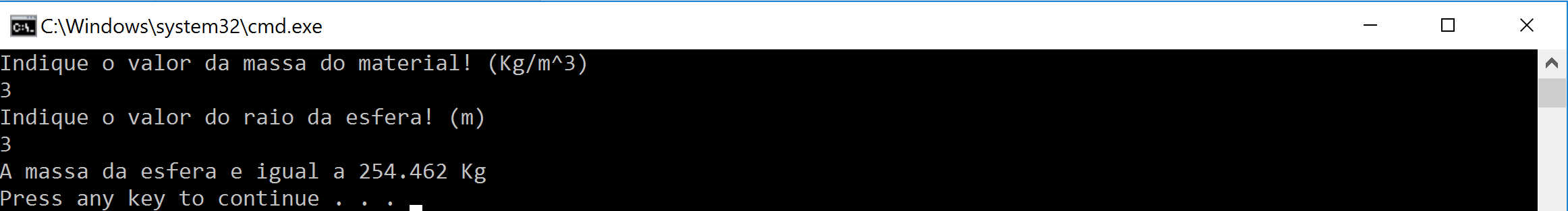
cout << "Indique o valor do raio da esfera! (m)\n";

cin >> raio;

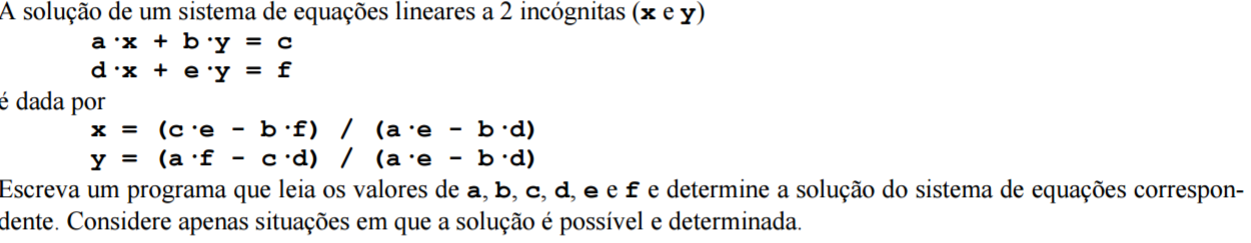
cout << "A massa da esfera e igual a " << 4 / 3 \* p\*pi\*raio\*raio\*raio << " Kg\n";

return 0;

}



## 1.4



#include <iostream>

#include <ctime>

#include <vector>

using namespace std;

int main(void)

{

float x, y;

vector<float> sistema (6);

cout << "a.x + b.y = c" << endl << "d.x + e.y = f" << endl;

for (int i = 97; i < 103; i++)

{

cout << "Insira " << (char)i << endl;

cin >> sistema[i - 97];

}

for (int i = 0; i < 6; i++)

cout << (char) (i + 97) << " = " << sistema[i] << endl;

x = (sistema[2] \* sistema[4] - sistema[1] \* sistema[5]) / (sistema[0] \* sistema[4] - sistema[1] \* sistema[3]);

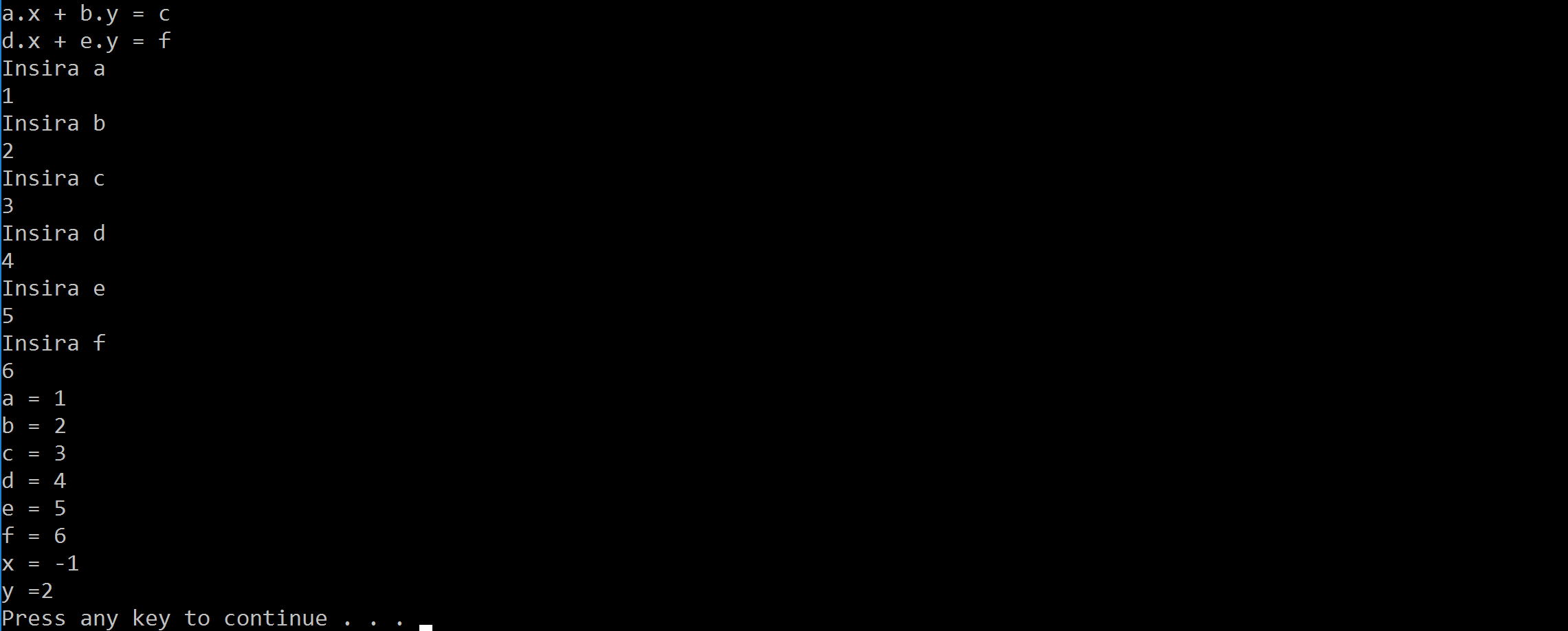
y = (sistema[0] \* sistema[5] - sistema[2] \* sistema[3]) / (sistema[0] \* sistema[4] - sistema[1] \* sistema[3]);

cout << "x = " << x << endl;

cout << "y =" << y << endl;

return 0;

}



## 1.5



#include <iostream>

#include <ctime>

#include <vector>

using namespace std;

int main(void)

{

long int total = 0;

vector<int> t1 (3);

vector<int> t2 (3);

vector<int> final (4);

cout << "Tempo 1 (horas minutos segundos) ?";

cin >> t1[0] >> t1[1] >> t1[2];

cout << "Tempo 2 (horas minutos segundos) ?";

cin >> t2[0] >> t2[1] >> t2[2];

total = (t1[0] + t2[0]) \* 3600 + (t1[1] + t2[1]) \* 60 + t1[2] + t2[2];

final[0] = total / (3600 \* 24);

total -= final[0] \* (3600 \* 24);

final[1] = total / 3600;

total -= final[1] \* (3600);

final[2] = total / 60;

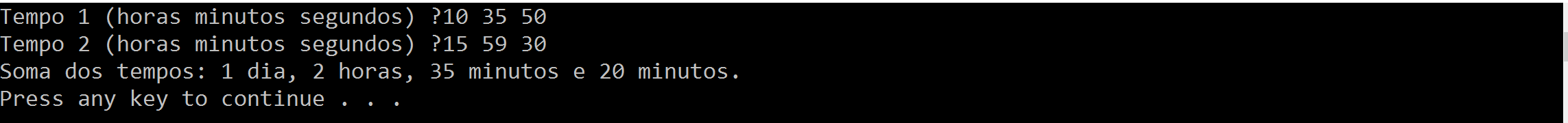
total -= final[2] \* 60;

final[3] =total;

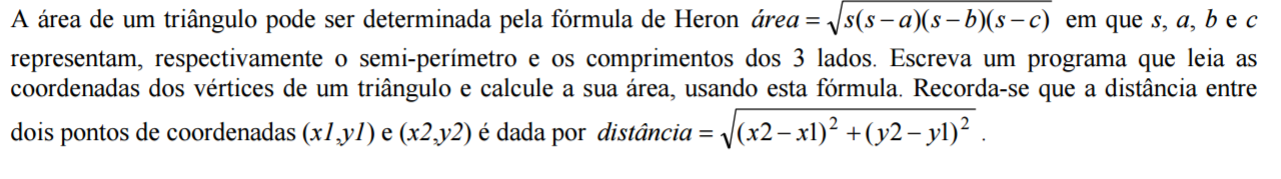
cout << "Soma dos tempos: " << final[0] << " dia, " << final[1] << " horas, " << final[2] << " minutos e " << final[3] << " minutos.\n";

return 0;

}



## 1.6



#include <iostream>

#include <ctime>

#include <vector>

#include <cmath>

using namespace std;

int main(void)

{

vector<int> a(2);

vector<int> b(2);

vector<int> c(2);

double ab, ac, bc, s, area;

cout << "Introduza as coordenadas do ponto A:";

cin >> a[0] >> a[1];

cout << "Introduza as coordenadas do ponto B:";

cin >> b[0] >> b[1];

cout << "Introduza as coordenadas do ponto C:";

cin >> c[0] >> c[1];

/\*Calcular o tamano das Arestas\*/

ab = sqrt(pow((b[0] - a[0]), 2) + pow((b[1] - a[1]), 2));

ac = sqrt(pow((c[0] - a[0]), 2) + pow((c[1] - a[1]), 2));

bc = sqrt(pow((c[0] - b[0]), 2) + pow((c[1] - b[1]), 2));

/\*Calcular o semi-perímetro\*/

s = (ab + ac + bc) / 2;

/\*Calcular Área (fórmula de Heron)\*/

area = sqrt(s\*(s - ab)\*(s - ac)\*(s - bc));

cout << "A area do triangulo e = " << area << endl;

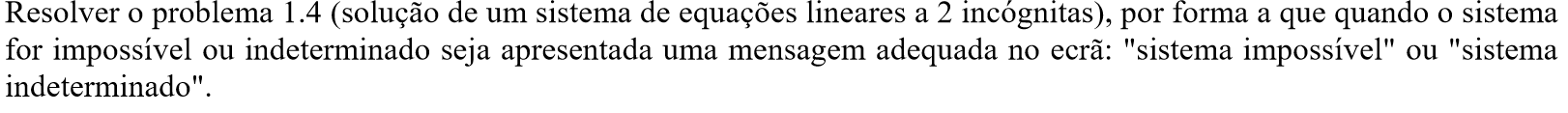
return 0;

}



# Estruturas de selecção e de repetição

## 2.1



#include <iostream>

#include <ctime>

#include <vector>

#include <cmath>

using namespace std;

int main(void)

{

float x, y;

vector<float> sistema(6);

cout << "a.x + b.y = c" << endl << "d.x + e.y = f" << endl;

for (int i = 97; i < 103; i++)

{

cout << "Insira " << (char)i << endl;

cin >> sistema[i - 97];

}

for (int i = 0; i < 6; i++)

cout << (char)(i + 97) << " = " << sistema[i] << endl;

if (sistema[0] \* sistema[4] - sistema[3] \* sistema[1] != 0)

{

x = (sistema[2] \* sistema[4] - sistema[1] \* sistema[5]) / (sistema[0] \* sistema[4] - sistema[1] \* sistema[3]);

y = (sistema[0] \* sistema[5] - sistema[2] \* sistema[3]) / (sistema[0] \* sistema[4] - sistema[1] \* sistema[3]);

cout << "x = " << x << endl;

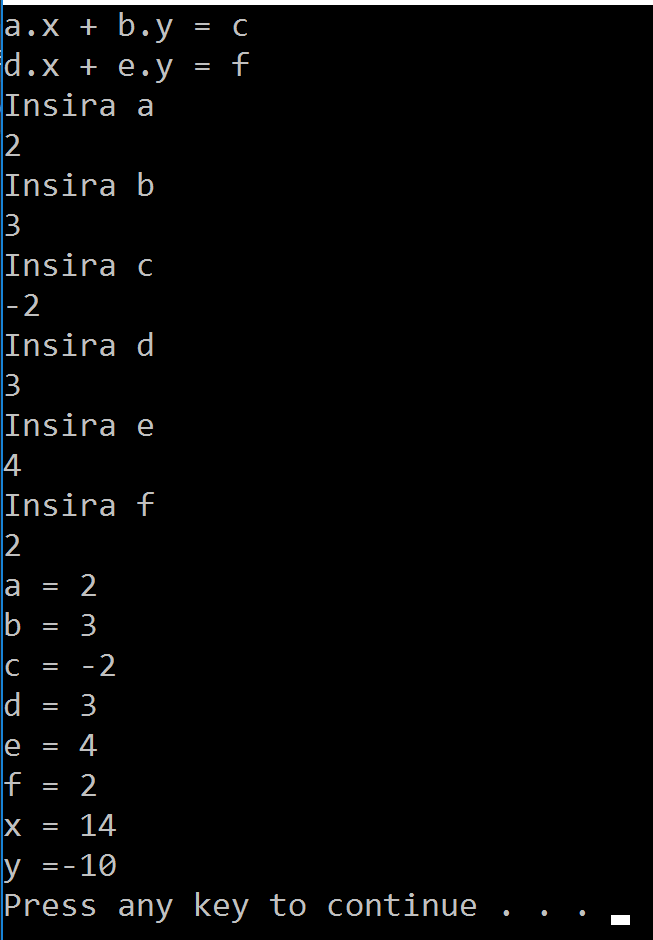
cout << "y =" << y << endl;

}

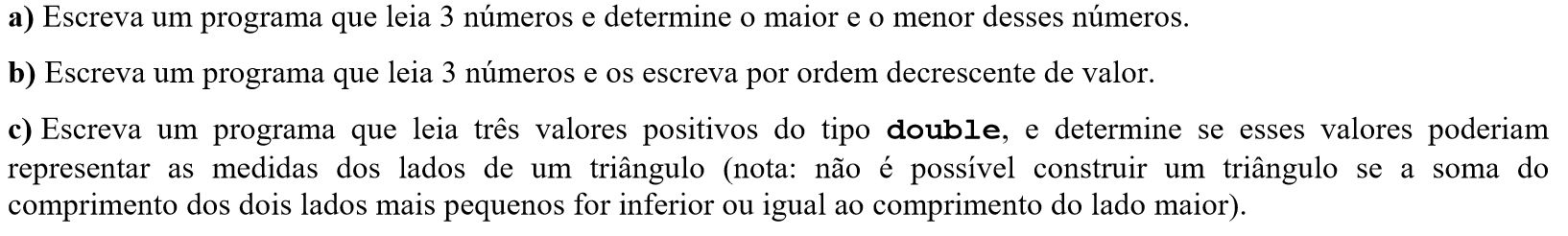
else

cout << "Sistema nao e possivel determinado!" << endl;

return 0;

}

## 2.2



### 2.2-a)

#include <iostream>

#include <ctime>

#include <vector>

#include <cmath>

using namespace std;

int main(void)

{

float aux;

vector<float> valores(3);

cout << "Insira 3 valores!" << endl;

cin >> valores[0] >> valores[1] >> valores[2];

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

{

for (int j = i + 1; j < 3; j++)

{

if (valores[i] < valores[j])

{

aux = valores[i];

valores[i] = valores[j];

valores[j] = aux;

}

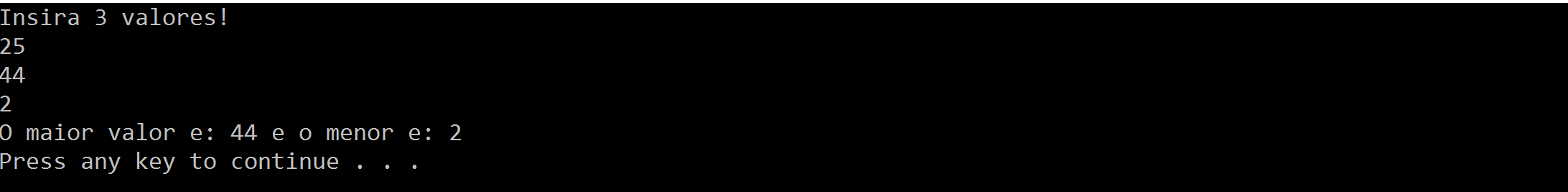
}

}

cout << "O maior valor e: " << valores[0] << " e o menor e: " << valores[2] << endl;

return 0;

}



### 2.2-b)

#include <iostream>

#include <ctime>

#include <vector>

#include <cmath>

using namespace std;

int main(void)

{

float aux;

vector<float> valores(3);

cout << "Insira 3 valores!" << endl;

cin >> valores[0] >> valores[1] >> valores[2];

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

{

for (int j = i + 1; j < 3; j++)

{

if (valores[i] < valores[j])

{

aux = valores[i];

valores[i] = valores[j];

valores[j] = aux;

}

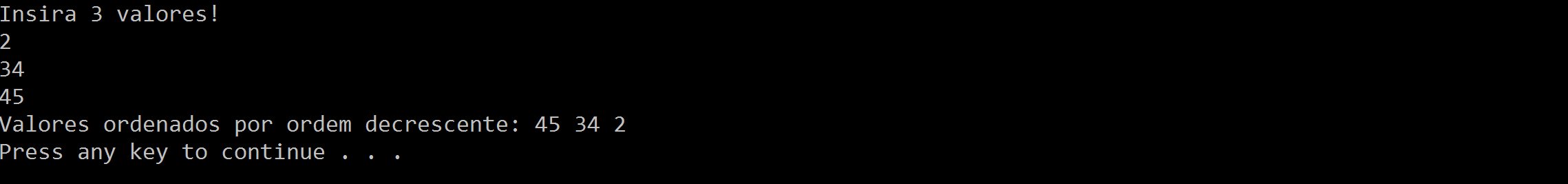
}

}

cout << "Valores ordenados por ordem decrescente: " << valores[0] << " " << valores[1] << " " << valores[2] << endl;

return 0;

}



### 2.2-c)

#include <iostream>

#include <ctime>

#include <vector>

#include <cmath>

using namespace std;

int main(void)

{

float aux;

vector<float> valores(3);

cout << "Insira 3 valores!" << endl;

cin >> valores[0] >> valores[1] >> valores[2];

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

{

for (int j = i + 1; j < 3; j++)

{

if (valores[i] < valores[j])

{

aux = valores[i];

valores[i] = valores[j];

valores[j] = aux;

}

}

}

if (valores[0] > valores[1] + valores[2])

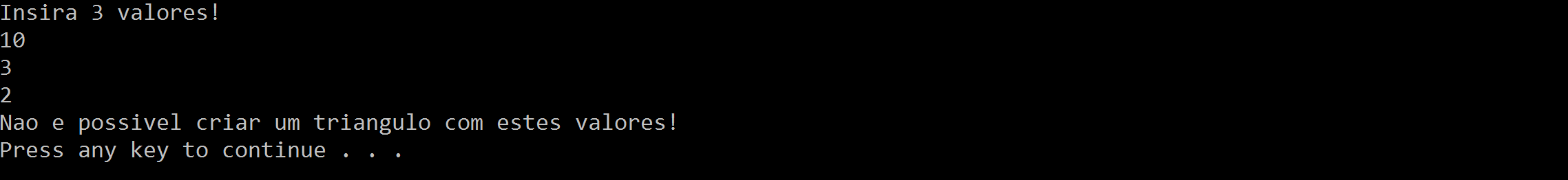
cout << "Nao e possivel criar um triangulo com estes valores!\n";

else

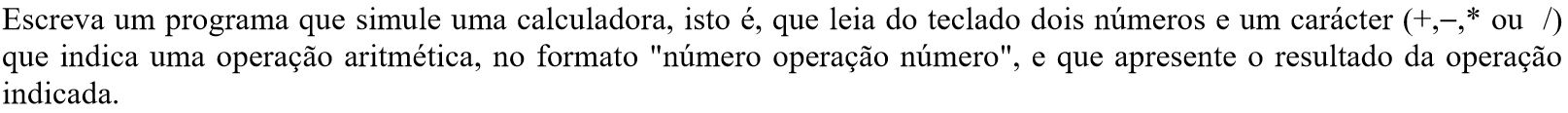
cout << "E possivel criar um triangulo com estes valores!\n";

return 0;

}



## 2.3



#include <iostream>

#include <ctime>

#include <vector>

#include <cmath>

using namespace std;

int main(void)

{

char op;

float x1, x2, resultado;

cout << "Indique dois numeros:";

cin >> x1 >> x2;

cout << "Indique o operador da operacao que pretende realizar:";

cin >> op;

if (op == '+')

resultado = x1 + x2;

else if (op == '-')

resultado = x1 - x2;

else if (op == '\*')

resultado = x1 \* x2;

else

if (x2 != 0)

resultado = x1 / x2;

else

{

cout << "Nao e possivel dividir por 0!\n";

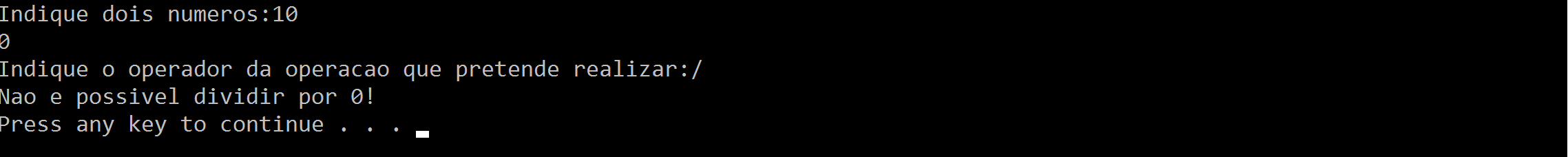
return 0;

}

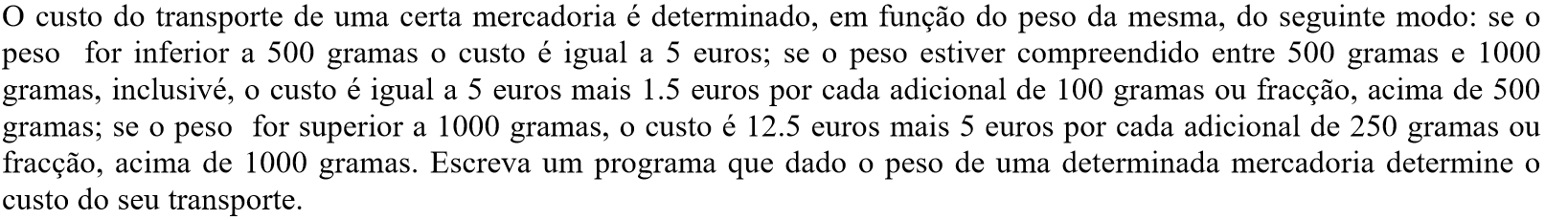
cout << x1 << " " << op << " " << x2 << " = " << resultado << endl;

return 0;

}



## 2.4



#include <iostream>

using namespace std;

int main(void)

{

int peso;

float custo;

cout << "Introduza o peso em gramas: ";

cin >> peso;

if (peso < 500)

custo = 5;

else if (peso >= 500 && peso <= 1000)

custo = 5 + 1.5 \* ((int)(peso - 500) / 100);

else if (peso > 1000)

custo = 12.5 + 5 \* ((int)(peso - 1000) / 250);

else

{

cout << "Peso inválido!\n";

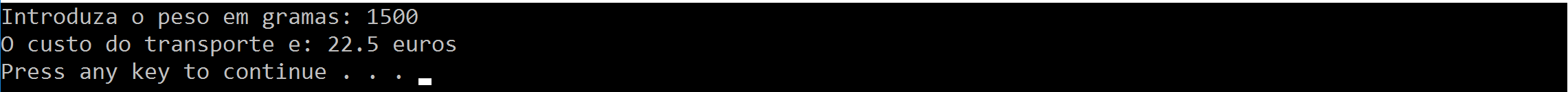
return 0;

}

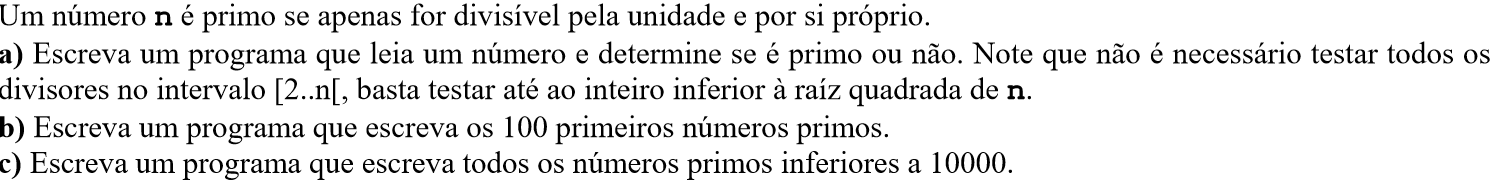
cout << "O custo do transporte e: " << custo << " euros\n";

return 0;

}



## 2.6



### 2.6-a)

#include <iostream>

#include <cmath>

using namespace std;

int primo(int num)

{

if (num % 2 == 0)

return 1;

for (int i = 3; i < sqrt(num); i += 2)

{

if (num % i == 0)

return 1;

}

return 0;

}

int main(void)

{

int x;

cout << "Introduza um numero: ";

cin >> x;

if (primo(x) == 0)

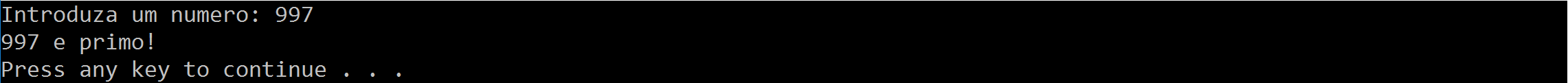
cout << x << " e primo!\n";

else

cout << x << " nao e primo!\n";

return 0;

}



### 2.6-b)

#include <iostream>

#include <cmath>

using namespace std;

int primo(int num)

{

if (num % 2 == 0)

return 1;

for (int i = 3; i < sqrt(num) + 1; i += 2)

{

if (num % i == 0)

return 1;

}

return 0;

}

int main(void)

{

int n\_primos = 0;

int i = 1;

while (n\_primos < 100)

{

if (primo(i) == 0)

{

n\_primos++;

//cout << "Primo numero " << n\_primos << ": " << i << endl;

if (n\_primos == 1)

cout << i << " ,";

else if (n\_primos == 100)

cout << " " << i << endl;

else

cout << " " << i << " ,";

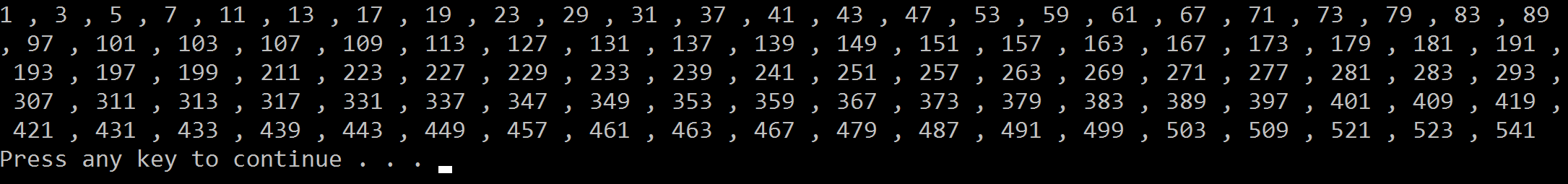
}

i++;

}

return 0;

}



### 2.6-c)

#include <iostream>

#include <cmath>

using namespace std;

int primo(int num)

{

if (num % 2 == 0)

return 1;

for (int i = 3; i < sqrt(num) + 1; i += 2)

{

if (num % i == 0)

return 1;

}

return 0;

}

int main(void)

{

int n\_primos = 0;

int i = 1;

while (i < 10000)

{

if (primo(i) == 0)

{

n\_primos++;

cout << "Primo numero " << n\_primos << ": " << i << endl;

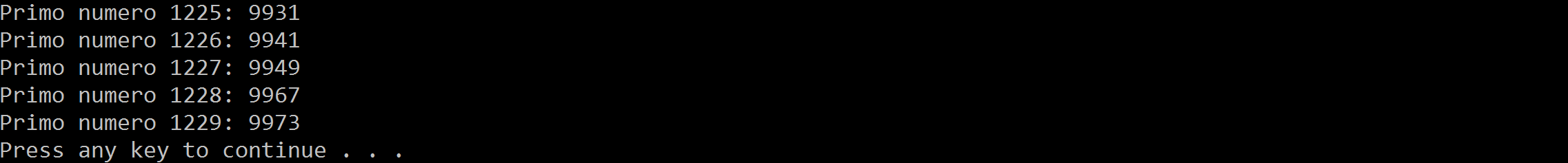
}

i++;

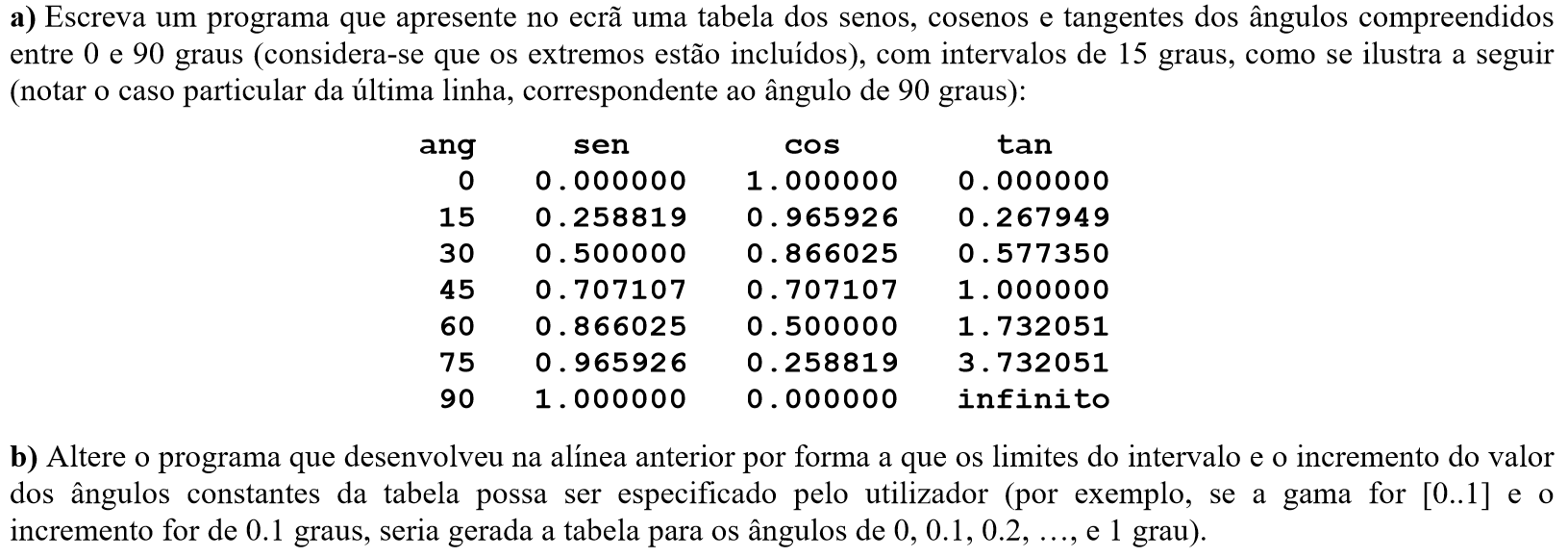
}

return 0;

}



## 2.7



### 2.7-a)

#include <iostream>

#include <cmath>

#include <iomanip>

#define PI 3.14159265

using namespace std;

int main(void)

{

cout << "ang\tsen\tcos\ttan\n";

for (int i = 0; i <= 90; i+=15)

cout << setprecision(5) << i << "\t" << sin(i \* PI / 180.0) << "\t" << cos(i \* PI / 180.0) << "\t" << tan(i \* PI / 180.0) << endl;

return 0;

}

### 2.7-b)

#include <iostream>

#include <cmath>

#include <iomanip>

#define PI 3.14159265

using namespace std;

int main(void)

{

float x1, x2, h;

cout << "Indique os intervalos e o incremento (x1,x2,h):";

cin >> x1 >> x2 >> h;

cout << "ang\tsen\tcos\ttan\n";

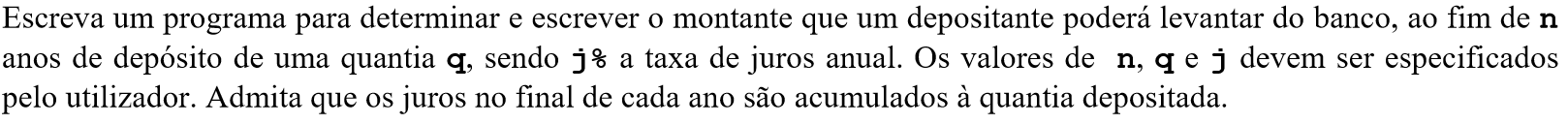
for (x1; x1 <= x2; x1 += h)

cout << setprecision(3) << x1 << "\t" << sin(x1 \* PI / 180.0) << "\t" << cos(x1 \* PI / 180.0) << "\t" << tan(x1 \* PI / 180.0) << endl;

return 0;

}

## 2.8



#include <iostream>

using namespace std;

int main(void)

{

int anos;

float juros, quantia;

cout << "Insira os dados (quantia-juros-anos): ";

cin >> quantia >> juros >> anos;

for (int i = 1; i <= anos; i++)

{

quantia += quantia\*juros/100;

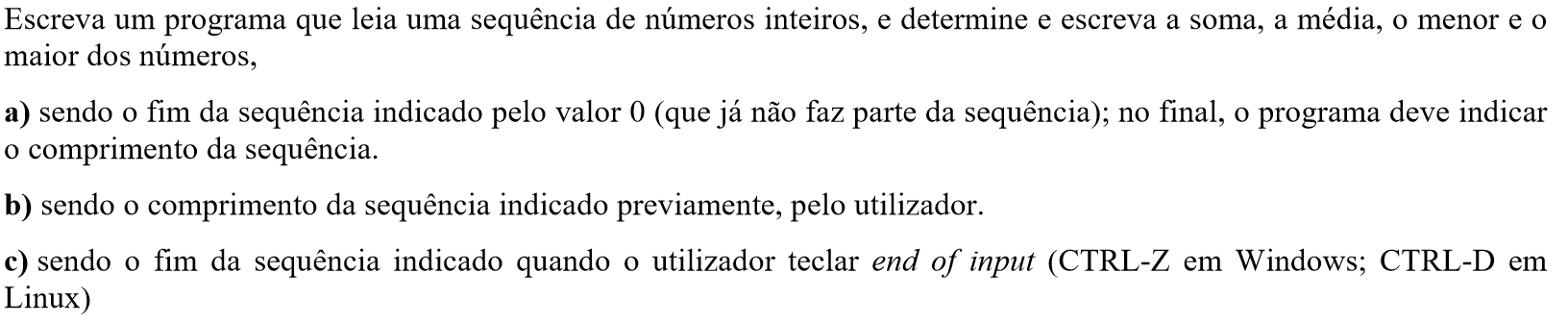
cout << "Ao fim de " << i << " anos tera: " << quantia << " euros\n";

}

return 0;

}

## 2.9



### 2.9-a)

#include <iostream>

#include <vector>

using namespace std;

int main(void)

{

int x, soma;

double media;

vector<int> valores;

soma = 0;

while (1)

{

cin >> x;

if (x == 0)

break;

else

valores.push\_back(x); //Inserir no fim do vetor

}

for (int j = 0; j < (int)valores.size(); j++) //imprimir o vetor

cout << valores[j] << " ";

cout << endl;

/\*Ordenar o vetor e aproveitar para fazer a soma\*/

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

{

for (int j = i + 1; j < (int)valores.size(); j++)

{

if (valores[j] > valores[i])

{

x = valores[j];

valores[j] = valores[i];

valores[i] = x;

}

}

soma += valores[i];

}

for (int j = 0; j < (int)valores.size(); j++) //imprimir o vetor ordenado

cout << valores[j] << " ";

cout << endl;

media = soma \* 1.0 / valores.size(); //calcular a media

cout << "A soma dos elementos do vetor e: " << soma << endl;

cout << "Tendo a media sido: " << media << " e o minimo e maximo: " << valores[0] << " " << valores[valores.size() - 1] << endl;

return 0;

}

### 2.9-b)

#include <iostream>

#include <vector>

using namespace std;

int main(void)

{

int x, soma, n\_num;

double media;

vector<int> valores;

soma = 0;

cout << "Introduza o numero de valores que pretende inserir: ";

cin >> n\_num;

cout << "Insira os numeros:";

for(int i = 0; i< n\_num; i++)

{

cin >> x;

valores.push\_back(x); //Inserir no fim do vetor

}

for (int j = 0; j < (int)valores.size(); j++) //imprimir o vetor

cout << valores[j] << " ";

cout << endl;

/\*Ordenar o vetor e aproveitar para fazer a soma\*/

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

{

for (int j = i + 1; j < (int)valores.size(); j++)

{

if (valores[j] > valores[i])

{

x = valores[j];

valores[j] = valores[i];

valores[i] = x;

}

}

soma += valores[i];

}

for (int j = 0; j < (int)valores.size(); j++) //imprimir o vetor ordenado

cout << valores[j] << " ";

cout << endl;

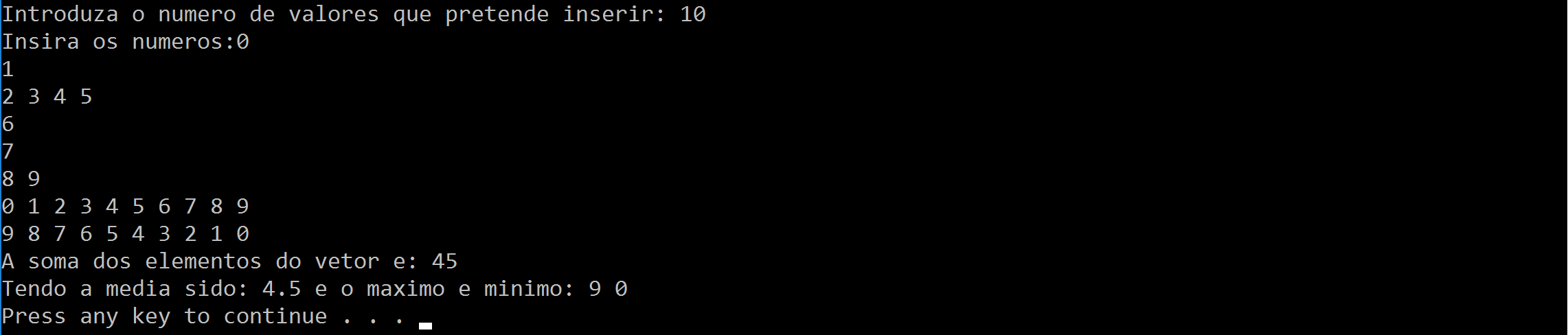
media = soma \* 1.0 / valores.size(); //calcular a media

cout << "A soma dos elementos do vetor e: " << soma << endl;

cout << "Tendo a media sido: " << media << " e o maximo e minimo: " << valores[0] << " " << valores[valores.size() - 1] << endl;

return 0;

}



### 2.9-c)

#include <iostream>

#include <vector>

using namespace std;

int main(void)

{

int x, soma;

double media;

vector<int> valores;

soma = 0;

cout << "Insira os numeros:";

while(cin.eof() != true)

{

cin >> x;

valores.push\_back(x); //Inserir no fim do vetor

}

for (int j = 0; j < (int)valores.size(); j++) //imprimir o vetor

cout << valores[j] << " ";

cout << endl;

/\*Ordenar o vetor e aproveitar para fazer a soma\*/

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

{

for (int j = i + 1; j < (int)valores.size(); j++)

{

if (valores[j] > valores[i])

{

x = valores[j];

valores[j] = valores[i];

valores[i] = x;

}

}

soma += valores[i];

}

for (int j = 0; j < (int)valores.size(); j++) //imprimir o vetor ordenado

cout << valores[j] << " ";

cout << endl;

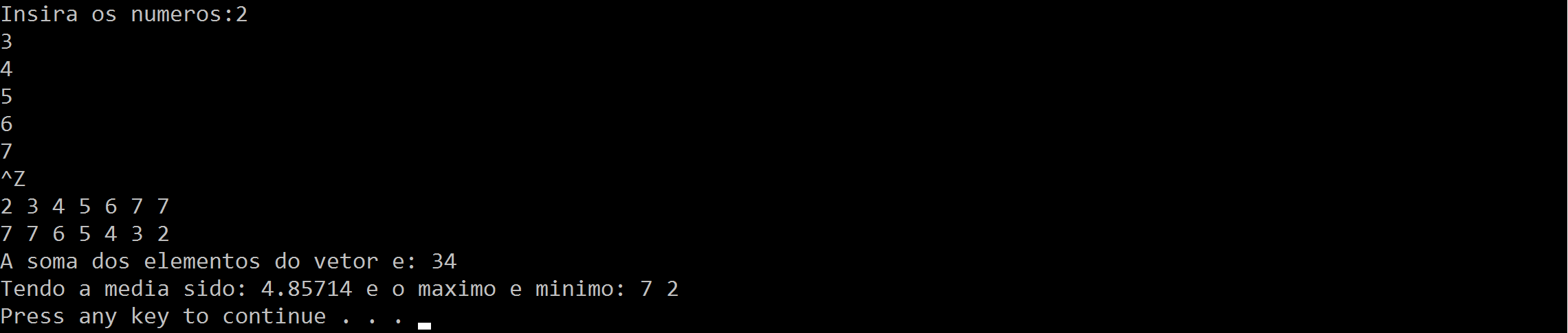
media = soma \* 1.0 / valores.size(); //calcular a media

cout << "A soma dos elementos do vetor e: " << soma << endl;

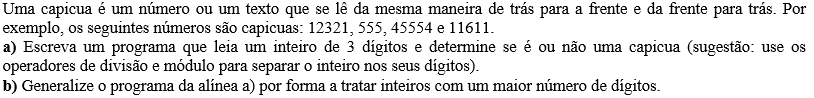
cout << "Tendo a media sido: " << media << " e o maximo e minimo: " << valores[0] << " " << valores[valores.size() - 1] << endl;

return 0;

}



### 2.10



### 2.10-a)

#include <iostream>

#include <string>

using namespace std;

int main(void)

{

string capicua;

cout << "Insira um numero de tres digitos: ";

cin >> capicua;

if (capicua.size() == 3)

{

if (capicua[0] == capicua[2])

cout << capicua << " e capicua!" << endl;

else

cout << capicua << " nao e capica" << endl;

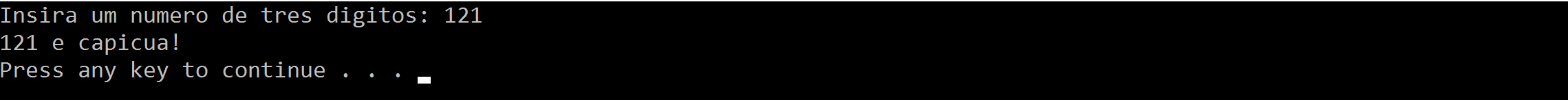
}

else

cout << capicua << " nao tem tres digitos\n";

return 0;

}



### 2.10-b)

#include <iostream>

#include <string>

using namespace std;

int verifica\_capicua(string str, int x0, int x1)

{

if (str.size() % 2 == 0)

{

if (x0 == str.size() / 2 - 1)

{

if (str[x0] == str[x1])

return 0;

}

else

{

if (str[x0] == str[x1])

return verifica\_capicua(str,x0 + 1, x1 - 1);

else

return 1;

}

}

else

{

if (x0 == x1)

return 0;

else

{

if (str[x0] == str[x1])

return verifica\_capicua(str, x0 + 1, x1 - 1);

else

return 1;

}

}

}

int main(void)

{

string capicua;

cout << "Insira um numero de tres digitos: ";

cin >> capicua;

if (verifica\_capicua(capicua, 0, capicua.size() - 1) == 0)

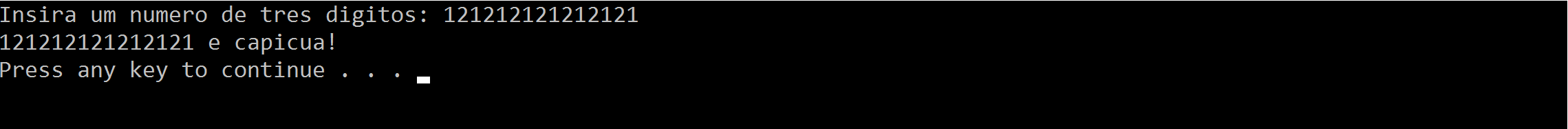
cout << capicua << " e capicua!\n";

else

cout << capicua << " nao e capica\n";

return 0;

}



## 2.11

### 2.11-a)

#include <iostream>

using namespace std;

int main(void)

{

int n, divisor;

double termo = 4;

divisor = 1;

cout << "Insira o numero de termos: ";

cin >> n;

cout << "4";

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

{

divisor += 2;

if (i % 2 == 0)

{

termo -= 4 \* 1.0 / divisor;

cout << " - " << "4/" << divisor;

}

else

{

termo += 4 \* 1.0 / divisor;

cout << " + " << "4/" << divisor;

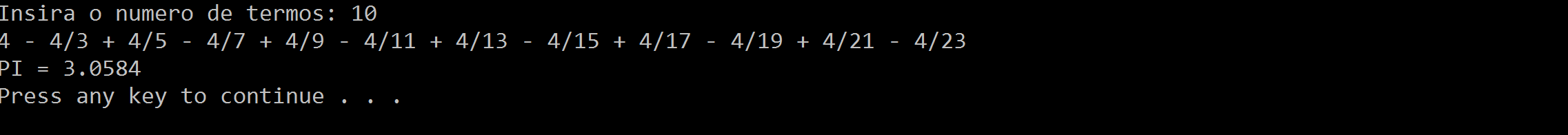
}

}

cout << endl << "PI = " << termo << endl;

return 0;

}



/\* Fingir que está la 3.14 xD com 100000 deu 3.14158\*/

### 2.11-b)

#include <iostream>

using namespace std;

int fatorial(int n)

{

if (n == 1)

return 1;

else

return n \* fatorial(n - 1);

}

int main(void)

{

int n;

double termo = 1;

cout << "Insira o numero de termos: ";

cin >> n;

cout << "1";

for (int i = 1; i <= n; i++)

{

cout << " + " << "1/" << fatorial(i);

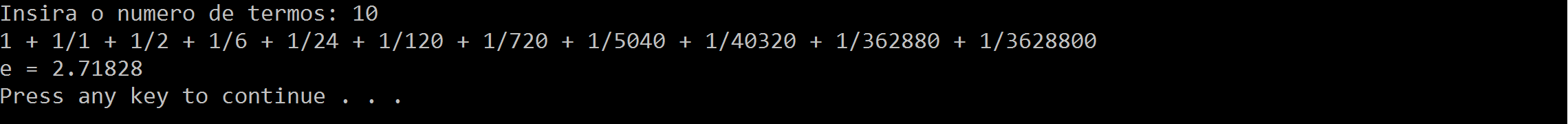
termo += 1 \* 1.0 / fatorial(i);

}

cout << endl << "e = " << termo << endl;

return 0;

}



### 2.11-c)

#include <iostream>

using namespace std;

int fatorial(int n)

{

if (n == 1)

return 1;

else

return n \* fatorial(n - 1);

}

int main(void)

{

int n, x, dividendo;

double termo = 1;

cout << "Insira o numero de termos: ";

cin >> n;

cout << "Insira o valor de x:";

cin >> x;

cout << "1";

dividendo = 1;

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

{

dividendo \*= x;

if (i % 2 == 0)

{

termo -= dividendo \* 1.0 / fatorial(i+1);

cout << " - " << dividendo << "/" << fatorial(i+1);

}

else

{

termo += dividendo \* 1.0 / fatorial(i+1);

cout << " + " << dividendo << "/" << fatorial(i+1);

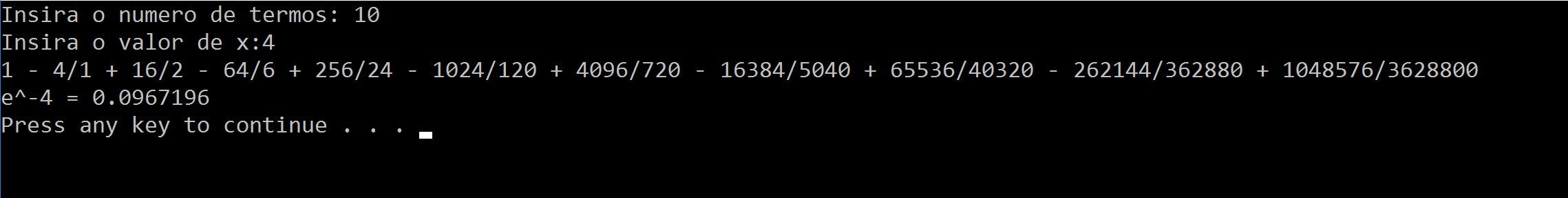
}

}

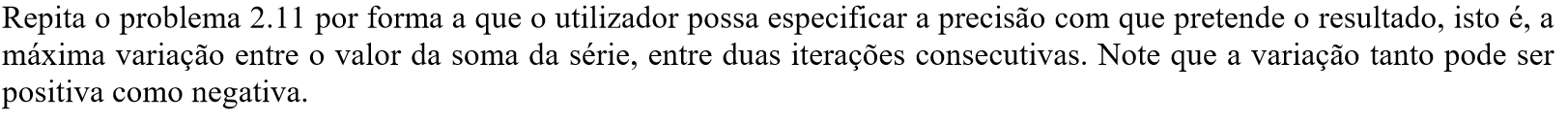
cout << endl << "e^-" << x << " = " << termo << endl;

return 0;

}



## 2.12



### 2.12-a)

#include <iostream>

#include <cmath>

using namespace std;

int fatorial(int n)

{

if (n == 1)

return 1;

else

return n \* fatorial(n - 1);

}

int main(void)

{

int divisor, i;

double termo, termo\_ant, precision;

divisor = 1;

termo = 4;

cout << "Insira a precisão: ";

cin >> precision;

cout << "4";

i = 0;

while(1)

{

divisor += 2;

if (i % 2 == 0)

{

termo\_ant = termo;

termo -= 4 \* 1.0 / divisor;

if (fabs(termo - termo\_ant) > precision)

cout << " - " << "4/" << divisor;

else

{

termo = termo\_ant;

break;

}

}

else

{

termo\_ant = termo;

termo += 4 \* 1.0 / divisor;

if (fabs(termo - termo\_ant) > precision)

cout << " + " << "4/" << divisor;

else

{

termo = termo\_ant;

break;

}

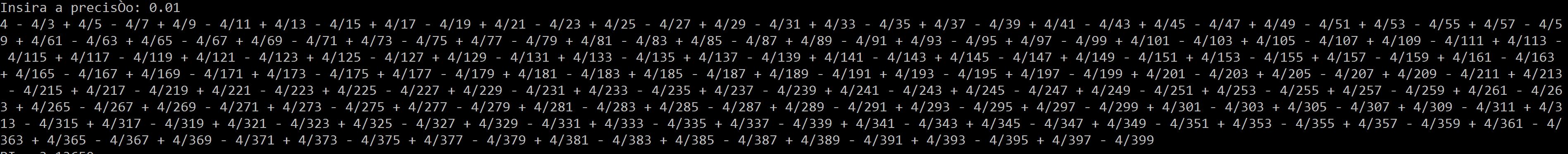
}

i++;

}

cout << endl << "PI = " << termo << endl;

return 0;}



### 2.12-b)

#include <iostream>

#include <cmath>

using namespace std;

int fatorial(int n)

{

if (n == 1)

return 1;

else

return n \* fatorial(n - 1);

}

int main(void)

{

int i = 0;

double termo, termo\_ant, precision;

termo = 1;

termo\_ant = 0;

cout << "Insira a precisao: ";

cin >> precision;

cout << "1";

while(1)

{

if (fabs(termo - termo\_ant) > precision)

{

termo\_ant = termo;

cout << " + " << "1/" << fatorial(i + 1);

termo += 1 \* 1.0 / fatorial(i + 1);

}

else

break;

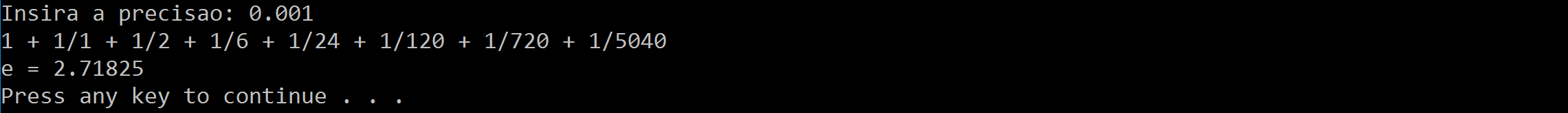
i++;

}

cout << endl << "e = " << termo << endl;

return 0;

}



### 2.12-c)

#include <iostream>

#include <cmath>

using namespace std;

int fatorial(int n)

{

if (n == 1)

return 1;

else

return n \* fatorial(n - 1);

}

int main(void)

{

int x, dividendo, i;

double termo, precision, termo\_ant;

termo = 1;

termo\_ant = 0;

i = 0;

cout << "Insira a precisao: ";

cin >> precision;

cout << "Insira o valor de x:";

cin >> x;

cout << "1";

dividendo = 1;

while(1)

{

dividendo \*= x;

if (i % 2 == 0)

{

termo -= dividendo \* 1.0 / fatorial(i + 1);

if (fabs(termo - termo\_ant) > precision)

{

cout << " - " << dividendo << "/" << fatorial(i + 1);

termo\_ant = termo;

}

else

break;

}

else

{

termo += dividendo \* 1.0 / fatorial(i + 1);

if (fabs(termo - termo\_ant) > precision)

{

cout << " + " << dividendo << "/" << fatorial(i + 1);

termo\_ant = termo;

}

else

break;

}

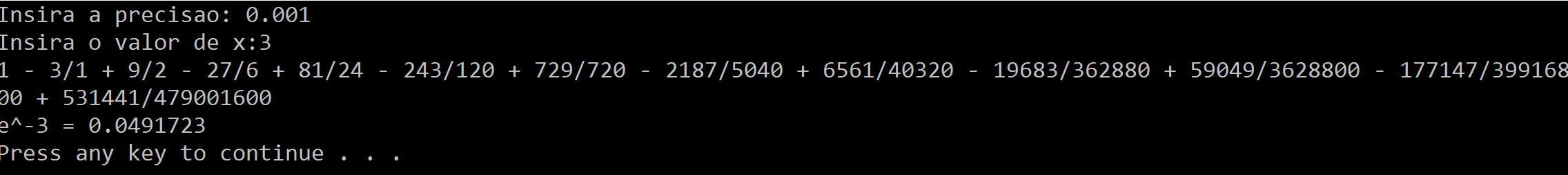
i++;

}

cout << endl << "e^-" << x << " = " << termo << endl;

return 0;

}



## 2.13

#include <iostream>

using namespace std;

int main(void)

{

int num, divisor;

divisor = 2;

cout << "Insira o numero que pretende decompor: ";

cin >> num;

cout << num << " = ";

while (num != 1)

{

if (num % divisor == 0)

{

num /= divisor;

if (num != 1)

cout << divisor << " x ";

else

cout << divisor << endl;

}

else

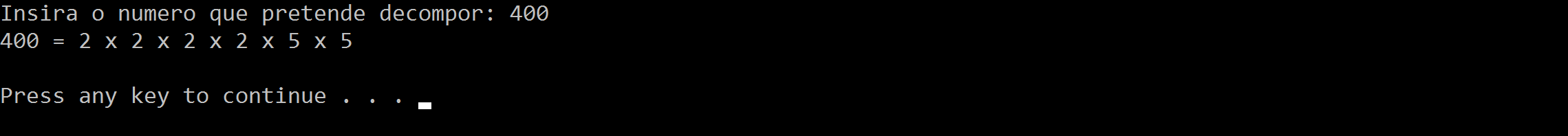
divisor++;

}

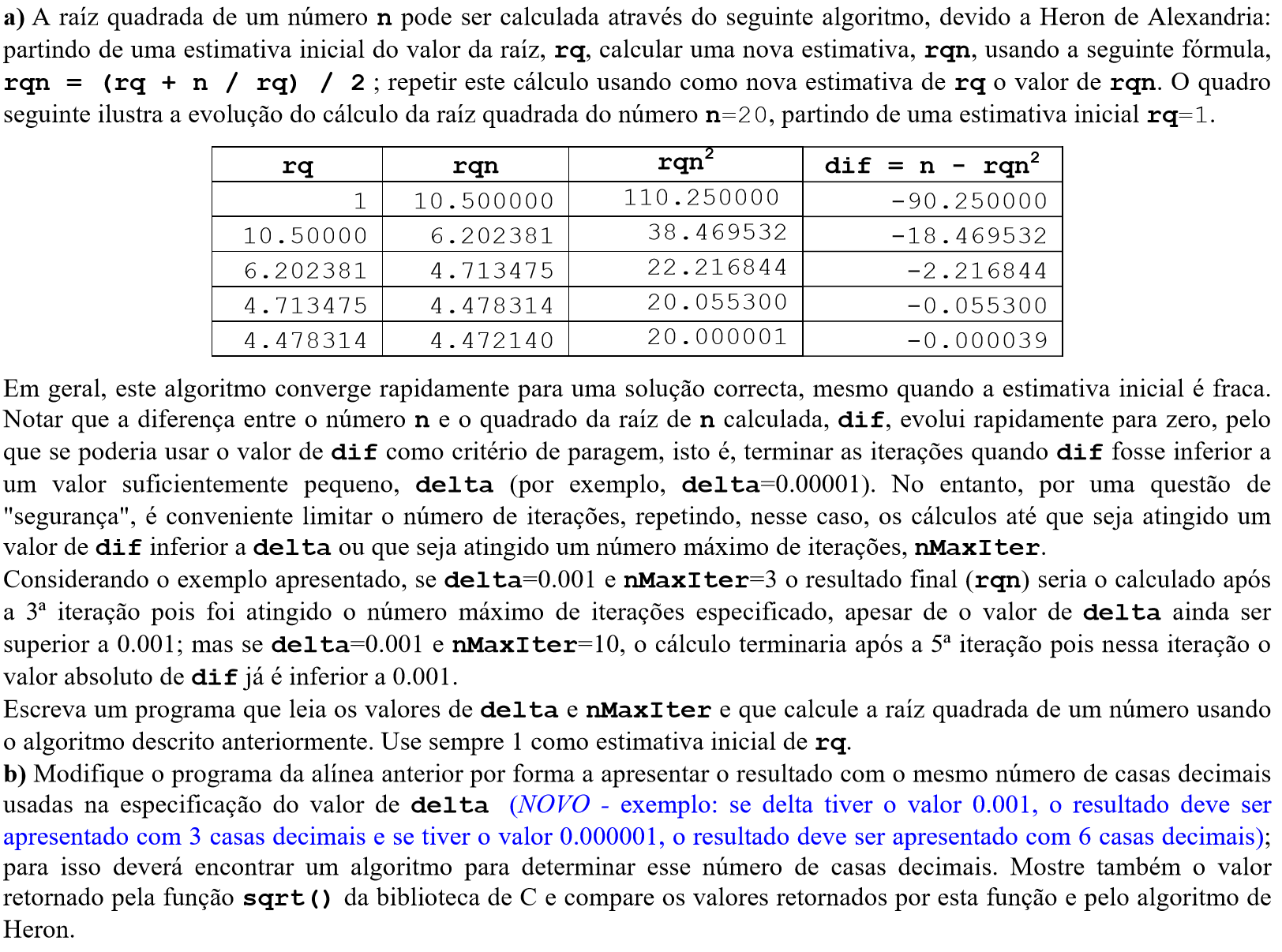
cout << endl;

return 0;

}



## 2.14



### 2.14-a)

#include <iostream>

#include <cmath>

using namespace std;

int main(void)

{

double delta;

float rqn, rq;

int nMaxIter, n, i;

i = 1;

rq = 1;

rqn = 0;

cout << "Insira o delta e o numero maximo de iteracoes: ";

cin >> delta >> nMaxIter;

cout << "Insira o numero de que pretende calcular a raiz: ";

cin >> n;

cout << "iterações\trq\trqn\trqn\*rqn\tDiferenca" << endl;

while (i <= nMaxIter && fabs(n - rqn\*rqn) > delta)

{

if(i != 1)

rq = rqn;

rqn = (rq + n / rq) / 2;

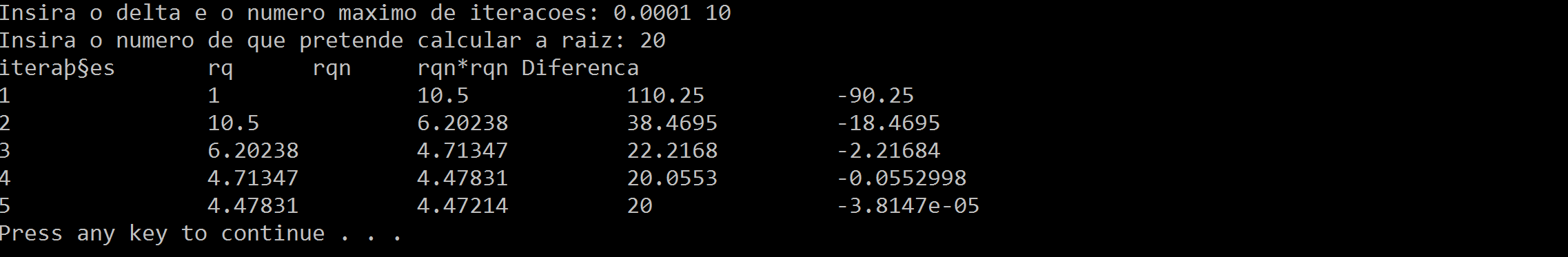
cout << i << " " << rq << " " << rqn << " " << rqn\*rqn << " " << n - rqn\*rqn << endl;

i++;

}

return 0;

}



### 2.14-b)

#include <iostream>

#include <cmath>

#include <string>

#include <iomanip>

using namespace std;

float precisao(string str)

{

int x;

float precisao = 1;

for (int i = 0; i < (int)str.size() - 1; i++)

{

if(str[i] == '.')

{

x = str.size() - 1 - i;

break;

}

}

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

precisao += 1;

return precisao;

}

int main(void)

{

string delta;

float rqn = 0, rq = 1;

int nMaxIter = 10, n, i = 1;

cout << "Insira o numero de que pretende calcular a raiz: ";

cin >> n;

cout << "Insira o delta: ";

cin >> delta;

cout << "iteracoes\trq\t\trqn\t\trqn\*rqn\t\tDiferenca" << endl;

while (i <= nMaxIter && fabs(n - rqn\*rqn) > 1/pow(10,precisao(delta)))

{

if(i != 1)

rq = rqn;

rqn = (rq + n / rq) / 2;

cout << i << fixed << setprecision(precisao(delta) - 1) << " " << rq << " " << rqn << " " << rqn\*rqn << " " << n - rqn\*rqn << endl;

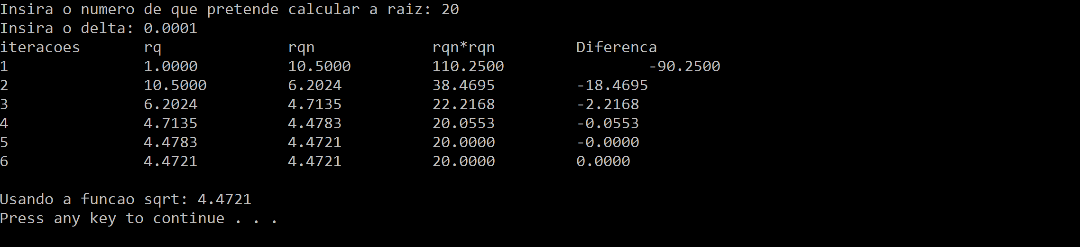
i++;

}

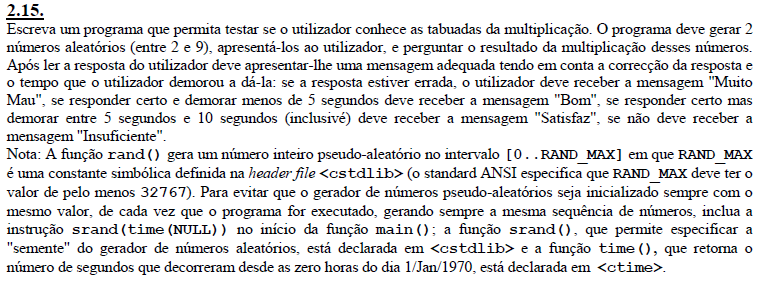
cout << endl << "Usando a funcao sqrt: " << sqrt(n) << endl;

return 0;

}



## 2.15



#include <iostream>

#include <ctime>

#include <cstdlib>

#include <thread> // std::this\_thread::sleep\_for

#include <chrono> // std::chrono::seconds

using namespace std;

int main(void)

{

int a, b, resultado, resposta;

double duracao;

time\_t inicio, fim;

srand(time(NULL)); //inicializar a semente

a = rand() % 9 + 2;

//obtenção dos numeros aleatórios

b = rand() % 9 + 2;

resultado = a \* b;

cout << "Preparado? 3 segundos para comecar!\n";

for (int i = 3; i > 0; --i) {

cout << i << std::endl; //CountDown

this\_thread::sleep\_for(std::chrono::seconds(1));

}

inicio = clock();

cout << a << " \* " << b << " = ";

cin >> resposta;

fim = clock();

duracao = (float)(fim - inicio) / CLOCKS\_PER\_SEC; //Obtencao do tempo

if (resposta != resultado)

cout << "Muito mau! ( " << duracao << " segundos)\n";

else

{

if(duracao < 5)

cout << "BOM! ( " << duracao << " segundos)\n";

else if (duracao >= 5 && duracao <= 10)

cout << "Satisfaz! ( " << duracao << " segundos)\n";

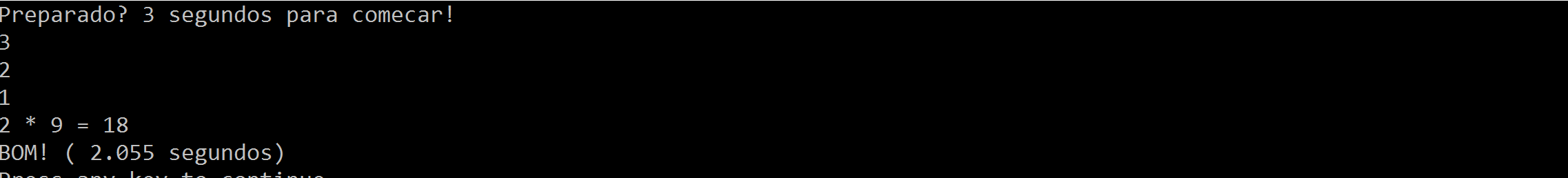
else

cout << "Insuficiente! ( " << duracao << " segundos)\n";

}

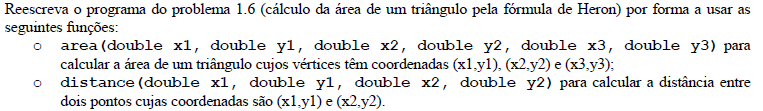
return 0;

}



# **Funções**

## 3.1



#include <iostream>

#include <cmath>

#include <vector>

using namespace std;

float distancia (double x1, double y1, double x2, double y2)

{

return (float) sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2));

}

float area(double x1, double y1, double x2, double y2, double x3, double y3)

{

float ab, ac, bc, s;

ab = distancia(x1, y1, x2, y2);

ac = distancia(x1, y1, x3, y3); /\*Calcular as arestas\*/

bc = distancia(x2, y2, x3, y3);

s = (ab + ac + bc) / 2; /\*Calcular o semi-perímetro\*/

return sqrt(s\*(s - ab)\*(s - ac)\*(s - bc)); /\*Calcular Área (fórmula de Heron)\*/

}

int main(void)

{

vector<int> a(2);

vector<int> b(2);

vector<int> c(2);

cout << "Introduza as coordenadas do ponto A:";

cin >> a[0] >> a[1];

cout << "Introduza as coordenadas do ponto B:";

cin >> b[0] >> b[1];

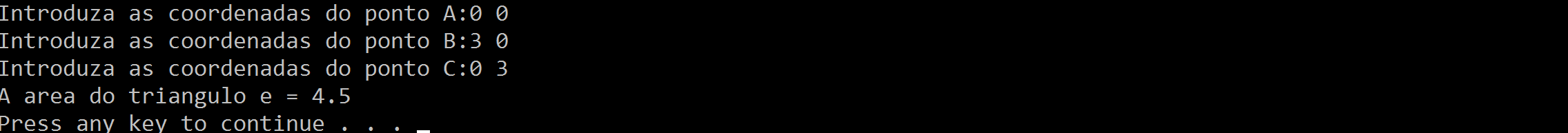
cout << "Introduza as coordenadas do ponto C:";

cin >> c[0] >> c[1];

cout << "A area do triangulo e = " << area(a[0], a[1], b[0], b[1], c[0], c[1]) << endl;

return 0;

}

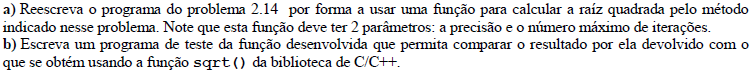


## 3.2



Já foi feito no 2.6 #MyBad

## 3.3



### 3.3-a)

### 3.3-b)

#include <iostream>

#include <cmath>

#include <iomanip>

#include <string>

using namespace std;

int precisao(string str)

{

int x;

for (int i = 0; i < (int)str.size() - 1; i++)

{

if (str[i] == '.')

{

x = str.size() - i;

break;

}

}

return x;

}

float raiz(int raiz, int delt, int max)

{

float rqn, rq;

int i;

i = 1;

rq = 1;

rqn = 0;

while (i <= max && fabs(raiz - rqn\*rqn) > pow(1/10,delt))

{

if (i != 1)

rq = rqn;

rqn = (rq + raiz / rq) / 2;

cout << fixed << setprecision(delt) << i << " " << rq << " " << rqn << " " << rqn\*rqn << " " << raiz - rqn\*rqn << endl;

i++;

}

return rqn;

}

int main(void)

{

double erro;

int maxi, n;

float resultado;

string delta;

cout << "Insira o delta e o numero maximo de iteracoes: ";

cin >> delta >> maxi;

cout << "Insira o numero de que pretende calcular a raiz: ";

cin >> n;

resultado = raiz(n, precisao(delta), maxi);

erro = (double)(abs(resultado - sqrt(n)));

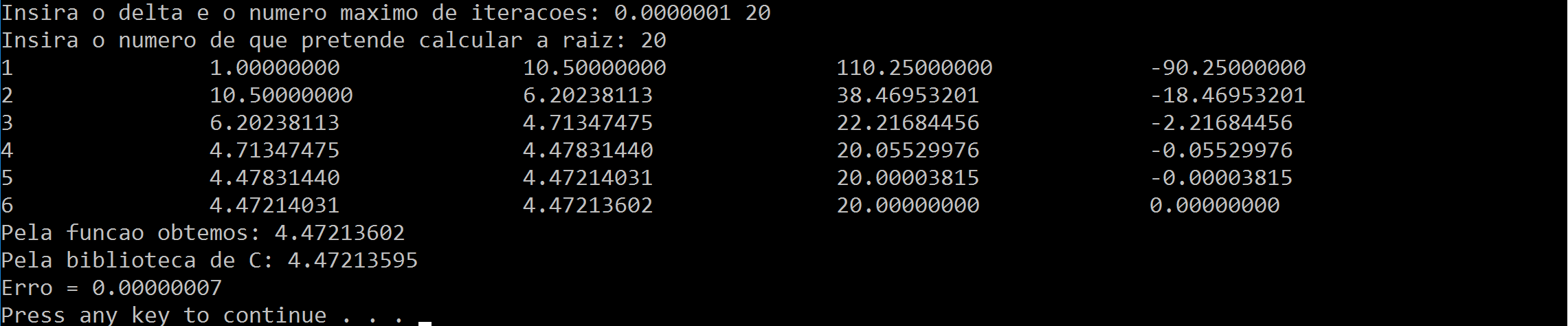
cout << fixed << setprecision(precisao(delta)) << "Pela funcao obtemos: " << resultado << endl;

cout << fixed << setprecision(precisao(delta)) << "Pela biblioteca de C: " << sqrt(n) << endl;

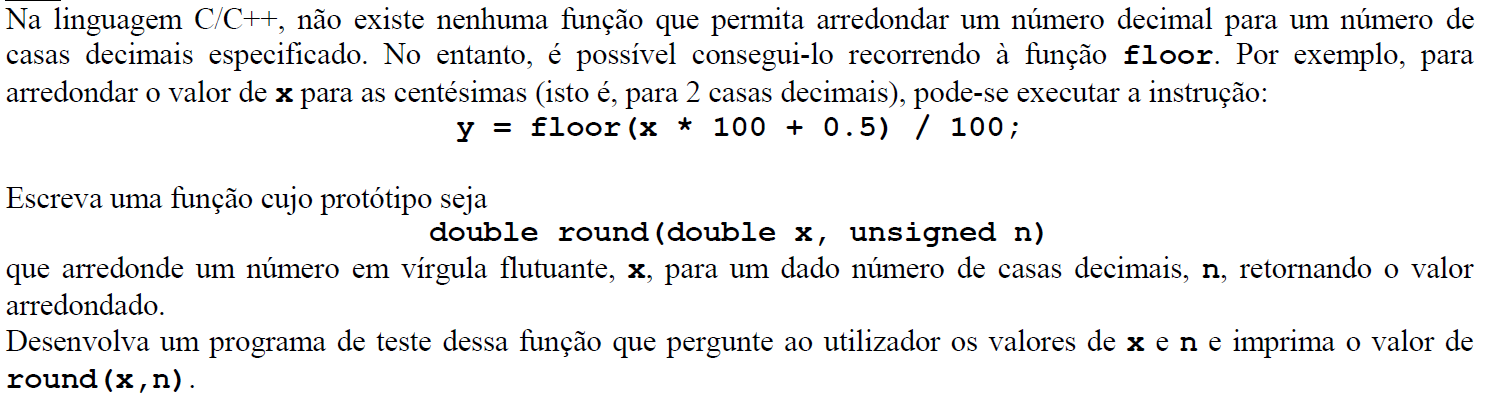
cout << fixed << setprecision(precisao(delta)) << "Erro = " << erro << endl;

return 0;

}



## 3.4



#include <iostream>

#include <cmath>

#include <iomanip>

#include <string>

using namespace std;

double round(double x, unsigned n)

{

double final;

long long int aux;

aux =(long long int) x \* pow(10, n); /\*Long long int para nao dar overflow\*/

final = (long long int)(x \* pow(10, n));

final = 1.0 \* final / pow(10, n);

if (abs(aux - x \* pow(10, n)) >= 1.5)

final += 1 / pow(10, n);

return final;

}

int main(void)

{

double input;

int casas;

cout << "Insira o numero que pretende arredondar e o numero de casas decimais: ";

cin >> input >> casas;

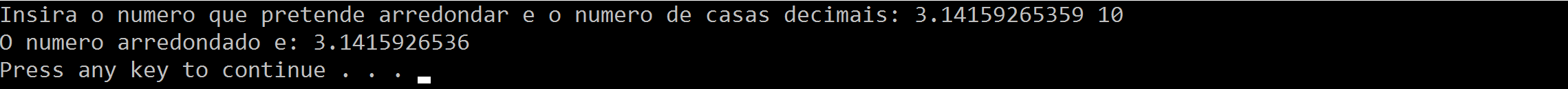
input = round(input, casas);

cout << "O numero arredondado e: ";

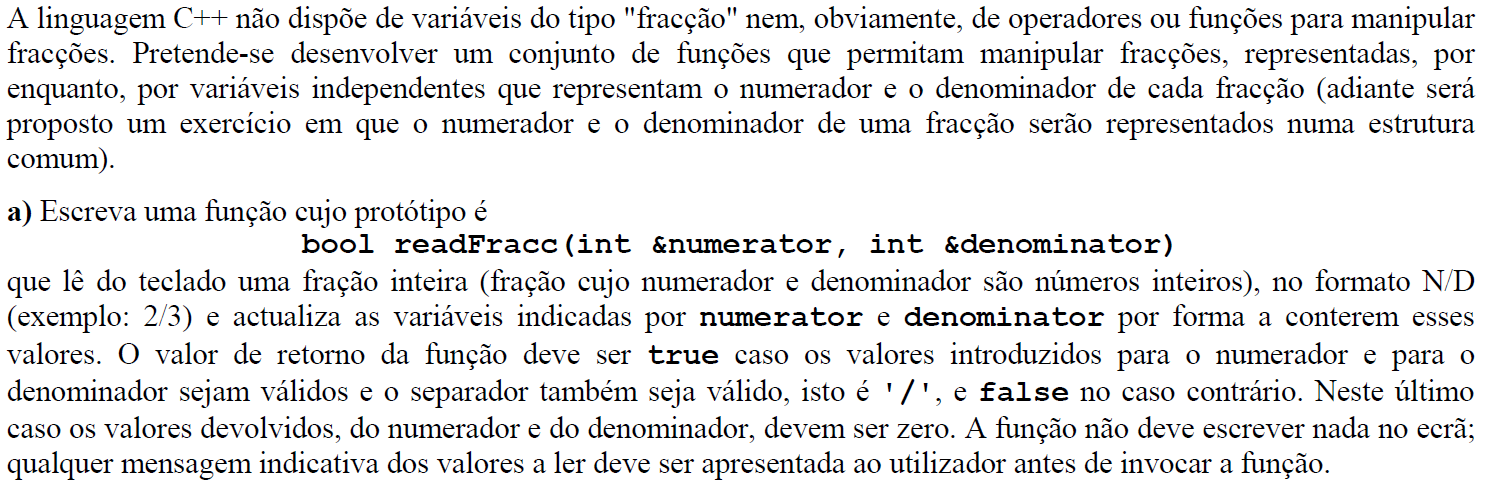
cout << fixed << setprecision(casas) << input << endl;

return 0;

}



## 3.5



### 3.5-a)

#include <iostream>

#include <string>

using namespace std;

bool readFracc(int &numerator, int &denominator)

{

string input;

size\_t pos;

cout << "Introduza a fraccao ( N/D ): ";

cin >> input;

pos = input.find("/");

if (pos == string::npos)

return false;

numerator = stoi(input.substr(0, pos));

denominator = stoi(input.substr(pos + 1));

return true;

}

int main(void)

{

int numerador, denominador;

if (readFracc(numerador, denominador))

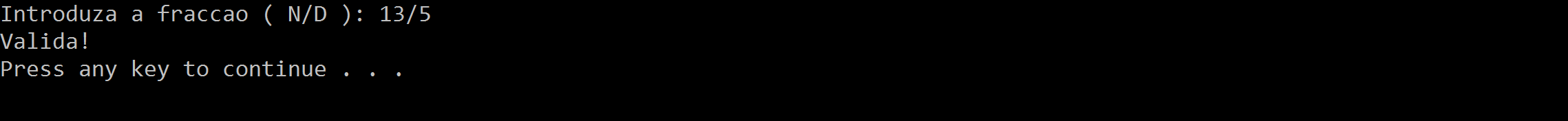
cout << "Valida!\n";

else

cout << "Nao valida!\n";

return 0;

}



### 3.5-b)

#include <iostream>

#include <string>

using namespace std;

bool readFracc(int &numerator, int &denominator)

{

string input;

size\_t pos;

cout << "Introduza a fraccao ( N/D ): ";

cin >> input;

pos = input.find("/");

if (pos == string::npos)

return false;

numerator = stoi(input.substr(0, pos));

denominator = stoi(input.substr(pos + 1));

return true;

}

void writeFracc(int numerator, int denominator)

{

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

}

int main(void)

{

int numerador, denominador;

if (readFracc(numerador, denominador))

cout << "Valida!\n";

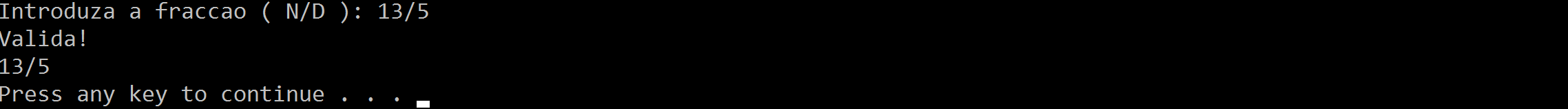
else

cout << "Nao valida!\n";

writeFracc(numerador, denominador);

return 0;

}



### 3.5-c)

#include <iostream>

#include <string>

using namespace std;

bool readFracc(int &numerator, int &denominator)

{

… [/\*3.5-a)\*/](#_3.5-a))

}

void writeFracc(int numerator, int denominator)

{

…[/\*3.5-b)\*/](#_3.5-b))

}

void reduceFracc(int &numerator, int &denominator)

{

int maior = 1;

if (numerator < denominator)

{

for (int i = 1; i <= numerator; i++)

{

if (numerator % i == 0 && denominator % i == 0)

maior = i;

}

numerator /= maior;

denominator /= maior;

}

else if(numerator > denominator)

{

for (int i = 1; i <= denominator; i++)

{

if (numerator % i == 0 && denominator % i == 0)

maior = i;

}

numerator /= maior;

denominator /= maior;

}

else

{

numerator /= numerator;

denominator /= denominator;

}

}

int main(void)

{

int numerador, denominador;

if (readFracc(numerador, denominador))

cout << "Valida!\n";

else

cout << "Nao valida!\n";

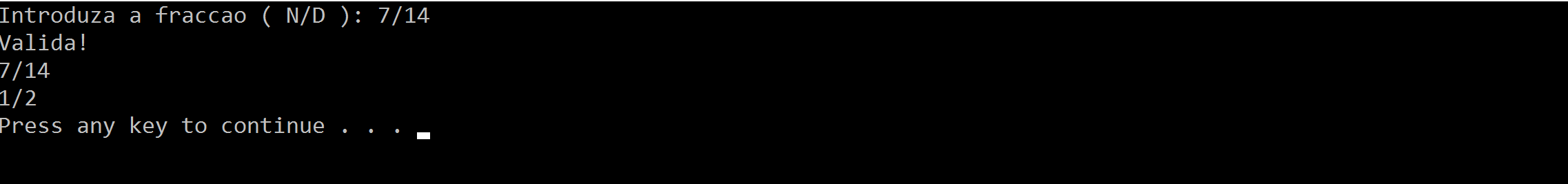
writeFracc(numerador, denominador);

reduceFracc(numerador, denominador);

writeFracc(numerador, denominador);

return 0;

}



### 3.5-d)

/\*Funções para soma e multiplicação\*/

void somaFracc(int &numeratorA, int &denominatorA, int &numeratorB, int &denominatorB, int &numeres, int &denores)

{

//Primeiro -> Multiplicar os numeradores pelos denominadores para ter fraccoes equivalentes

numeres = numeratorA \* denominatorB + numeratorB \* denominatorA;

//Segundo-> Meter as fraccoes com o mesmo denominador

denores = denominatorB \* denominatorA;

}

void multFracc(int &numeratorA, int &denominatorA, int &numeratorB, int &denominatorB, int &numeres, int &denores)

{

numeres = numeratorA \* numeratorB;

denores = denominatorA \* denominatorB;

}

/\*subtrair é adicionar a segunda fração multiplicada por -1 e dividir é multiplicar pelo inverso\*/

### 3.5-e)

#include <iostream>

#include <string>

#include <vector>

using namespace std;

bool readFracc(int &numerator, int &denominator)

{

…

}

void reduceFracc(int &numerator, int &denominator)

{

…

}

void somaFracc(int &numeratorA, int &denominatorA, int &numeratorB, int &denominatorB, int &numeres, int &denores)

{

…

}

void multFracc(int &numeratorA, int &denominatorA, int &numeratorB, int &denominatorB, int &numeres, int &denores)

{

…

}

int main(void)

{

vector<int> fracc1(2);

vector<int> fracc2(2);

vector<int> fracc3(2);

char op;

if (readFracc(fracc1[0], fracc1[1]) && readFracc(fracc2[0], fracc2[1]))

cout << "Fracoes validas!\n";

else

cout << "Uma das fracoes nao e valida!\n";

for (int i = 0; i < 4; i++)

{

cout << "Indique a operaçao que pretende fazer (+ - \* /): ";

cin >> op;

switch (op)

{

case '+':

somaFracc(fracc1[0], fracc1[1], fracc2[0], fracc2[1], fracc3[0], fracc3[1]);

writeFracc(fracc1[0], fracc1[1]);

cout << " + ";

writeFracc(fracc2[0], fracc2[1]);

cout << " = ";

writeFracc(fracc3[0], fracc3[1]);

cout << endl;

break;

case '-':

fracc2[0] \*= -1;

somaFracc(fracc1[0], fracc1[1], fracc2[0], fracc2[1], fracc3[0], fracc3[1]);

fracc2[0] \*= -1; /\*Volar a ter o valor original\*/

writeFracc(fracc1[0], fracc1[1]);

cout << " - ";

writeFracc(fracc2[0], fracc2[1]);

cout << " = ";

writeFracc(fracc3[0], fracc3[1]);

cout << endl;

break;

case '\*':

multFracc(fracc1[0], fracc1[1], fracc2[0], fracc2[1], fracc3[0], fracc3[1]);

writeFracc(fracc1[0], fracc1[1]);

cout << " \* ";

writeFracc(fracc2[0], fracc2[1]);

cout << " = ";

writeFracc(fracc3[0], fracc3[1]);

cout << endl;

break;

case '/':

multFracc(fracc1[0], fracc1[1], fracc2[1], fracc2[0], fracc3[0], fracc3[1]); /\*dividir = a multiplicar pelo inverso, logo é so trocar os parametros da segunda fracao \*/

writeFracc(fracc1[0], fracc1[1]);

cout << " / ";

writeFracc(fracc2[0], fracc2[1]);

cout << " = ";

writeFracc(fracc3[0], fracc3[1]);

cout << endl;

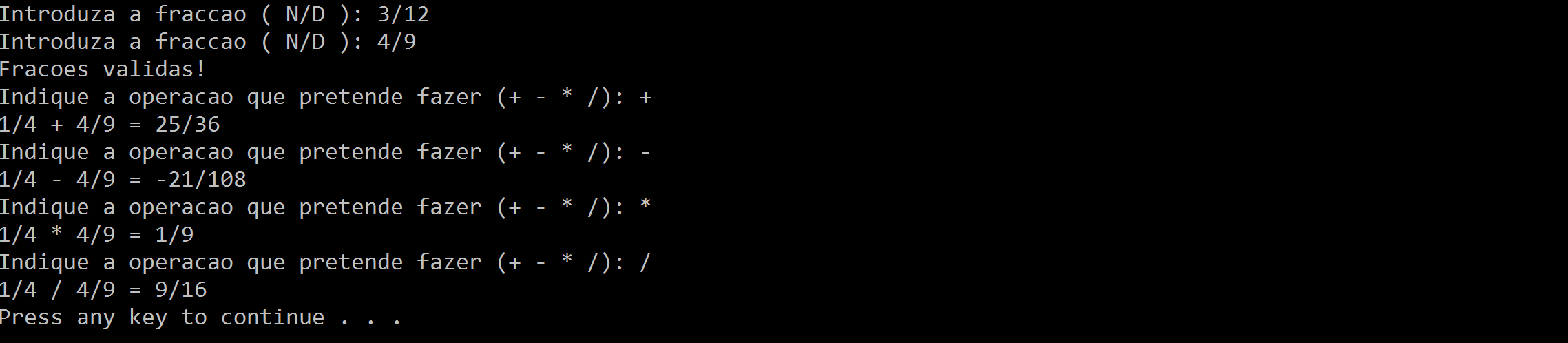
break;

}

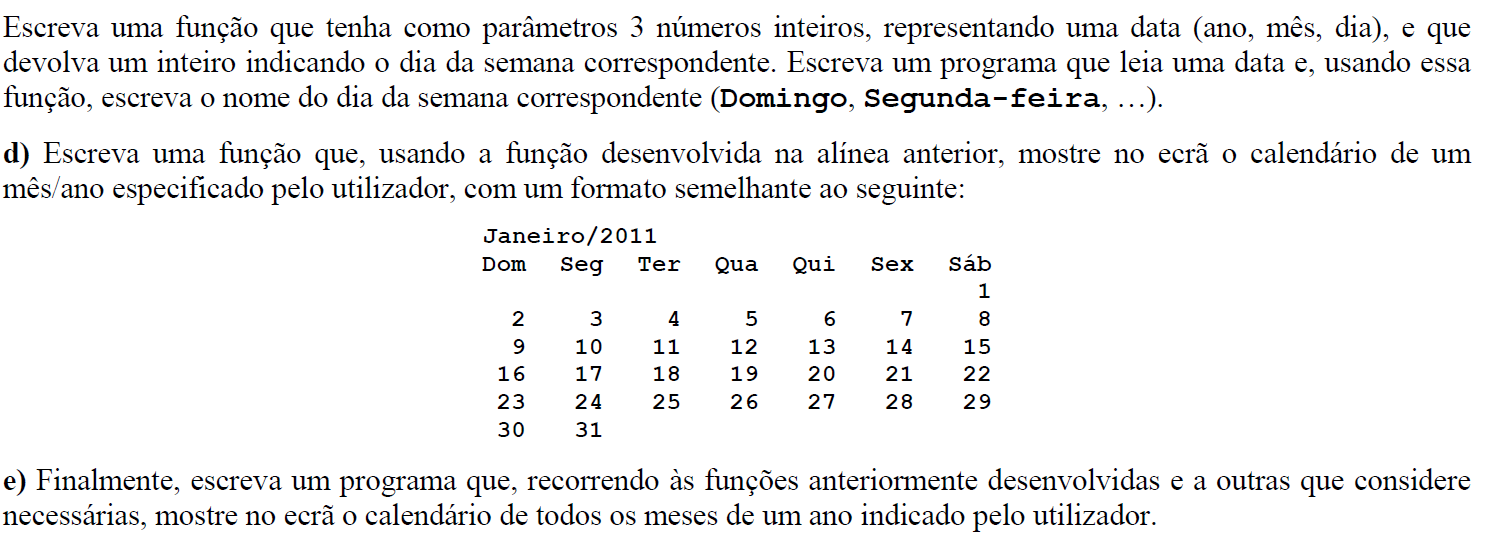
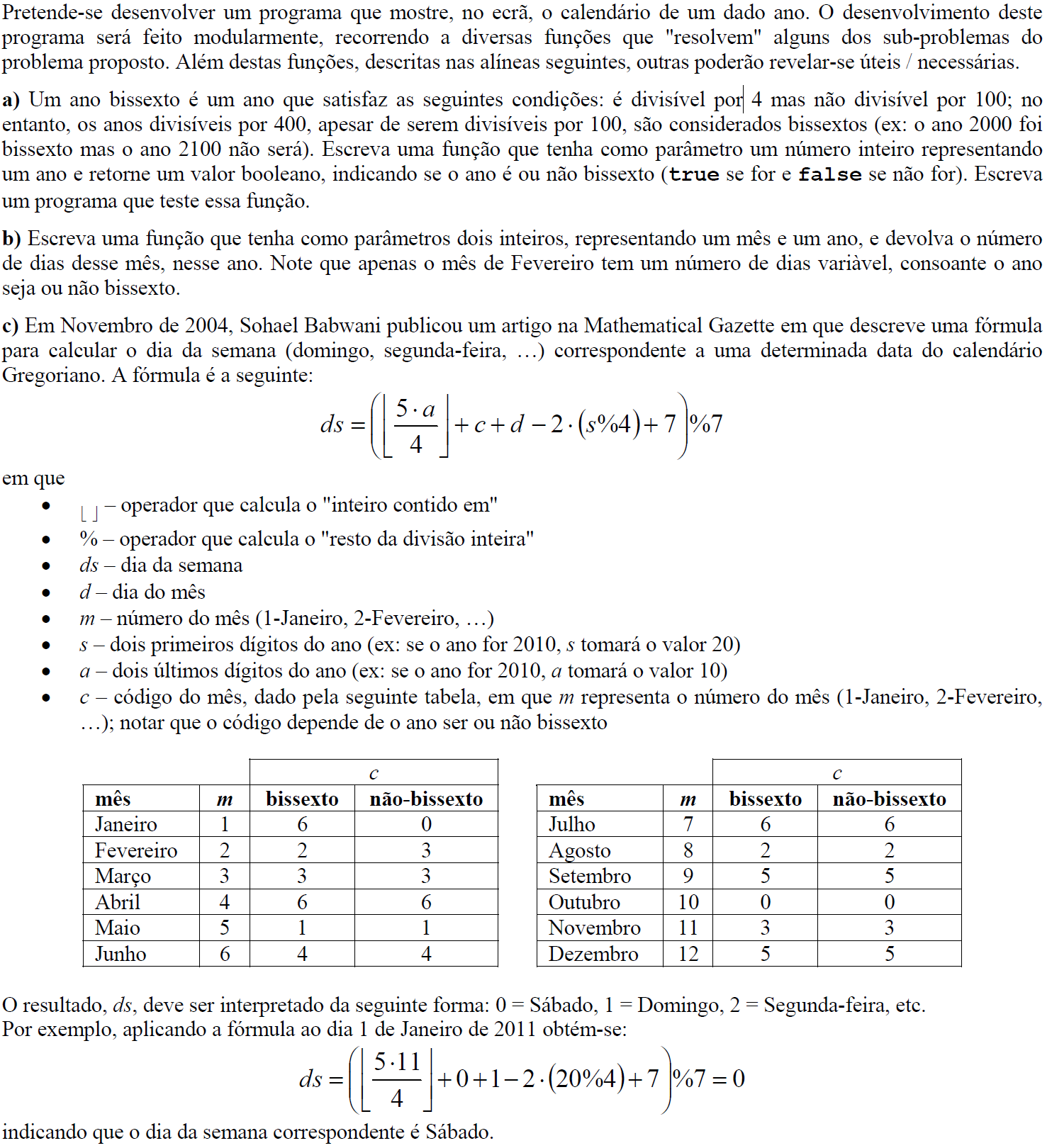
}

return 0;

}



## 3.6



### 3.6-a)

#include <iostream>

using namespace std;

bool isBisexto(int x)

{

if (x % 4 == 0)

if (x % 100 != 0 || x % 400 == 0)

return true;

else

return false;

else

return false;

}

int main(void)

{

int n;

for (int i = 0; i < 2; i++)

{

cin >> n;

if (isBisexto(n))

cout << "Bisexto\n";

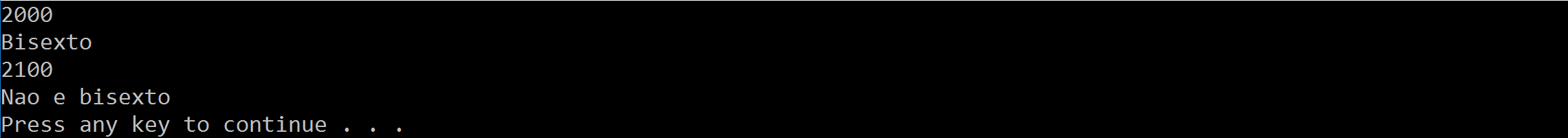
else

cout << "Nao e bisexto\n";

}

return 0;

}



### 3.6-b)

#include <iostream>

using namespace std;

bool isBisexto(int x)

{

…

}

int number\_of\_days(int ano, int mes)

{

switch (mes)

{

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

return 31;

case 6:

case 4:

case 9:

case 11:

return 30;

case 2:

if (isBisexto(ano) == true)

return 29;

else

return 28;

default:

return 0;

}

}

int main(void)

{

int n;

cout << "Insira um ano: ";

cin >> n;

if (isBisexto(n))

cout << "Bisexto\n";

else

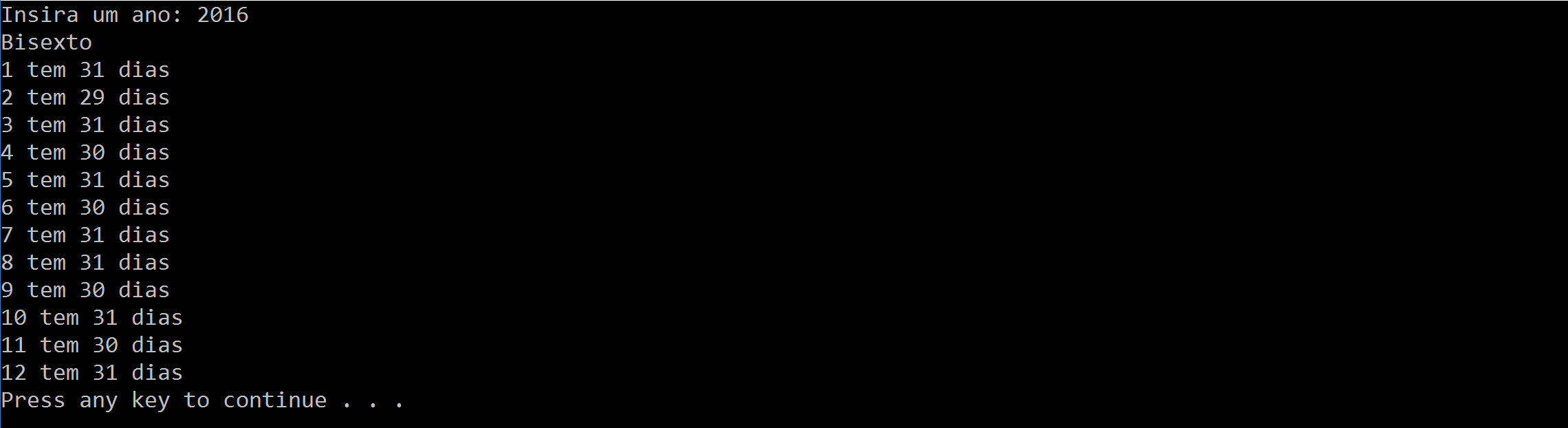
cout << "Nao e bisexto\n";

for (int i = 1; i <= 12; i++)

cout << i << " tem " << number\_of\_days(n, i) << " dias\n";

return 0;

}



### 3.6-c)

#include <iostream>

#include <string>

using namespace std;

bool isBisexto(int x)

{

…

}

int number\_of\_days(int ano, int mes)

{

…

}

int month\_code(int ano, int mes)

{

if (mes < 1 || mes > 12)

return -1;

int bi[12] = { 6,2,3,6,1,4,6,2,5,0,3,5 };

int nor[12] = { 0,3,3,6,1,4,6,2,5,0,3,5 };

if (isBisexto(ano))

return bi[mes - 1];

else

return nor[mes - 1];

}

int day\_of\_the\_week(int ano, int mes, int dia)

{

int a, c, s;

a = ano % 100;

s = ano / 100;

c = month\_code(ano, mes);

return ((int)(5 \* a / 4) + c + dia - 2 \* (s % 4) + 7) % 7;

}

string day(int x)

{

switch (x)

{

case 0:

return "Sabado";

case 1:

return "Domingo";

case 2:

return "Segunda-feira";

case 3:

return "Terca-feira";

case 4:

return "Quarta-feira";

case 5:

return "Quinta-feira";

case 6:

return "Sexta-feira";

default:

break;

}

}

int main(void)

{

int ano, dia, mes;

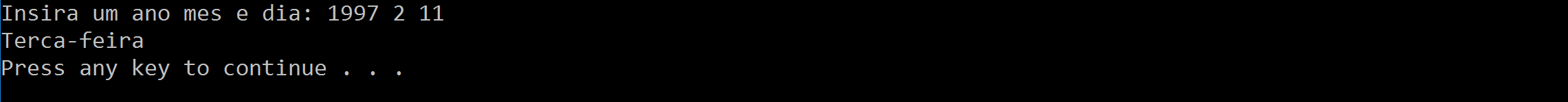
cout << "Insira um ano mes e dia: ";

cin >> ano >> mes >> dia;

cout << day(day\_of\_the\_week(ano, mes, dia)) << endl;

return 0;

}



### 3.6-d)

#include <iostream>

#include <string>

#include <cstring>

using namespace std;

bool isBisexto(int x)

{

…

}

int number\_of\_days(int ano, int mes)

{

…

}

int month\_code(int ano, int mes)

{

…

}

int day\_of\_the\_week(int ano, int mes, int dia)

{

…

}

string day(int x)

{

…

}

string month(int x)

{

string meses[] = { "Janeiro", "Fevereiro", "Marco", "Abril", "Maio", "Junho", "Julho","Agosto","Setembro","Outubro","Novembro","Dezembro" };

return meses[x - 1];

}

int string\_pos(int x, int n\_digits)

{

if (n\_digits == 1)

{

switch (x)

{

case 0:

return 38;

case 1:

return 2;

case 2:

return 8;

case 3:

return 14;

case 4:

return 20;

case 5:

return 26;

case 6:

return 32;

default:

break;

}

}

else

{

switch (x)

{

case 0:

return 37;

case 1:

return 1;

case 2:

return 7;

case 3:

return 13;

case 4:

return 19;

case 5:

return 25;

case 6:

return 31;

default:

break;

}

}

return -1;

}

void print\_month(int ano, int mes)

{

int dias = number\_of\_days(ano, mes);

string linha;

linha.resize(39);

cout << endl << month(mes) << "/" << ano << endl;

cout << endl << "Dom Seg Ter Qua Qui Sex Sab" << endl;

for (int i = 1; i <= dias; i++)

{

if (i / 10 == 0) //numeros de um digito

{

linha.insert(string\_pos(day\_of\_the\_week(ano, mes,i), 1), to\_string(i));

if (string\_pos(day\_of\_the\_week(ano, mes, i), 1) == 38 || i == dias)

{

cout << linha << endl;

linha.clear();

linha.resize(39);

}

else

linha.insert(string\_pos(day\_of\_the\_week(ano, mes, i), 1) + 1, " ");

}

else

{

linha.insert(string\_pos(day\_of\_the\_week(ano, mes, i), 2), to\_string(i));

if (string\_pos(day\_of\_the\_week(ano, mes, i), 2) == 37 || i == dias)

{

cout << linha << endl;

linha.clear();

linha.resize(39);

}

else

linha.insert(string\_pos(day\_of\_the\_week(ano, mes, i), 2) + 2, " ");

}

}

cout << endl;

}

int main(void)

{

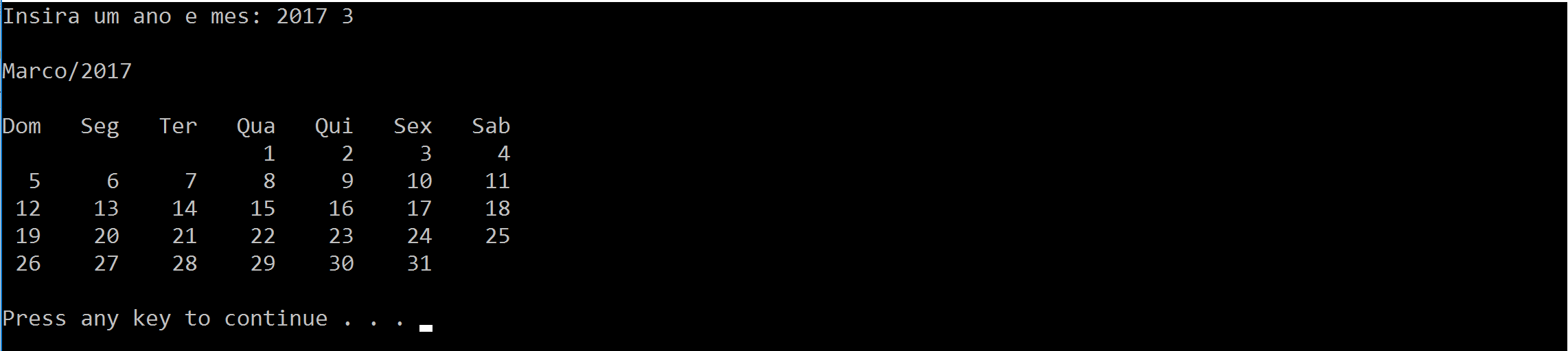
int ano, mes;

cout << "Insira um ano e mes: ";

cin >> ano >> mes ;

print\_month(ano, mes);

return 0;}



### 3.6-e)

#include <iostream>

#include <string>

#include <cstring>

using namespace std;

bool isBisexto(int x)

{

…

}

int number\_of\_days(int ano, int mes)

{

…

}

int month\_code(int ano, int mes)

{

…

}

int day\_of\_the\_week(int ano, int mes, int dia)

{

…

}

string day(int x)

{

…

}

string month(int x)

{

…

}

int string\_pos(int x, int n\_digits)

{

…

}

void print\_month(int ano, int mes)

{

…

}

int main(void)

{

int ano;

cout << "Insira um ano: ";

cin >> ano;

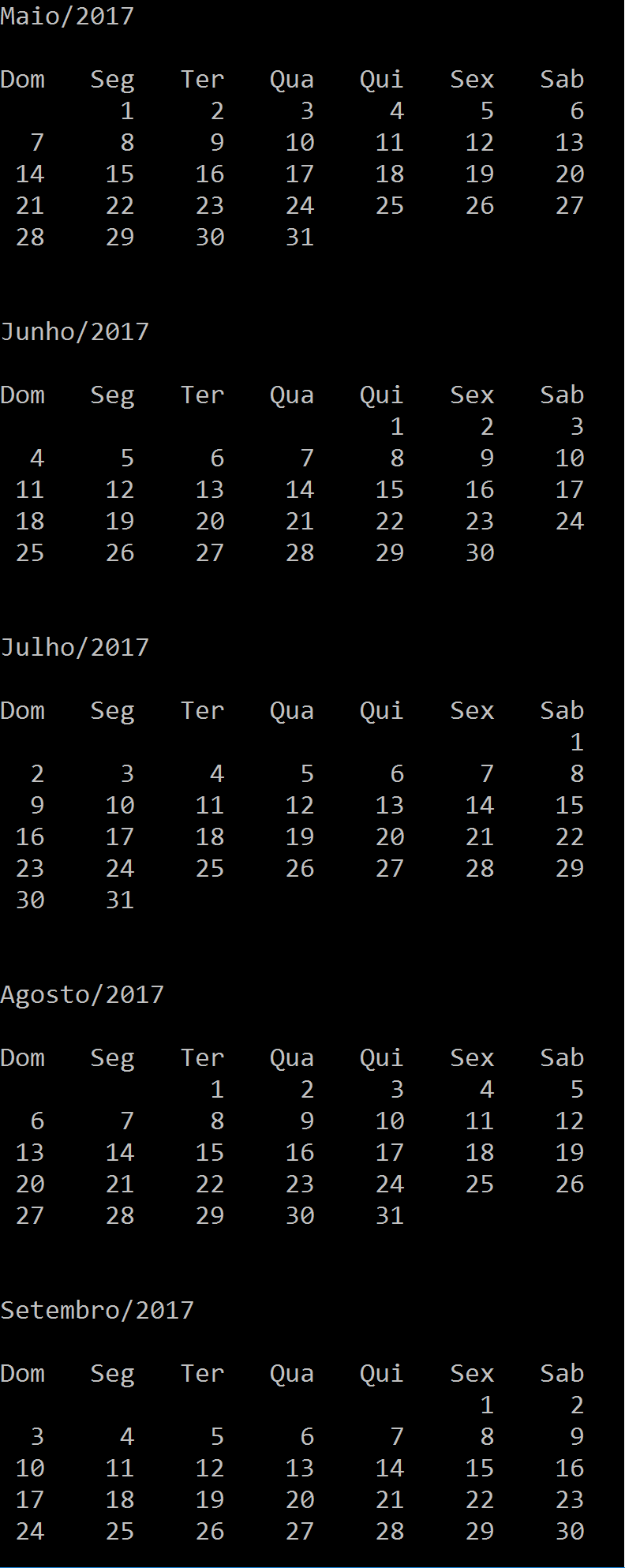
for(int i = 1; i <= 12; i++)

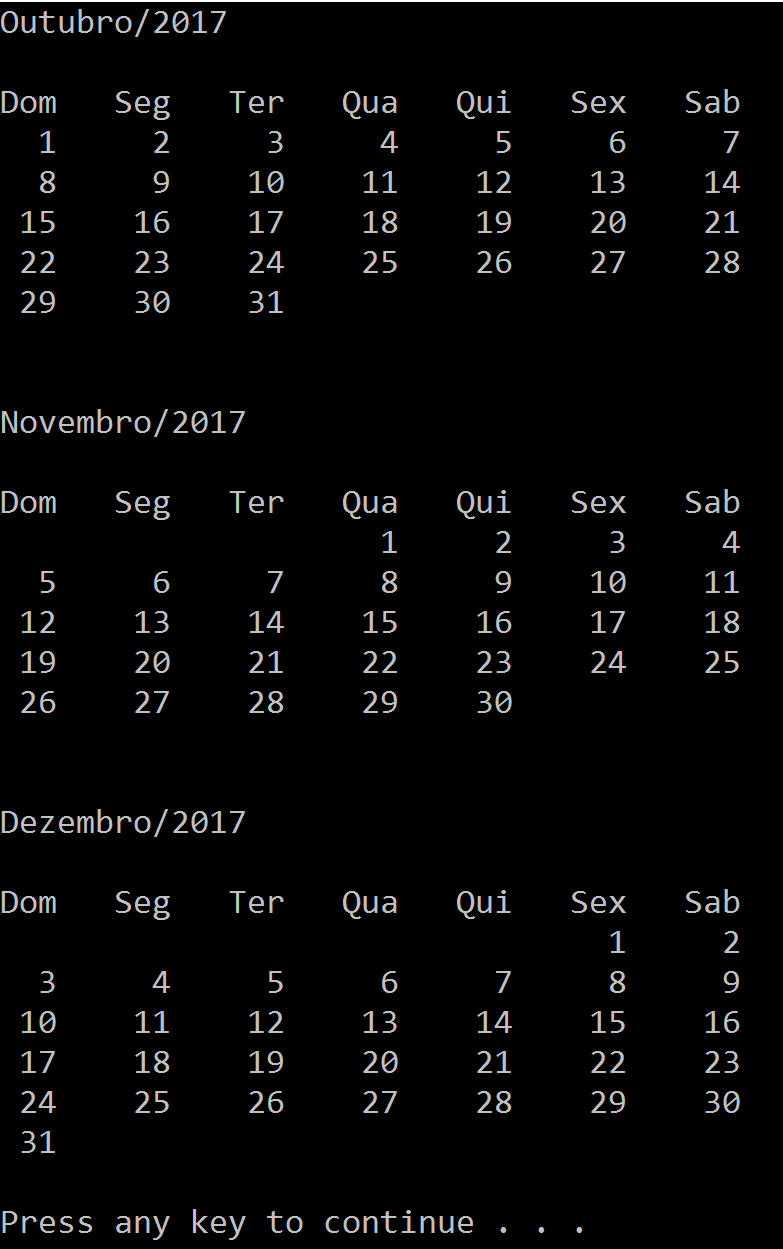
print\_month(ano, i);

return 0;

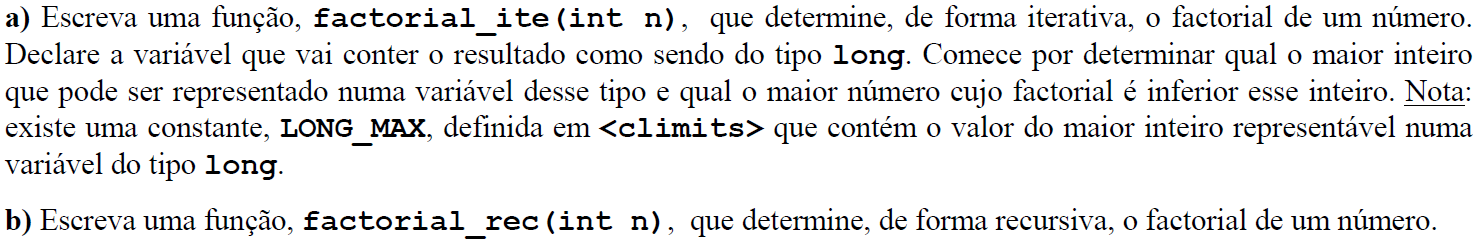
}







## 3.7



#include <iostream>

#include <climits>

using namespace std;

long alinea\_a\_factorial\_ite (int n)

{

int i;

long fatorial = 1;

for (i = n; i > 0; i--)

fatorial \*= i;

if(fatorial <= LONG\_MAX)

return fatorial;

else

return -1;

}

long alinea\_b\_fatorial\_rec(int n)

{

if (n == 1)

return 1;

else

return n \* alinea\_b\_fatorial\_rec(n - 1);

}

int main(void)

{

long fatorial;

cout << "Insira n:";

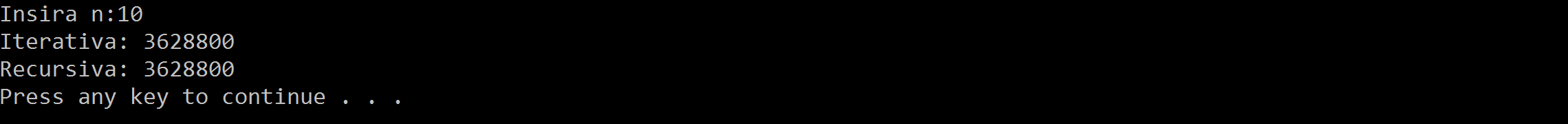
cin >> fatorial;

cout << "Iterativa: " << alinea\_a\_factorial\_ite(fatorial) << endl;

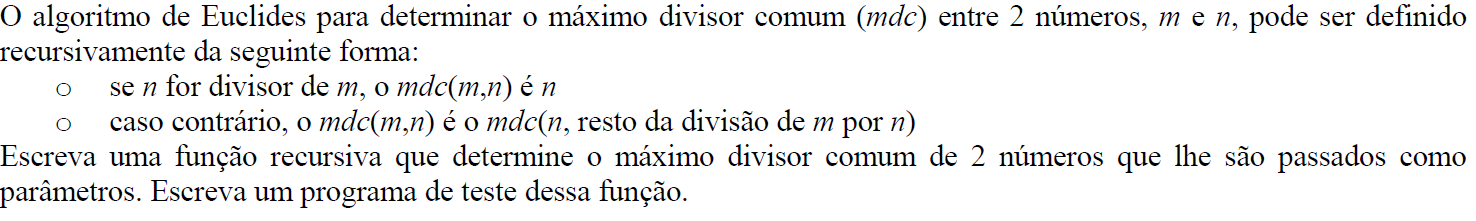
cout << "Recursiva: " << alinea\_b\_fatorial\_rec(fatorial) << endl;

return 0;

}



## 3.8



#include <iostream>

using namespace std;

int mdc (int m, int n)

{

if (m % n == 0)

return n;

else

return mdc(n, m % n);

}

int main(void)

{

int m, n;

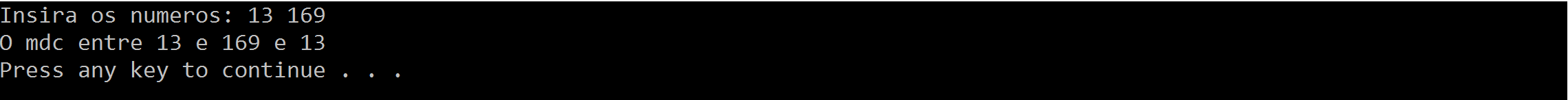
cout << "Insira os numeros: ";

cin >> m >> n;

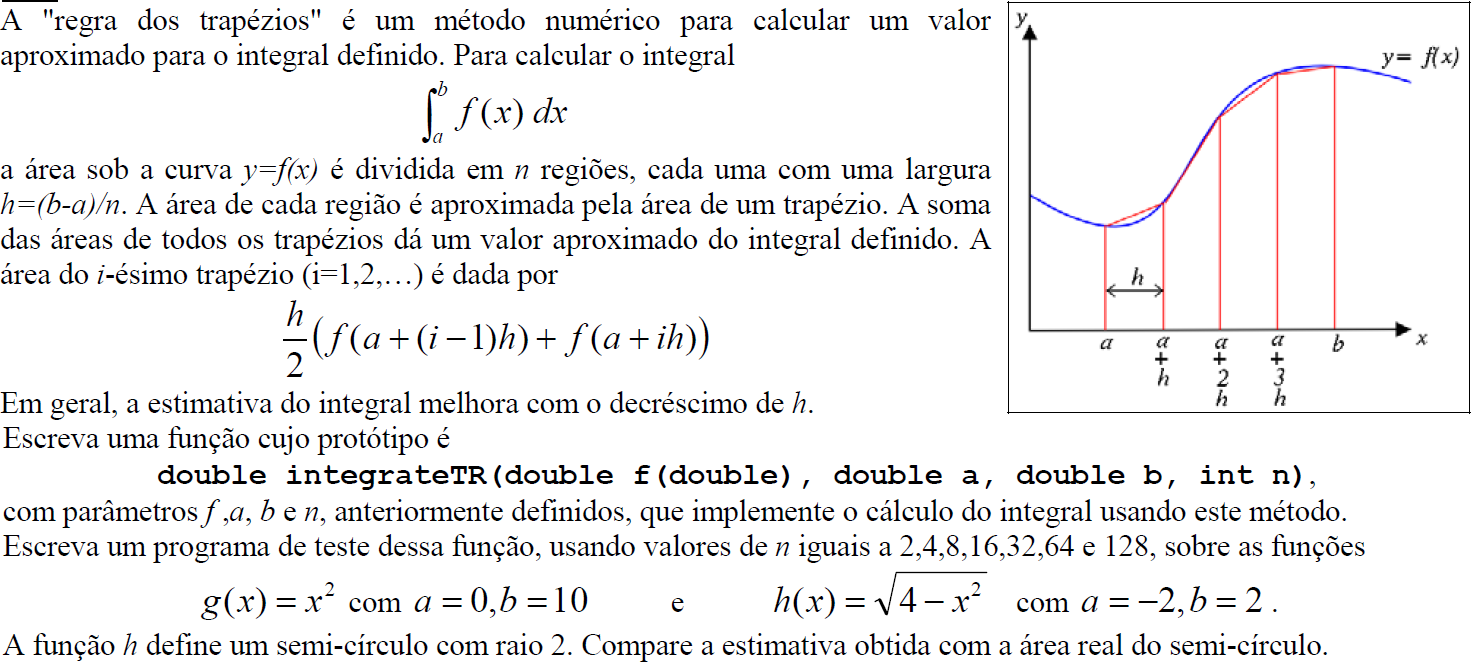
cout << "O mdc entre " << m << " e " << n << " e " << mdc(m, n) << endl;

return 0;

}



## 3.9



#include <iostream>

#include <cmath>

using namespace std;

double g(double x)

{

return x\*x;

}

double h(double x)

{

return sqrt(4 - x\*x);

}

double integrateTR(double f(double), double a, double b, int n)

{

double h, i;

double integral = 0;

h = (b - a) / n;

i = a + h;

while(i < b)

{

integral += 2 \* f(i);

i = i + h;

}

return (f(a) + integral + f(b)) \* h / 2;

}

int main(void)

{

for (int i = 2; i < 256; i \*= 2)

{

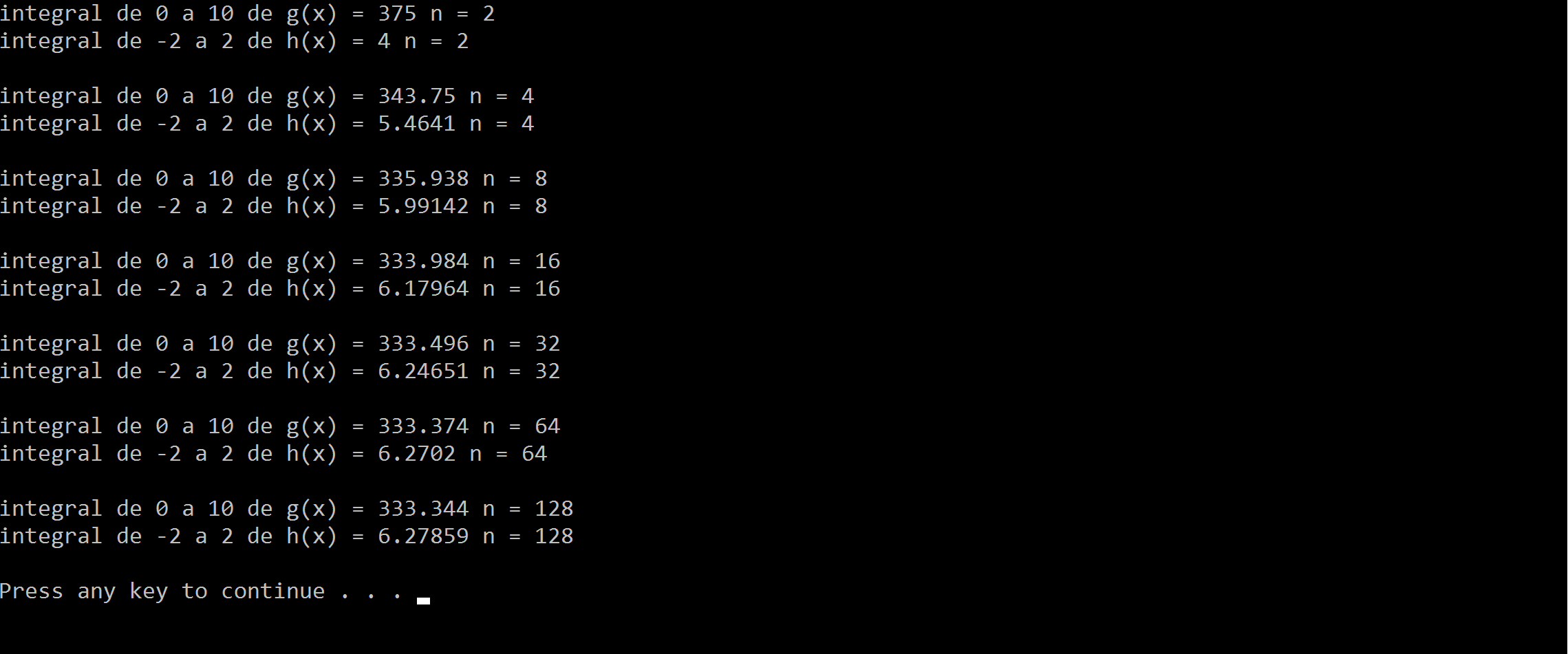
cout << "integral de 0 a 10 de g(x) = " << integrateTR(g, 0, 10, i) << " n = " << i << endl;

cout << "integral de -2 a 2 de h(x) = " << integrateTR(h, -2, 2, i) << " n = " << i << endl << endl;

}

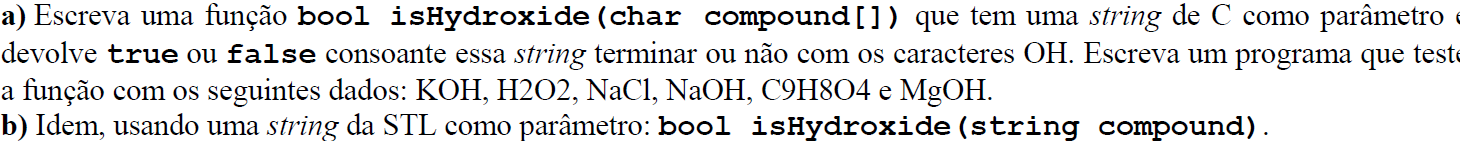
return 0;

}



# ***Arrays, Vectors, Strings*, *Structs* e *Files***

## 4.1



### 4.1-a)

#include <iostream>

#include <string.h>

using namespace std;

bool isHydroxide(char compound[])

{

if (compound[strlen(compound) - 2] == 'O' && compound[strlen(compound) - 1] == 'H')

return true;

else

return false;

}

int main(void)

{

char \*materias[] = { "KOH", "H2O2", "NaCl", "NaOH", "C9H8O4", "MgOH" };

for (int i = 0; i < 6; i++)

{

if (isHydroxide(materias[i]) == true)

cout << materias[i] << " e um hidroxido\n";

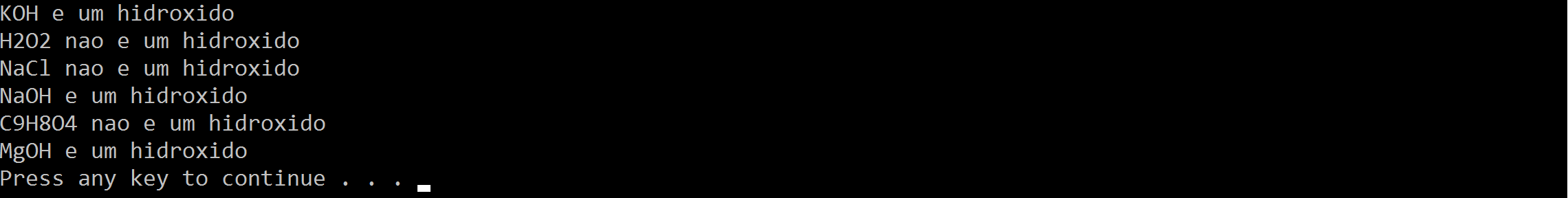
else

cout << materias[i] << " nao e um hidroxido\n";

}

return 0;

}



### 4.1-b)

#include <iostream>

#include <string>

#include <vector>

using namespace std;

bool isHydroxide(string compound)

{

if (compound[compound.length() - 2] == 'O' && compound[compound.length() - 1] == 'H')

return true;

else

return false;

}

int main(void)

{

vector<string> materias = { "KOH", "H2O2", "NaCl", "NaOH", "C9H8O4", "MgOH" };

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

{

if (isHydroxide(materias[i]) == true)

cout << materias[i] << " e um hidroxido\n";

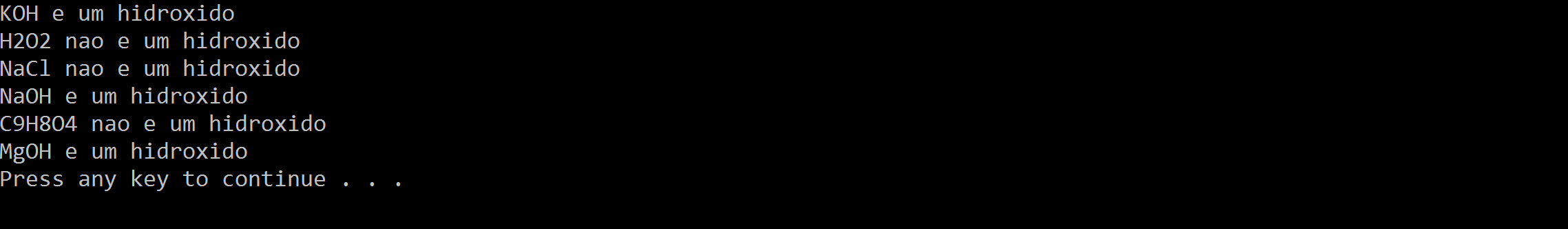
else

cout << materias[i] << " nao e um hidroxido\n";

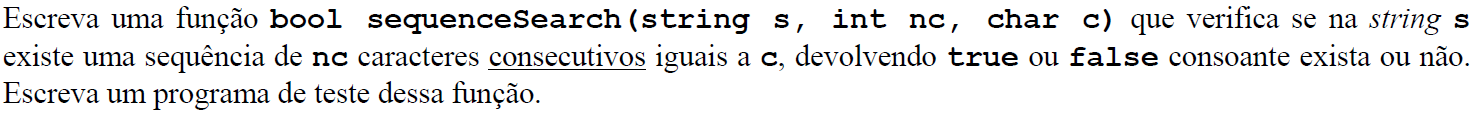
}

return 0;

}



## 4.2



#include <iostream>

#include <string>

#include <vector>

using namespace std;

bool sequenceSearch(string s, int nc, char c)

{

string cmp;

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

cmp.push\_back(c);

if (s.find(cmp) != s.npos)

return true;

else

return false;

}

int main(void)

{

vector<string> testes = { "AAAmasm" ,"AA2dse", "ssAAAsfg", "asdaAAA", "A A A", "AA A", "AAa" };

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

{

if (sequenceSearch(testes[i], 3, 'A') == true)

cout << testes[i] << " contem sub-string\n" << endl;

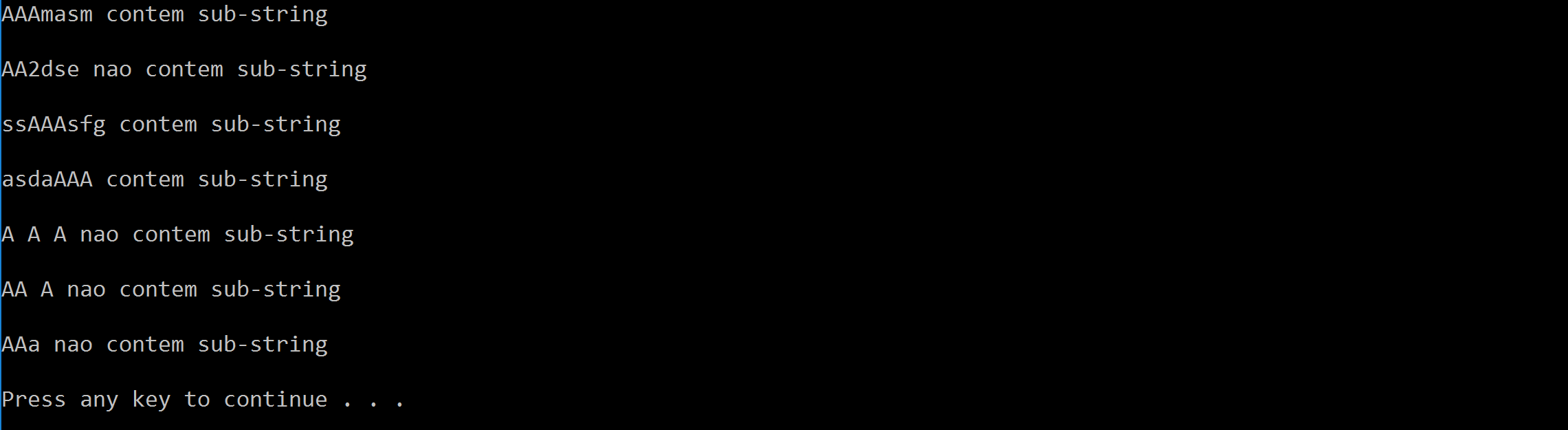
else

cout << testes[i] << " nao contem sub-string\n" << endl;

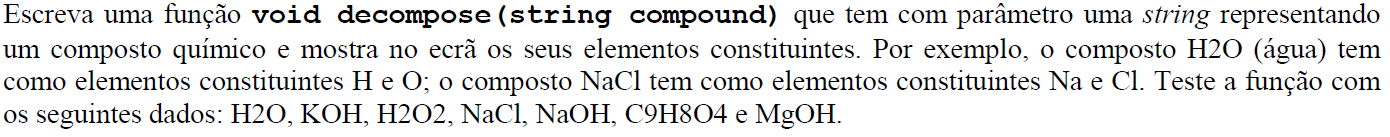
}

return 0;

}



## 4.3



#include <iostream>

#include <string>

#include <vector>

using namespace std;

void decompose (string compound)

{

vector<string> matter = { "H", "O", "Na", "Cl", "Mg", "K", "C"};

bool cl = false; //Para verificar se nao esta a confundir C com Cl

cout << compound << " e composto por: ";

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

{

if (compound.find(matter[i]) != compound.npos && i == 3)

{

cout << matter[i] << " ";

cl = true;

}

else if (compound.find(matter[i]) != compound.npos && i == 6 && cl == false)

cout << matter[i] << " ";

else if (compound.find(matter[i]) != compound.npos && i != 6 && i != 3)

cout << matter[i] << " ";

}

cout << endl;

}

int main(void)

{

vector<string> testes = { "H2O", "KOH", "H2O2", "NaCl", "NaOH", "C9H8O4", "MgOH" };

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

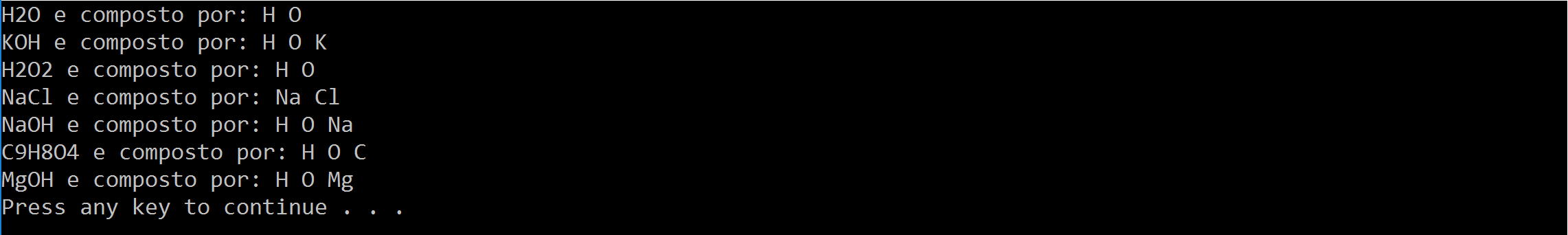
{

decompose(testes[i]);

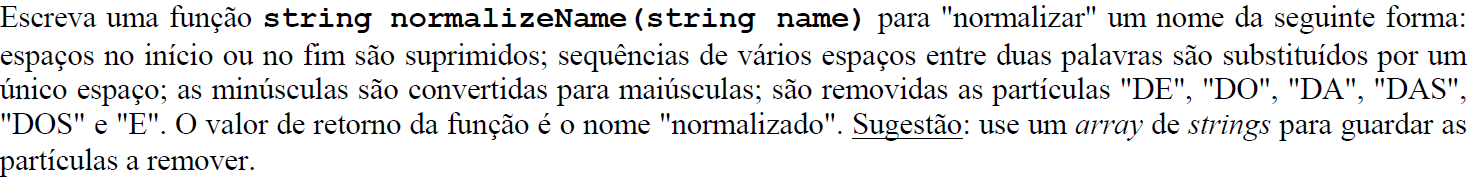
}

return 0;

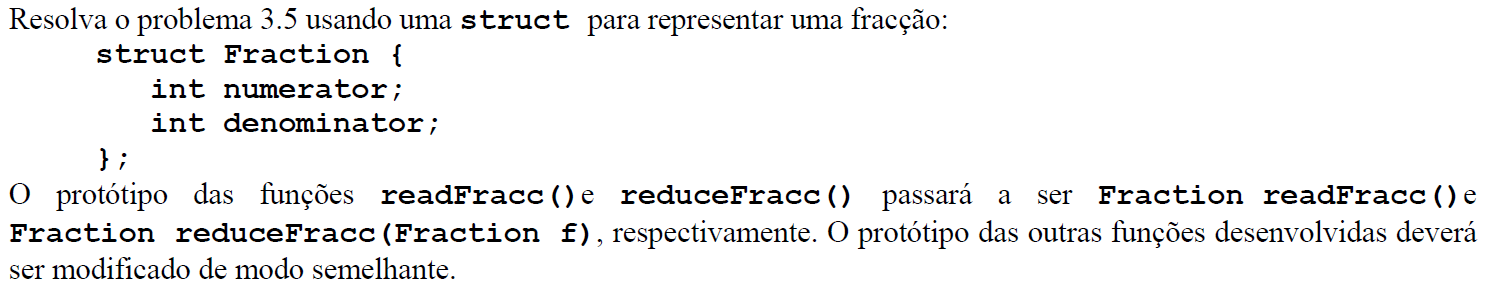
}



## 4.4



## 4.5



#include <iostream>

#include <string>

using namespace std;

#define ADICAO 0

#define SUBTRACAO 1

#define MULTIPLICACAO 2

#define DIVISAO 3

typedef struct {

int numerator;

int denominator;

} fraction;

void writeFracc(fraction f);

fraction readFracc();

void reduceFracc(fraction &f);

fraction somaFracc(fraction a, fraction b, int op); //se op == ADICAO entao soma, se op == SUBTRACAO subtrai

fraction multFracc(fraction a, fraction b, int op); // se op == MULTIPLICACAO entao multiplica, se op == DIVISAO divide

int main(void)

{

fraction a, b;

char op;

a = readFracc();

b = readFracc();

if (!(a.denominator == NULL || a.numerator == NULL) && !(b.denominator == NULL || b.numerator == NULL))

cout << "Fracoes validas!\n";

else

cout << "fracoes nao validas!\n";

reduceFracc(a);

reduceFracc(b);

for (int i = 0; i < 4; i++)

{

cout << "Indique a operaçao que pretende fazer (+ - \* /): ";

cin >> op;

switch (op)

{

case '+':

writeFracc(a);

cout << " + ";

writeFracc(b);

cout << " = ";

writeFracc(somaFracc(a, b, ADICAO));

cout << endl;

break;

case '-':

writeFracc(a);

cout << " - ";

writeFracc(b);

cout << " = ";

writeFracc(somaFracc(a, b, SUBTRACAO));

cout << endl;

break;

case '\*':

writeFracc(a);

cout << " \* ";

writeFracc(b);

cout << " = ";

writeFracc(multFracc(a, b, MULTIPLICACAO));

cout << endl;

break;

case '/':

writeFracc(a);

cout << " / ";

writeFracc(b);

cout << " = ";

writeFracc(multFracc(a, b, DIVISAO));

cout << endl;

break;

}

}

return 0;

}

void writeFracc(fraction f)

{

cout << f.numerator << "/" << f.denominator;

}

fraction readFracc()

{

string input;

size\_t pos;

fraction f;

cout << "Introduza a fraccao ( N/D ): ";

cin >> input;

pos = input.find("/");

if (pos == string::npos)

{

f.numerator = NULL;

f.denominator = NULL;

return f;

}

else

{

f.numerator = stoi(input.substr(0, pos));

f.denominator = stoi(input.substr(pos + 1));

return f;

}

}

void reduceFracc(fraction &f)

{

int maior = 1;

if (f.numerator < f.denominator)

{

for (int i = 1; i <= f.numerator; i++)

{

if (f.numerator % i == 0 && f.denominator % i == 0)

maior = i;

}

f.numerator /= maior;

f.denominator /= maior;

}

else if (f.numerator > f.denominator)

{

for (int i = 1; i <= f.denominator; i++)

{

if (f.numerator % i == 0 && f.denominator % i == 0)

maior = i;

}

f.numerator /= maior;

f.denominator /= maior;

}

else

{

f.numerator /= f.numerator;

f.denominator /= f.denominator;

}

}

fraction somaFracc(fraction a, fraction b, int op) //se op 0 entao soma, se op b subtrai

{

fraction c;

if (op == 0)

{

//Primeiro -> Multiplicar os numeradores pelos denominadores para ter fraccoes equivalentes

c.numerator = a.numerator \* b.denominator + b.numerator \* a.denominator;

//Segundo-> Meter as fraccoes com o mesmo denominador

c.denominator = b.denominator \* a.denominator;

}

else

{

//Primeiro -> Multiplicar os numeradores pelos denominadores para ter fraccoes equivalentes

c.numerator = a.numerator \* b.denominator - b.numerator \* a.denominator;

//Segundo-> Meter as fraccoes com o mesmo denominador

c.denominator = b.denominator \* a.denominator;

}

reduceFracc(c);

return c;

}

fraction multFracc(fraction a, fraction b, int op) // se op 0 entao multiplica, se op 1 divide

{

fraction c;

if (op == 2)

{

c.numerator = a.numerator \* b.numerator;

c.denominator = a.denominator \* b.denominator;

}

else

{

c.numerator = a.numerator \* b.denominator;

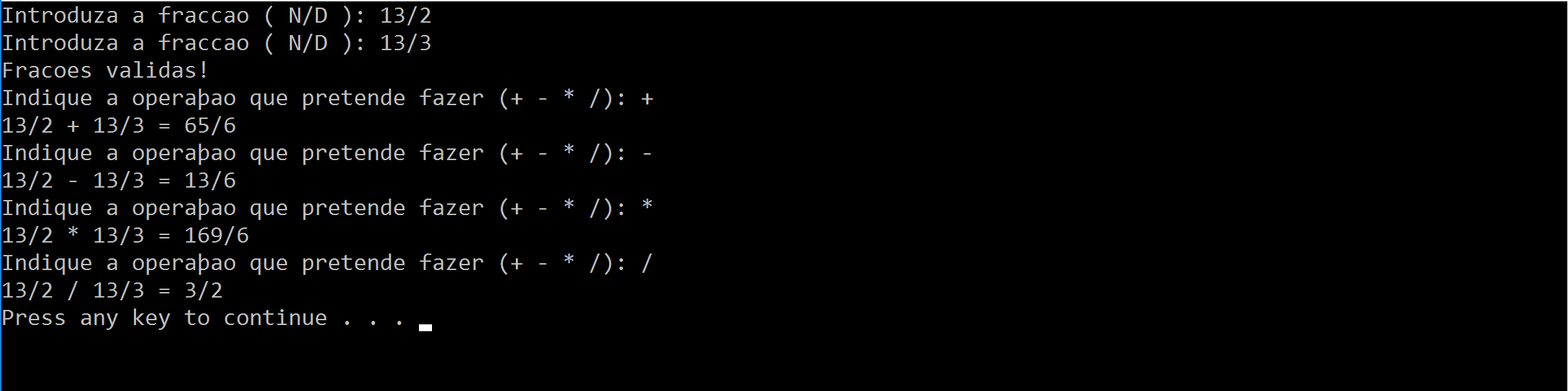
c.denominator = b.numerator \* a.denominator;

}

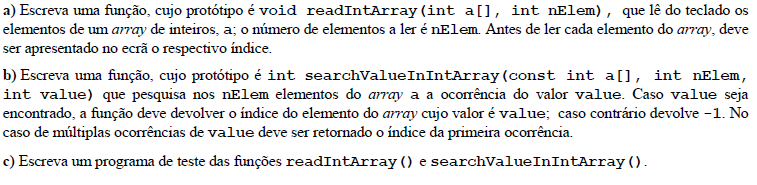
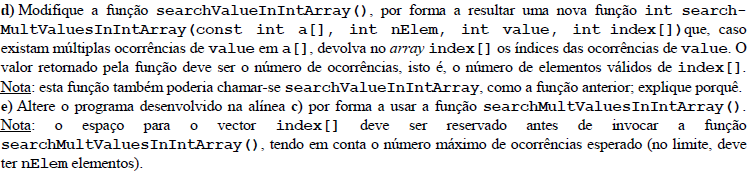
reduceFracc(c);

return c;

}



## 4.6



### 4.6-a)

#include <iostream>

#include <string>

#include <stdlib.h>

using namespace std;

void readIntArray(int a[], int nElem)

{

a = (int\*)realloc(a, nElem \* sizeof(int)); //realocar para o numero de bytes correspondentes (n de elementos \* tamanho de int)

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

{

cout << i << " -> ";

cin >> a[i];

cout << endl;

}

}



### 4.6-b)

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

{

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

{

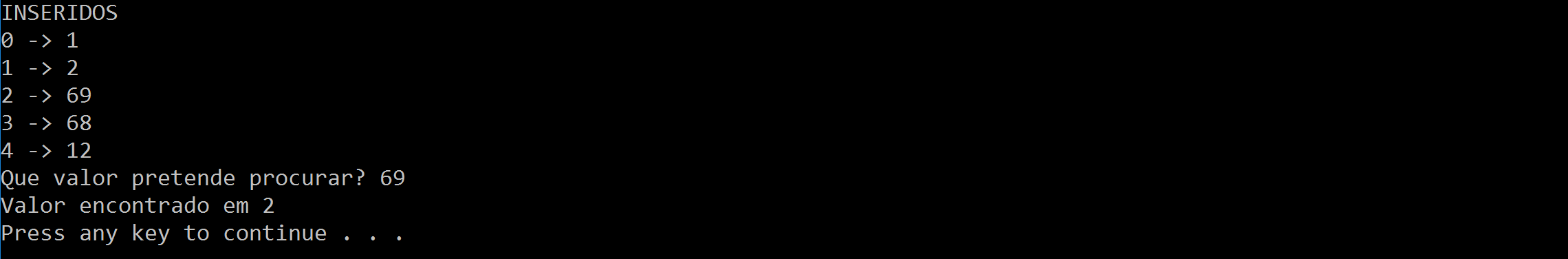
if (a[i] == value)

return i;

}

return -1;

}



### 4.6-c)

#include <iostream>

#include <string>

#include <stdlib.h>

using namespace std;

void readIntArray(int a[], int nElem)

{

…

}

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

{

…

}

int main(void)

{

int \*v1 = (int \*)malloc(sizeof(int)); //inicializar apontador para vetor

int elem, valor, pos;

cout << "Quantos elementos pretende inserir?";

cin >> elem;

readIntArray(v1, elem);

cout << "INSERIDOS\n";

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

cout << i << " -> " << v1[i] << endl;

cout << "Que valor pretende procurar? ";

cin >> valor;

pos = searchValueInIntArray(v1, elem, valor);

if (pos == -1)

cout << "Nao encontrou o valor\n";

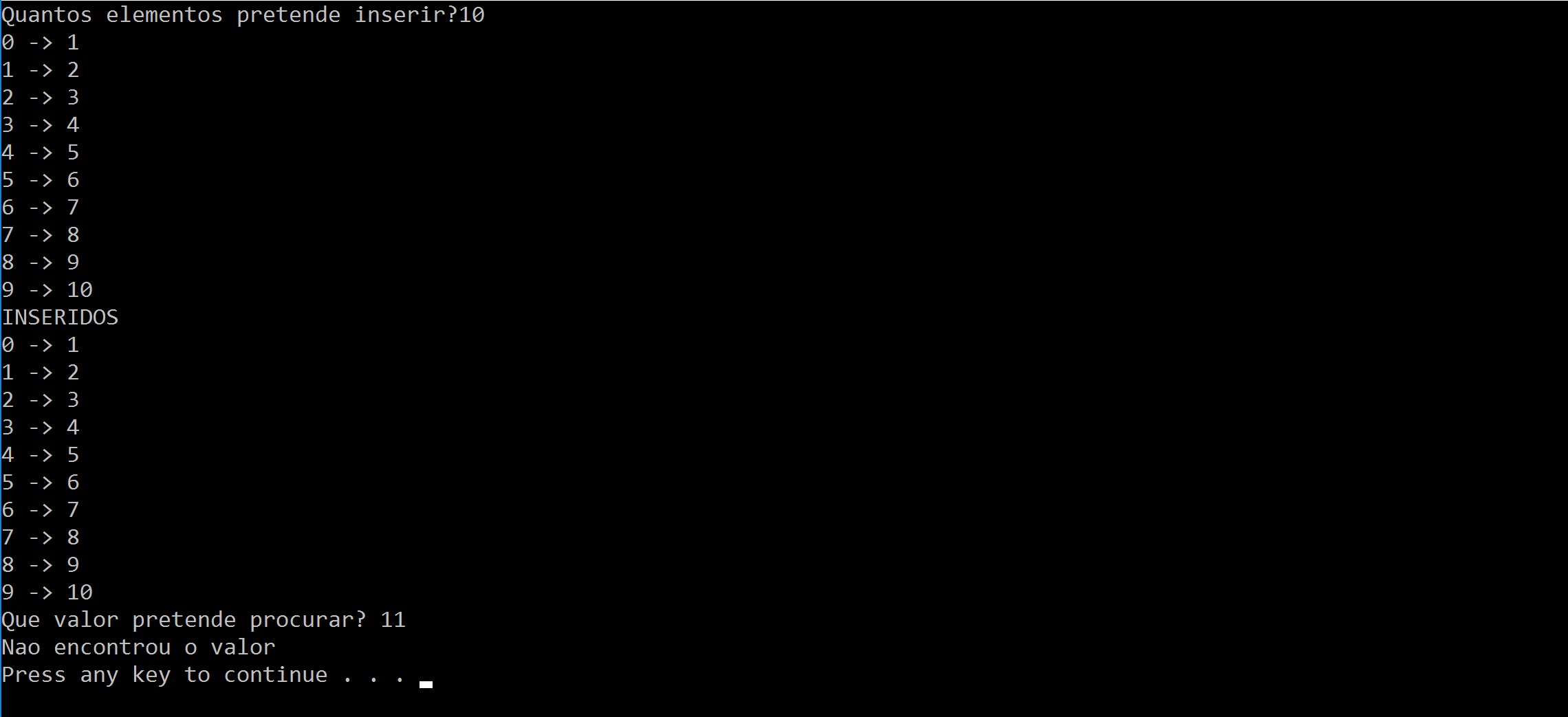
else

cout << "Valor encontrado em " << pos << endl;

free(v1); //Libertar a memoria

return 0;

}



### 4.6-d)

int searchMultValuesInIntArray(const int a[], int nElem, int value, int index[])

{

int n\_i = 0;

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

{

if (a[i] == value)

{

index[n\_i] = i;

n\_i++;

}

}

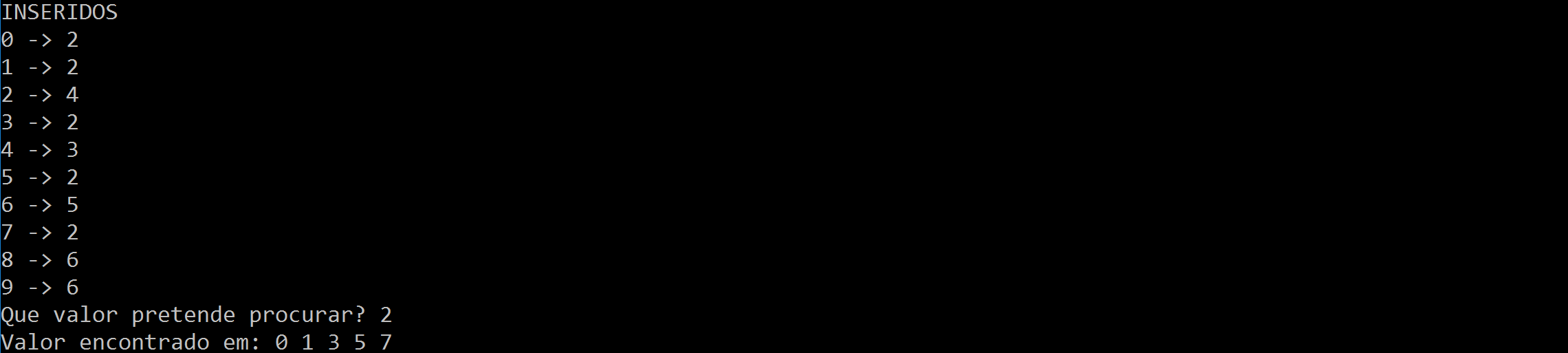
if (n\_i != 0)

return n\_i;

else

return -1;

}



### 4.6-e)

#include <iostream>

#include <string>

#include <stdlib.h>

using namespace std;

void readIntArray(int a[], int nElem){…}

int searchValueInIntArray(const int a[], int nElem, int value){…}

int searchMultValuesInIntArray(const int a[], int nElem, int value, int index[]){…}

int main(void)

{

int \*v1 = (int \*)malloc(sizeof(int)); //inicializar apontador para vetor

int \*indices = (int \*)malloc(sizeof(int));

int elem, valor, pos;

cout << "Quantos elementos pretende inserir?";

cin >> elem;

indices = (int\*)realloc(indices, (elem+1) \* sizeof(int));

readIntArray(v1, elem);

cout << "INSERIDOS\n";

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

cout << i << " -> " << v1[i] << endl;

cout << "Que valor pretende procurar? ";

cin >> valor;

pos = searchMultValuesInIntArray(v1, elem, valor, indices);

if (pos == -1)

cout << "Nao encontrou o valor\n";

else

{

if (pos == 1)

cout << "Valor encontrado em " << pos << endl;

else

{

cout << "Valor encontrado em: ";

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

{

cout << indices[i] << " ";

}

cout << endl;

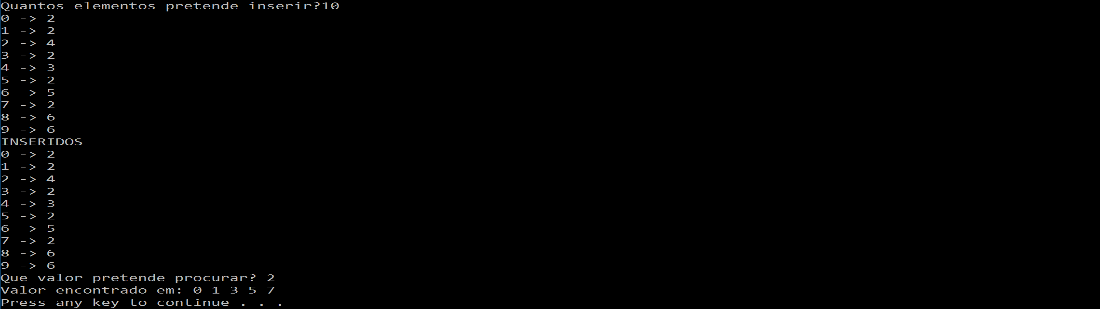
}

}

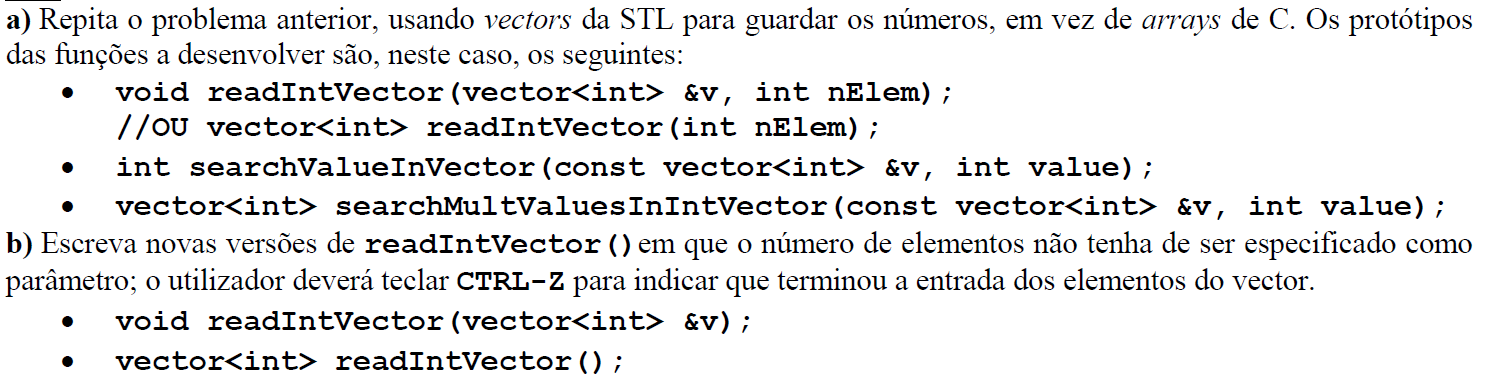
free(v1);

free(indices); //Libertar a memoria

return 0;

}

## 4.7



### 4.7-a)

#include <iostream>

#include <vector>

using namespace std;

vector<int> readIntVector(int nElem)

{

vector<int> v1;

int x;

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

{

cout << i << " -> ";

cin >> x;

v1.push\_back(x);

}

return v1;

}

int searchValueInIntArray(const vector<int> &v, int value)

{

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

if (v[i] == value)

return i;

return -1;

}

vector<int> searchMultValuesInIntArray(const vector<int> &v, int value)

{

vector<int> r;

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

if (v[i] == value)

r.push\_back(i);

return r;

}

int main(void)

{

vector<int> elementos, indices;

int elem, value;

cout << "Quantos valores pretende inserir?";

cin >> elem;

elementos = readIntVector(elem);

cout << "\nINSERIDOS" << endl;

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

cout << i << " -> " << elementos[i] << endl;

cout << "Que valor quer procurar?\n";

cin >> value;

indices = searchMultValuesInIntArray(elementos, value);

if ((int)indices.size() == 0)

cout << "nao foram encontrados valores\n";

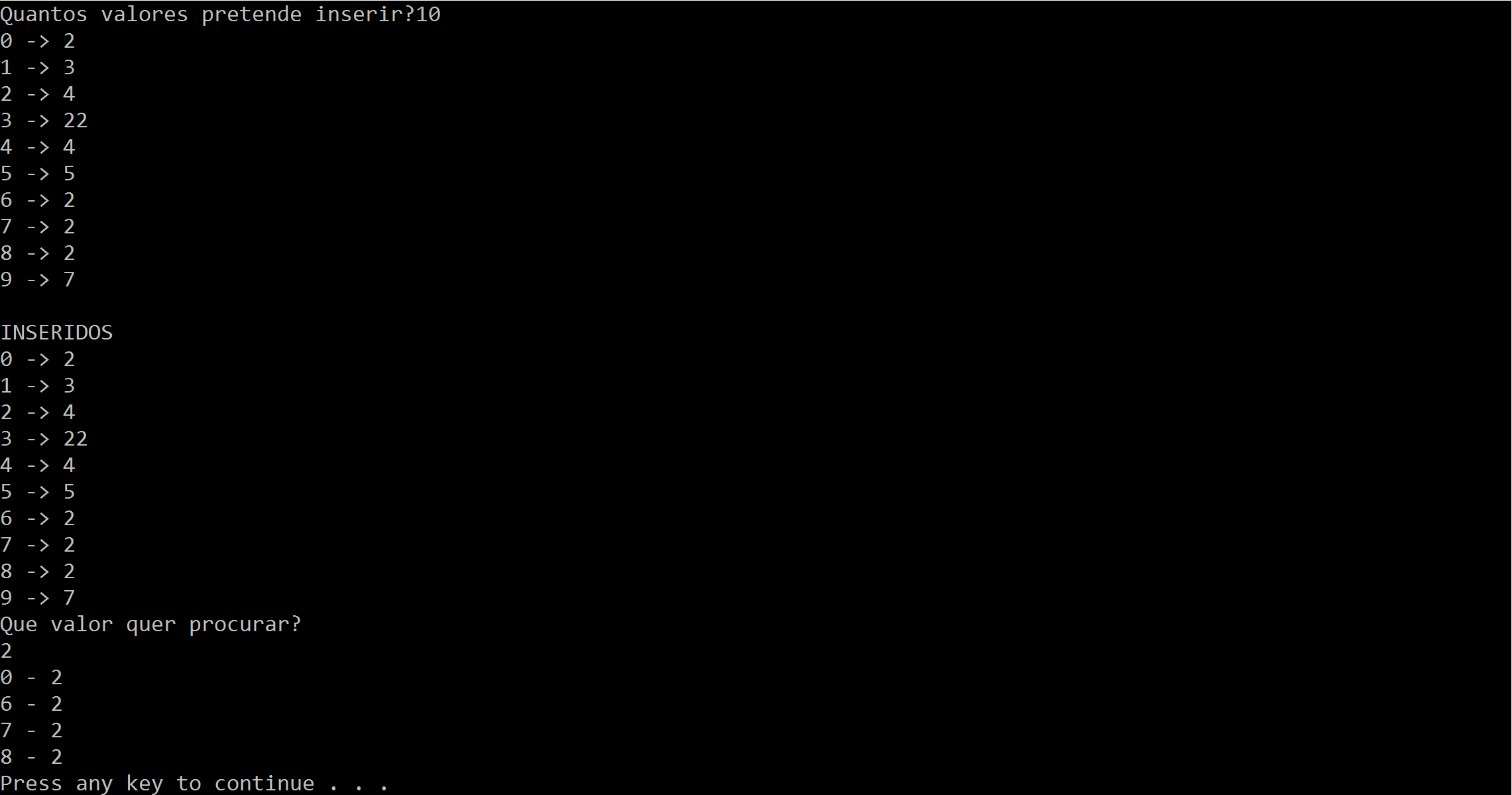
else

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

cout << indices[i] << " - " << elementos[indices[i]] << endl;

return 0;

}



### 4.7-b)

vector<int> readIntVector()

{

vector<int> v1;

int x, i = 0;

while (cin.eof() != true)

{

cout << i << " -> ";

cin >> x;

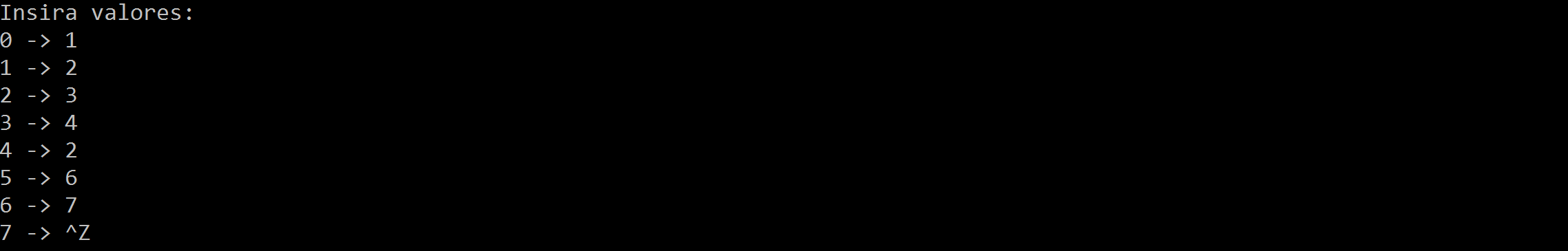
v1.push\_back(x);

i++;

}

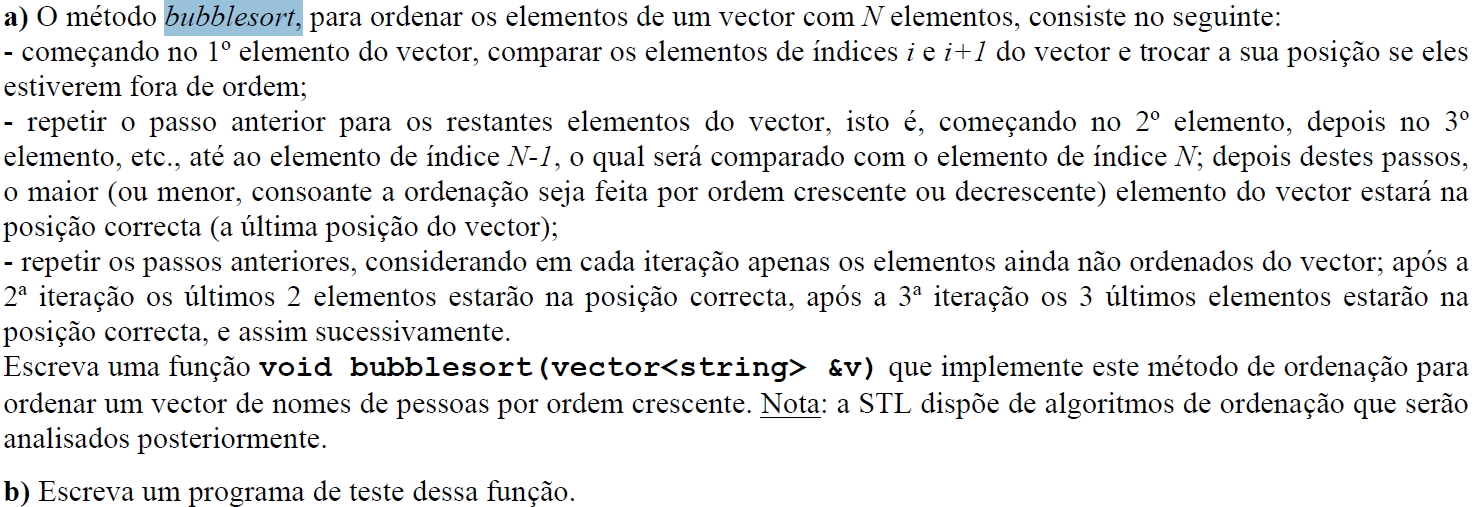
return v1;

}



## 4.8

<https://visualgo.net/sorting> -> para perceber o bubblesorting



### 4.8-a)

#include <iostream>

#include <vector>

#include <string>

#include <cstring>

#include <stdlib.h>

using namespace std;

void to\_upper(string &str)

{

for (int i = 0; i < (int)str.length(); i++)

{

if (str[i] > 96 && str[i] < 123)

str[i] -= 32;

}

}

void bubblesort(vector<string> &v)

{

int j;

string aux;

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

to\_upper(v[i]);

j = (int)v.size();

while (j != 1)

{

for (int i = 0; i < j - 1; i++)

{

if (v[i].compare(v[i + 1]) > 0) //por ordem alfabetica

{

aux.assign((const char \*)v[i].c\_str());

v[i].assign((const char \*)v[i + 1].c\_str());

v[i + 1].assign((const char \*)aux.c\_str());

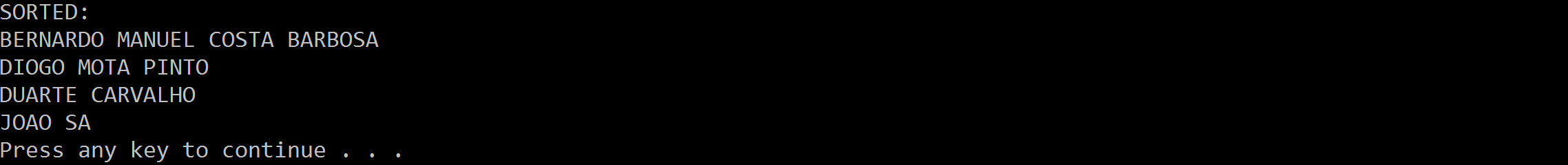
}

}

j--;

}

}



### 4.8-b)

#include <iostream>

#include <vector>

#include <string>

#include <cstring>

#include <stdlib.h>

using namespace std;

void to\_upper(string &str)

{

…

}

void bubblesort(vector<string> &v)

{

…

}

int main(void)

{

vector<string> nomes = { "Joao Sa", "Duarte Carvalho", "Diogo Mota Pinto", "Bernardo Manuel Costa Barbosa" };

cout << "Without sorting:\n";

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

cout << nomes[i] << endl;

bubblesort(nomes);

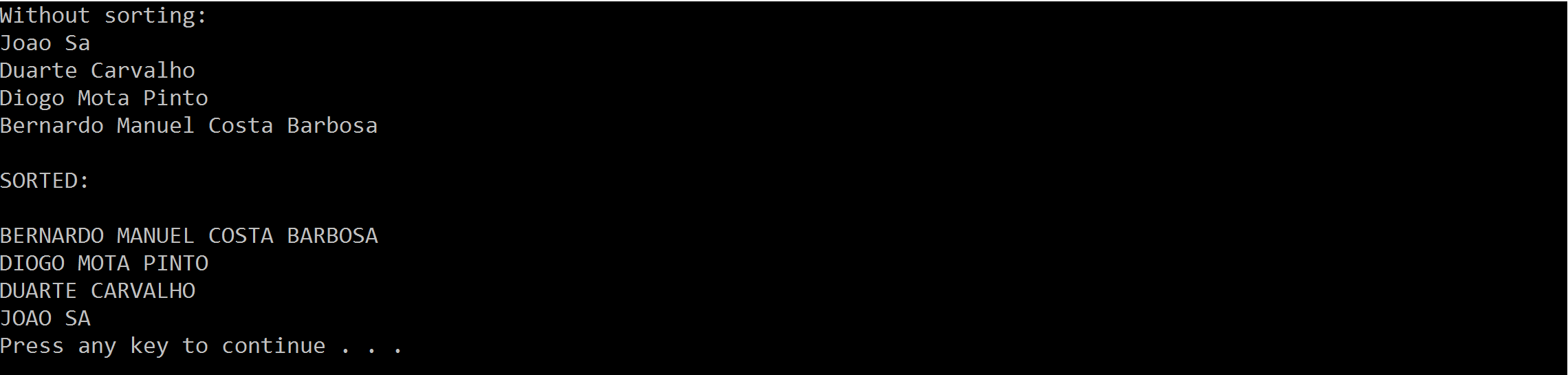
cout << endl << "SORTED:\n" << endl;

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

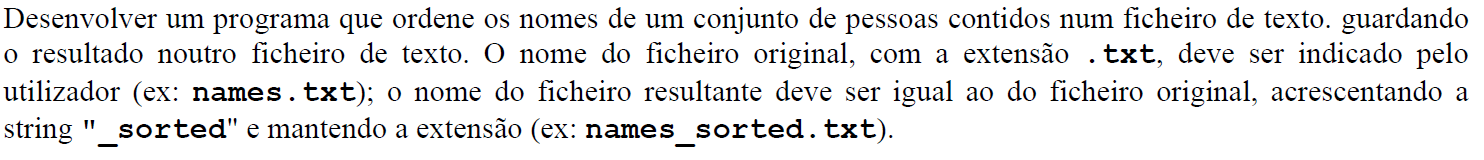
cout << nomes[i] << endl;

return 0;

}



## 4.13



#include …

using namespace std;

void to\_upper(string &str)

{…}

void bubblesort(vector<string> &v)

{…}

int main(void)

{

string input\_filename, output\_filename, line;

ifstream fin;

ofstream fout;

vector<string> nomes;

cout << "File name?";

cin >> input\_filename;

output\_filename.append("\_sorted");

fin.open(input\_filename);

if (!fin.is\_open())

{

cerr << "File " << input\_filename << " not found !\n";

return(1);

}

while (getline(fin, line))

{

nomes.push\_back(line); // TO DO: process the line

}

fin.close();

bubblesort(nomes);

fout.open("nomes\_sorted.txt");

if (!fout.is\_open())

{

cerr << "File " << output\_filename << " not found !\n";

return(1);

}

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

fout << nomes[i] << endl;

fout.close();

return 0;

}