**2ª Lista de Exercícios**

1. Desenvolva um programa em linguagem C/C++ que receba dez valores reais (*float*) e os armazene em um vetor. Em seguida, exiba os valores armazenados no vetor.

Exemplo:

**Entrada**

10.3 4.4 5.23 7.28 15.9 1928.0 1010123.1234 1.1 1.2 20.4

**Saída**

10.3

4.4

5.23

7.28

15.9

1928.0

1010123.1234

1.1

1.2

20.4

**Observação:** exiba apenas um valor por linha, seguido imediatamente de <enter> (‘\n’ ou std::endl) sem espaços

1. Desenvolva um programa em linguagem C/C++ que receba dez valores reais (*float*) e os armazene em um vetor. Em seguida, exiba os valores do vetor na seguinte ordem:

v9, v8, v7, ..., v1, v

Exemplo:

**Entrada**

10.3 4.4 5.23 7.28 15.9 1928.0 1010123.1234 1.1 1.2 20.4

**Saída**

|  |
| --- |
| 20.4 |
| 1.2 |
| 1.1 |
| 10101231234 |
| 1928.0 |
| 15.9 |
| 7.28 |
| 5.23 |
| 4.4 |
| 10.3 |

**Observação:** exiba apenas um valor por linha, seguido imediatamente de <enter> (‘\n’ ou std::endl) sem espaços

1. Desenvolva um programa em linguagem C/C++ que receba dez valores reais (*float*) e os armazene em um vetor. Em seguida, exiba os valores armazenados nas posições pares do vetor seguido dos valores armazenados nas posições ímpares. (considere a posição 0 (zero) como par).

Exemplo:

**Entrada**

10.3 4.4 5.23 7.28 15.9 1928.0 1010123.1234 1.1 1.2 20.4

**Saída**

10.3

5.23

15.9

1010123.1234

1.2

4.4

7.28

1928.0

1.1

20.4

1. Desenvolva um programa em linguagem C/C++ que receba *n* valores reais (*float*) e os armazene em um vetor. O primeiro valor recebido deve ser considerado o tamanho do vetor (*n)* os *n* valores subsequentes devem ser armazenados consecutivamente no vetor. Em seguida, exiba os valores armazenados nas posições cujos valores forem números primos.

Exemplo:

**Entrada**

12 10.3 4.4 5.23 7.28 15.9 1928.0 1010123.1234 1.1 1.2 20.4 1.123 0.55

**Saída**

4.4

5.23

1928.0

1.1

0.53

1. Desenvolva um programa em linguagem C/C++ que receba n valores reais (float) e os armazene em um vetor. O primeiro valor recebido deve ser considerado o tamanho do vetor (n) os n valores subsequentes devem ser armazenados consecutivamente no vetor. Em seguida exiba o valor da soma dos valores armazenados no vetor.

Exemplo:

**Entrada**

12 10.3 4.4 5.23 7.28 15.9 1928.0 1010123.1234 1.1 1.2 20.4 1.123 0.55

**Saída**

1012118,6064

1. Desenvolva um programa em linguagem C/C++ que receba *n* letras (char) e as armazene em um vetor. O primeiro valor recebido deve ser considerado o tamanho do vetor (n) os n valores subsequentes são letras que devem ser armazenados consecutivamente no vetor. Em seguida exiba a frase formada pelos caracteres recebidos.

Exemplo:

**Entrada**

12 A b a c a t e \_ e \_ u m a \_ f r u t a

**Saída**

Abacate e uma fruta

**Observação:** no exemplo anterior, considera-se o *underline* ( \_ ) como um espaço

1. Desenvolva um programa em linguagem C/C++ que receba vinte números e os armazene em dois vetores de números inteiros (va e vb). Os primeiros dez números devem ser armazenados no vetor va e os dez números seguintes no vetor vb. Em seguida gere o vetor vr, com vinte posições, com os valores de va e vb intercalados. Sendo assim, o vetor vr deverá armazenar os valores na seguinte ordem:

*vr = {va****0****,vb****0****, va****1****,vb****1****, va****2****,vb****2****, ... , va****n-1****,vb****n-1****}*

Exemplo:

**Entrada**

10 20 30 40 50 60 70 80 90 100 1 2 3 4 5 6 7 8 9 10

**Saída**

10

1

20

2

30

3

40

4

50

5

60

6

70

7

80

8

90

9

100

10

**Observação:** exiba apenas um valor por linha, seguido imediatamente de <enter> (‘\n’ ou std::endl) sem espaços

1. Desenvolva um programa em linguagem C/C++ que receba vinte números e os armazene em dois vetores de números inteiros (va e vb). Os primeiros dez números devem ser armazenados no vetor va e os dez números seguintes no vetor vb. Em seguida gere o vetor vr, com vinte posições, com os valores de va e vb intercalados garantindo que eles estejam armazenados em vr em ordem ascendente. Ao final exiba os valores de vr.

**Entrada**

10 20 30 40 50 60 70 80 90 100 1 2 3 4 5 6 7 8 9 10

**Saída**

1 2 3 4 5 6 7 8 9 10 10 20 30 40 50 60 70 80 90 100

**Observação:** entre um número e outro deve existir apenas um espaço e um <enter> ao final

1. Desenvolva um programa em linguagem C/C++ que defina um vetor com *n* posições. Cada elemento do vetor deve ter a estrutura *{chave, valor}*. Observe a declaração:

struct sItemVector {

int chave;

string valor;

} \* vetor;

O programa deverá receber *n* pares de valores armazenando cada par de valores no vetor. O primeiro valor recebido é o tamanho do vetor, ou seja, *n*. Em seguida o programa deverá receber *m* valores. Para cada valor recebido, o programa deverá pesquisar o par no vetor que tenha a chave igual ao valor e exibi-lo. Caso não exista uma chave igual ao valo informado o programa deverá exibir a sigla (**NE**) no lugar do valor. A pesquisa e exibição de valores deverá encerrar-se quando um valor menor do que zero for recebido.

Exemplo:

**Entrada**

5 10 A 30 B 101 AA 1001 CAB 100 WWWAAA 101 104 1001 12 -1

**Vetor de estrutura**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** | 1 | **2** | **3** | **4** |
| |  |  | | --- | --- | | **10** | **A** | | |  |  | | --- | --- | | **30** | **B** | | |  |  | | --- | --- | | **101** | **AA** | | |  |  | | --- | --- | | **1001** | **CAB** | | |  |  | | --- | --- | | **100** | **WWWAAA** | |

**Saída**

101 AA

104 NE

1001 CAB

12 NE

**Observação:** exibir um par por linha. Entre a chave e o valor deve existir apenas um espaço. Ao final da exibição do par, deve existir um <enter>.

1. Desenvolva um programa em linguagem C/C++ que defina um vetor de números inteiros com dez posições. O programa deverá gerar a distribuição de frequência de *n* números recebidos. Os números recebidos devem estar no intervalo fechado entre 0 e 99, qualquer número recebido que não pertencer ao intervalo deve ser descartado. Cada posição do vetor representa uma faixa (ou intervalo de frequência) da distribuição de frequência como segue:

|  |  |
| --- | --- |
| **índice** | **intervalo** |
| 0 | 0 a 9 |
| 1 | 10 a 19 |
| 2 | 20 a 29 |
| 3 | 30 a 39 |
| 4 | 40 a 49 |
| 5 | 50 a 59 |
| 6 | 60 a 69 |
| 7 | 70 a 79 |
| 8 | 80 a 89 |
| 9 | 90 a 99 |
|  |  |

O primeiro valor recebido é a quantidade de valores que serão recebidos na sequência. Para cada valor recebido deve-se:

1. Verificar se o valor é valido. Caso não seja o programa deve descarta-lo.
2. Caso o valor seja válido, o programa deverá procurar o intervalo de frequência ao qual o número pertence.
3. Somar um a posição referente ao intervalo selecionado.

Ao final o programa deverá exibir a distribuição de frequência dos valores recebidos. Exemplo:

**Entrada**

8 100 99 57 51 13 12 11 10

|  |
| --- |
| 0-9 0 |
| 10-19 4 |
| 20-29 0 |
| 30-39 0 |
| 40-49 0 |
| 50-59 2 |
| 60-69 0 |
| 70-79 0 |
| 80-89 0 |
| 90-99 1 |

**Saída**