Professor Pietro Martins de Oliveira

ALGORITMOS

do início ao fim

PACOTE DE EXERCÍCIOS 5: ESTRUTURAS DE DADOS BÁSICAS EM C

1) Desenvolva um algoritmo que preencha um vetor numérico de 10 posições. Ao final, o algoritmo deve mostrar o somatório de todos os elementos do vetor, bem como a média aritmética entre todos os termos.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

Exemplo de execução – Exercício 1 - Caso de teste

```
Insira o dado da posição  1:
1
Insira o dado da posição  2:
2
Insira o dado da posição  3:
3
Insira o dado da posição  4:
4
Insira o dado da posição  5:
5
Insira o dado da posição  6:
6
Insira o dado da posição  7:
7
Insira o dado da posição  8:
8
Insira o dado da posição  9:
9
Insira o dado da posição  10:
10
Somatório:  55
Média:  5.5
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

VETORES – Exercício 1 – Solução

```c
#include <stdio.h>
int main(){
    float vet[10];
    float soma, media;
    int i;
    soma = 0;
    for(i=0;i<10;i++){
        printf("Insira o dado da posição %d:\n", i+1);
        scanf("%f", &vet[i]);
        soma = soma + vet[i];
    }
    media = soma/10;
```

```
13      printf("Somatório: %f.\n", soma);
14      printf("Média: %f.\n", media);
15  }
```

2) Desenvolva um algoritmo que peça ao usuário que preencha os dados de um vetor de 5 posições com valores reais quaisquer, desde que estejam compreendidos entre 1 e 100 (suponha que o usuário irá respeitar o enunciado). Ao final, o algoritmo deve mostrar, na tela, o conteúdo de cada posição do vetor, dividido por 100.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução – Exercício 2 - Caso de teste
Insira o dado da posição  1:
10
Insira o dado da posição  2:
20
Insira o dado da posição  3:
30
Insira o dado da posição  4:
40
Insira o dado da posição  5:
50
Conteúdo dividido por 100:
 0.1
 0.2
 0.3
 0.4
 0.5
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

```
VETORES – Exercício 2 – Solução
01  #include <stdio.h>
02  int main(){
03      float vet[5];
04      int i;
05      for(i=0; i<5; i++){
06          printf("Insira o dado da posição %d:\n", i+1);
07          scanf("%f", &vet[i]);
08          vet[i] = vet[i]/100.0;
09      }
10      printf("Conteúdo divido por 100:\n");
```

```
11      for(i=0; i<5; i++){
12          printf("%f ", vet[i]);
13      }
14  }
```

3) Desenvolva um algoritmo que preencha um vetor numérico de 10 posições. Após preencher todo o vetor, o usuário deve inserir uma chave de busca X. Caso exista algum número igual a X, dentro do vetor, o algoritmo deve mostrar, na tela, o índice da primeira posição na qual X foi encontrado. Caso contrário, o algoritmo deve se encerrar com uma única mensagem, dizendo "Chave não encontrada.".

Quer dicas de como seu algoritmo deveria funcionar? Observe os quadros abaixo, nos quais você encontra simulações da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução – Exercício 3 - Caso de teste 1
Insira o dado da posição  1:
11
Insira o dado da posição  2:
12
Insira o dado da posição  3:
13
Insira o dado da posição  4:
14
Insira o dado da posição  5:
15
Insira o dado da posição  6:
16
Insira o dado da posição  7:
17
Insira o dado da posição  8:
18
Insira o dado da posição  9:
19
Insira o dado da posição  10:
20
Insira a chave de busca:
15
Chave encontrada na posição:  5
```

```
Exemplo de execução – Exercício 3 - Caso de teste 2
Insira o dado da posição  1:
11
Insira o dado da posição  2:
12
Insira o dado da posição  3:
13
```

```
Insira o dado da posição  4:
14
Insira o dado da posição  5:
15
Insira o dado da posição  6:
16
Insira o dado da posição  7:
17
Insira o dado da posição  8:
18
Insira o dado da posição  9:
19
Insira o dado da posição  10:
20
Insira a chave de busca:
100
Chave não encontrada.
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

VETORES – Exercício 3 – Solução

```c
#include <stdio.h>
#include <stdbool.h>
int main(){
    float X, vet[10];
    int i, p;
    bool achou;
    for(i=0; i<10; i++){
        printf("Insira o dado da posição %d:\n", i+1);
        scanf("%f", &vet[i]);
    }
    printf("Insira a chave de busca:\n");
    scanf("%f", &X);
    achou = false;
    i = 0;
    while((i < 10) && (achou == false)){
        if(vet[i] == X){
            achou = true;
            p = i;
        }
        i++;
    }
    if(!achou){ //!achou é a mesma coisa que achou == false
        printf("Chave não encontrada.\n");
    } else { //achou == true?
        printf("Chave encontrada na posição: %d.\n", p+1);
    }
}
```

4) (Adaptado de ASCENCIO e CAMPOS, 2008) Faça um programa que carregue um vetor e nove elementos numéricos inteiros, calcule e mostre os números primos e suas respectivas posições.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução – Exercício 5 - Caso de teste
Insira o dado da posição  1:
1
Insira o dado da posição  2:
3
Insira o dado da posição  3:
7
Insira o dado da posição  4:
9
Insira o dado da posição  5:
11
Insira o dado da posição  6:
13
Insira o dado da posição  7:
17
Insira o dado da posição  8:
19
Insira o dado da posição  9:
21
 3 é primo, posição:  2
 7 é primo, posição:  3
 11 é primo, posição:  5
 13 é primo, posição:  6
 17 é primo, posição:  7
 19 é primo, posição:  8
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

VETORES – Exercício 4 – Solução

```c
#include <stdio.h>
int main(){
    int N, i, j, cont;
    int vet[9];
    for(i=0; i<9; i++){
        printf("Insira o dado da posição %d:\n", i+1);
        scanf("%d", &vet[i]);
    }
```

```
09      for(i=0; i<9; i++){
10          cont = 0;
11          for(j=1; j<=vet[i]; j++){
12              if((vet[i] % j) == 0){
13                  cont++;
14              }
15          }
16          if(cont == 2){
17              printf("%d é primo, posição: %d.\n", vet[i], i+
    1);
18          }
19      }
20  }
```

5) (Adaptado de ASCENCIO e CAMPOS, 2008) Faça um programa que carregue dois vetores de dez elementos numéricos cada um e mostre um vetor resultante da intercalação desses dois vetores.

Quer dicas de como seu algoritmo deveria funcionar? Observe os quadros abaixo, nos quais você encontra simulações da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 5 - Caso de teste 1
Insira o dado vet1[ 0]:
10
Insira o dado vet1[ 1]:
20
Insira o dado vet1[ 2]:
30
Insira o dado vet1[ 3]:
40
Insira o dado vet1[ 4]:
50
Insira o dado vet1[ 5]:
60
Insira o dado vet1[ 6]:
70
Insira o dado vet1[ 7]:
80
Insira o dado vet1[ 8]:
90
Insira o dado vet1[ 9]:
100
Insira o dado vet2[ 0]:
11
Insira o dado vet2[ 1]:
12
Insira o dado vet2[ 2]:
```

```
13
Insira o dado vet2[ 3]:
14
Insira o dado vet2[ 4]:
15
Insira o dado vet2[ 5]:
16
Insira o dado vet2[ 6]:
17
Insira o dado vet2[ 7]:
18
Insira o dado vet2[ 8]:
19
Insira o dado vet2[ 9]:
20
 10  11  20  12  30  13  40  14  50  15  60  16  70  17  80  18  90
19  100  20
```

Exemplo de execução – Exercício 5 - Caso de teste 2

```
Insira o dado vet1[ 0]:
11
Insira o dado vet1[ 1]:
12
Insira o dado vet1[ 2]:
13
Insira o dado vet1[ 3]:
14
Insira o dado vet1[ 4]:
15
Insira o dado vet1[ 5]:
16
Insira o dado vet1[ 6]:
17
Insira o dado vet1[ 7]:
18
Insira o dado vet1[ 8]:
19
Insira o dado vet1[ 9]:
20
Insira o dado vet2[ 0]:
21
Insira o dado vet2[ 1]:
22
Insira o dado vet2[ 2]:
23
Insira o dado vet2[ 3]:
24
Insira o dado vet2[ 4]:
25
Insira o dado vet2[ 5]:
26
Insira o dado vet2[ 6]:
27
```

```
Insira o dado vet2[ 7]:
28
Insira o dado vet2[ 8]:
29
Insira o dado vet2[ 9]:
30
Vetor resultante:
 11  21  12  22  13  23  14  24  15  25  16  26  17  27  18  28  19
29  20  30
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

VETORES – Exercício 5 – Solução

```c
#include <stdio.h>
int main(){
    int vet1[10], vet2[10], inter[20];
    int  i, j;
    for(i=0; i<10; i++){
        printf("Insira o dado vet1[%d]:\n", i);
        scanf("%d", &vet1[i]);
    }
    for(i=0; i<10; i++){
        printf("Insira o dado vet2[%d]:\n", i);
        scanf("%d", &vet2[i]);
    }
    j=0;
    for(i=0; i<10; i++){
        inter[j] = vet1[i];
        inter[j+1] = vet2[i];
        j += 2;
    }
    for(i=0; i<20; i++){
        printf("%d ", inter[i]);
    }
}
```

6) (Adaptado de ASCENCIO e CAMPOS, 2008) Faça um programa que carregue um vetor com oito números inteiros, calcule e mostre dois vetores resultantes. O primeiro vetor resultante deve conter os números positivos. O segundo vetor resultante deve conter os números negativos. Cada vetor resultante vai ter no máximo oito posições, sendo que nem todas devem obrigatoriamente ser utilizadas. Imprima o conteúdo dos vetores resultantes, sem que sejam impressos "lixos de memória".

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em

branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução – Exercício 6 - Caso de teste
Insira o dado vet1[ 0]:
1
Insira o dado vet1[ 1]:
-1
Insira o dado vet1[ 2]:
2
Insira o dado vet1[ 3]:
-2
Insira o dado vet1[ 4]:
3
Insira o dado vet1[ 5]:
-3
Insira o dado vet1[ 6]:
4
Insira o dado vet1[ 7]:
-4
Vetor de positivos:
 1  2  3  4
Vetor de negativos:
 -1  -2  -3  -4
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

VETORES – Exercício 6 – Solução

```c
#include <stdio.h>
int main(){
    int vet[8], po[8], ne[8];
    int i, j, k;
    j=0;
    k=0;
    for(i=0; i<8; i++){
        printf("Insira o dado vet[%d]:\n", i);
        scanf("%d", &vet[i]);
        if(vet[i] > 0){
            po[j] = vet[i];
            j++;
        }
        if(vet[i] < 0){
            ne[k] = vet[i];
            k++;
        }
    }
    printf("Vetor de positivos:\n");
    for(i=0; i<j; i++){
        printf("%d ", po[i]);
    }
```

```
23      printf("\nVetor de negativos:\n");
24      for(i=0; i<k; i++){
25          printf("%d ", ne[i]);
26      }
27  }
```

7) Desenvolva um algoritmo que preencha cada elemento de uma matriz 3x3 com o quadrado do valor do índice da linha mais o valor do índice da coluna daquela posição. Ao final, o algoritmo deve mostrar a matriz, na tela.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 7 - Caso de teste
0 1 2
1 2 3
4 5 6
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

```
MATRIZES - Exercício 7 - Solução
01  int main(){
02      int i, j;
03      int mat[3][3];
04      for(i=0; i<3; i++){
05          for(j=0; j<3; j++){
06              mat[i][j] = i*i + j;
07              printf("%d ", mat[i][j]);
08          }
09          printf("\n");
10      }
11  }
```

8) Desenvolva um algoritmo que preencha uma matriz numérica de dimensões 3x3. Depois de a matriz ter sido populada, o algoritmo deverá imprimir a matriz da seguinte forma: os dados da diagonal principal devem ser impressos normalmente e os dados fora da diagonal principal devem substituídos por zero.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens

geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`)

```
Exemplo de execução - Exercício 8 - Caso de teste
Insira o dado da posição  1,  1:
1
Insira o dado da posição  1,  2:
2
Insira o dado da posição  1,  3:
3
Insira o dado da posição  2,  1:
4
Insira o dado da posição  2,  2:
5
Insira o dado da posição  2,  3:
6
Insira o dado da posição  3,  1:
7
Insira o dado da posição  3,  2:
8
Insira o dado da posição  3,  3:
9
1 0 0
0 5 0
0 0 9
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

MATRIZES – Exercício 8 – Solução

```c
#include <stdio.h>
int main(){
    float mat[3][3];
    int i, j;

    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            printf("Insira o dado da posição %d, %d:\n",
i+1, j+1);
            scanf("%f", &mat[i][j]);
        }
    }
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            if(i==j){
                printf("%.1f ", mat[i][j]);
            } else {
                printf("0.0 ");
            }
        }
```

```
21          printf("\n");
22      }
23  }
```

9) Desenvolva um algoritmo que preencha uma matriz numérica de dimensões 3x3. Ao final, o algoritmo deve mostrar o somatório de todos os elementos da matriz, bem como a média aritmética entre todos os termos.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 9 - Caso de teste
Insira o dado da posição  1,  1:
1
Insira o dado da posição  1,  2:
2
Insira o dado da posição  1,  3:
3
Insira o dado da posição  2,  1:
4
Insira o dado da posição  2,  2:
5
Insira o dado da posição  2,  3:
6
Insira o dado da posição  3,  1:
7
Insira o dado da posição  3,  2:
8
Insira o dado da posição  3,  3:
9
Somatório:  45
Média:  5
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

MATRIZES – Exercício 9 – Solução

```
01  #include <stdio.h>
02  int main(){
03      float mat[3][3];
04      float soma, media;
05      int i, j;
06      soma = 0;
07      for(i=0; i<3; i++){
```

```
08          for(j=0; j<3; j++){
09              printf("Insira o dado da posição %d, %d:\n", i
    +1, j+1);
10              scanf("%f", &mat[i][j]);
11              soma += mat[i][j];
12          }
13      }
14      media = soma / 9;
15      printf("Somatório: %f.\n", soma);
16      printf("Média: %f.\n", media);
17  }
```

10) (Adaptado de ASCENCIO e CAMPOS, 2008) Faça um programa que carregue uma matriz 2x2, calcule e mostre uma matriz resultante que será a própria matriz digitada multiplicada pelo maior elemento da matriz.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 10 - Caso de teste
Insira o dado da posição  1,  1:
1
Insira o dado da posição  1,  2:
2
Insira o dado da posição  2,  1:
3
Insira o dado da posição  2,  2:
4
 4   8
 12  16
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

```
MATRIZES - Exercício 10 - Solução
01  #include <stdio.h>
02  int main(){
03      float mat[2][2];
04      float maior;
05      int i, j;
06
07      for(i=0; i<2; i++){
08          for(j=0; j<2; j++){
09              printf("Insira o dado da posição %d, %d:\n",
```

```
i+1, j+1);
            scanf("%f", &mat[i][j]);
        }
    }
    maior = mat[0][0];
    for(i=0; i<2; i++){
        for(j=0; j<2; j++){
            if(mat[i][j] > maior){
                maior = mat[i][j];
            }
        }
    }
    for(i=0; i<2; i++){
        for(j=0; j<2; j++){
            mat[i][j] = mat[i][j]*maior;
            printf("%f ", mat[i][j]);
        }
        printf("\n");
    }
}
```

11) Desenvolva um algoritmo que preencha uma matriz numérico de dimensões 3x3. Após preencher todo a matriz, o usuário deve inserir uma chave de busca X. Caso exista algum número igual a X, dentro da matriz, o algoritmo deve mostrar, na tela, os índices da linha e da coluna da posição na qual na qual X foi encontrado pela primeira vez. Caso contrário, o algoritmo deve se encerrar com uma única mensagem, dizendo "Chave não encontrada.".

Quer dicas de como seu algoritmo deveria funcionar? Observe os quadros abaixo, nos quais você encontra simulações da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 11 - Caso de teste 1
Insira o dado da posição  1,  1:
1
Insira o dado da posição  1,  2:
2
Insira o dado da posição  1,  3:
3
Insira o dado da posição  2,  1:
4
Insira o dado da posição  2,  2:
5
Insira o dado da posição  2,  3:
6
Insira o dado da posição  3,  1:
7
Insira o dado da posição  3,  2:
```

```
8
Insira o dado da posição  3,  3:
9
Insira a chave de busca:
5
Chave encontrada na linha:  2 coluna:  2
```

Exemplo de execução – Exercício 11 - Caso de teste 2

```
Insira o dado da posição  1,  1:
1
Insira o dado da posição  1,  2:
2
Insira o dado da posição  1,  3:
3
Insira o dado da posição  2,  1:
4
Insira o dado da posição  2,  2:
5
Insira o dado da posição  2,  3:
6
Insira o dado da posição  3,  1:
7
Insira o dado da posição  3,  2:
8
Insira o dado da posição  3,  3:
9
Insira a chave de busca:
10
Chave não encontrada.
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

MATRIZES – Exercício 11 – Solução

```c
#include <stdio.h>
#include <stdbool.h>
int main(){
    float X, mat[3][3];
    int i, j, lin, col;
    bool achou;

    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            printf("Insira o elemento da posição %d, %d:\n", i+1, j+1);
            scanf("%f", &mat[i][j]);
        }
    }
    printf("Insira a chave de busca:\n");
    scanf("%f", &X);
    achou = false;
```

```
17      i = 0;
18      while((i < 3) && (!achou)){
19          j = 0;
20          while((j < 3) && (!achou)){
21              if(mat[i][j] == X){
22                  achou = true;
23                  lin = i;
24                  col = j;
25              }
26              j++;
27          }
28          i++;
29      }
30      if(achou){
31          printf("Chave encontrada na linha %d, coluna %d.\n"
    , lin+1, col+1);
32      } else {
33          printf("Chave não encontrada.\n");
34      }
35  }
```

12) (Adaptado de ASCENCIO e CAMPOS, 2008) Faça um programa que carregue uma matriz 3 x 5 com números inteiros e some cada uma das linhas, armazenando o resultado das somas em um vetor. A seguir, multiplique cada elemento da matriz pela soma da respectiva linha daquele elemento e mostre a matriz resultante.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 12 - Caso de teste
Insira o dado da posição  1,  1:
1
Insira o dado da posição  1,  2:
2
Insira o dado da posição  1,  3:
3
Insira o dado da posição  1,  4:
4
Insira o dado da posição  1,  5:
5
Insira o dado da posição  2,  1:
10
Insira o dado da posição  2,  2:
20
Insira o dado da posição  2,  3:
30
Insira o dado da posição  2,  4:
```

```
40
Insira o dado da posição  2,  5:
50
Insira o dado da posição  3,  1:
11
Insira o dado da posição  3,  2:
12
Insira o dado da posição  3,  3:
13
Insira o dado da posição  3,  4:
14
Insira o dado da posição  3,  5:
15
 15    30    45    60    75
 1500  3000  4500  6000  7500
 715   780   845   910   975
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

MATRIZES – Exercício 12 – Solução

```c
#include <stdio.h>
int main(){
    float mat[3][5], vet[3];
    int i, j;
    for(i=0; i<3; i++){
        vet[i] = 0;
        for(j=0; j<5; j++){
            printf("Insira o dado da posição %d, %d:\n", i+1, j+1);
            scanf("%f", &mat[i][j]);
            vet[i] += mat[i][j];
        }
    }
    for(i=0; i<3; i++){
        for(j=0; j<5; j++){
            mat[i][j] = vet[i]*mat[i][j];
            printf("%f ", mat[i][j]);
        }
        printf("\n");
    }
}
```

13) (Adaptado de ASCENCIO e CAMPOS, 2008) Faça um programa que carregue uma matriz 10x3 com as três notas de dez alunos. Mostre um relatório com o número do aluno (número da linha) e a prova em que cada aluno obteve menor nota. Ao final do relatório, mostre quantos alunos tiveram menor nota na prova 1, quantos alunos tiveram menor nota na prova 2 e quantos alunos tiveram menor nota na prova 3.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução – Exercício 13 - Caso de teste
Insira a nota  1 do aluno  1:
10
Insira a nota  2 do aluno  1:
10
Insira a nota  3 do aluno  1:
10
Insira a nota  1 do aluno  2:
9
Insira a nota  2 do aluno  2:
9
Insira a nota  3 do aluno  2:
9
Insira a nota  1 do aluno  3:
8
Insira a nota  2 do aluno  3:
8
Insira a nota  3 do aluno  3:
8
Insira a nota  1 do aluno  4:
7
Insira a nota  2 do aluno  4:
8
Insira a nota  3 do aluno  4:
9
Insira a nota  1 do aluno  5:
9
Insira a nota  2 do aluno  5:
8
Insira a nota  3 do aluno  5:
7
Insira a nota  1 do aluno  6:
8
Insira a nota  2 do aluno  6:
7
Insira a nota  3 do aluno  6:
9
Insira a nota  1 do aluno  7:
9
Insira a nota  2 do aluno  7:
7
Insira a nota  3 do aluno  7:
8
Insira a nota  1 do aluno  8:
4
Insira a nota  2 do aluno  8:
```

```
5
Insira a nota  3 do aluno  8:
6
Insira a nota  1 do aluno  9:
6
Insira a nota  2 do aluno  9:
5
Insira a nota  3 do aluno  9:
4
Insira a nota  1 do aluno  10:
6
Insira a nota  2 do aluno  10:
4
Insira a nota  3 do aluno  10:
5
Aluno  1
Prova de menor nota:  1
Aluno  2
Prova de menor nota:  1
Aluno  3
Prova de menor nota:  1
Aluno  4
Prova de menor nota:  1
Aluno  5
Prova de menor nota:  3
Aluno  6
Prova de menor nota:  2
Aluno  7
Prova de menor nota:  2
Aluno  8
Prova de menor nota:  1
Aluno  9
Prova de menor nota:  3
Aluno  10
Prova de menor nota:  2
Menores notas na primeira prova:  5
Menores notas na segunda prova:  3
Menores notas na terceira prova:  2
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

MATRIZES – Exercício 13 – Solução

```c
#include <stdio.h>
int main(){
    float notas[10][3];
    float menor;
    int i, j, p_menor, q1, q2, q3;
    for(i=0; i<10; i++){
        for(j=0; j<3; j++){

```

```
09              printf("Insira a nota %d do aluno %d:\n",
    j+1, i+1);
10              scanf("%f", &notas[i][j]);
11          }
12      }
13      q1 = 0;
14      q2 = 0;
15      q3 = 0;
16      for(i=0; i<10; i++){
17          printf("Aluno %d\n", i+1);
18          menor = notas[i][0];
19          p_menor = 1;
20          for(j=0; j<3; j++){
21              if(notas[i][j] < menor){
22                  menor = notas[i][j];
23                  p_menor = j+1;
24              }
25          }
26          printf("Prova de menor nota: %d\n", p_menor);
27          switch(p_menor){
28              case 1:
29                  q1++;
30                  break;
31              case 2:
32                  q2++;
33                  break;
34              case 3:
35                  q3++;
36                  break;
37          }
38      }
39      printf("Menores notas na primeira prova: %d.\n", q1);
40      printf("Menores notas na segunda prova: %d.\n", q2);
41      printf("Menores notas na terceira prova: %d.\n", q3);
42  }
```

14) Faça um programa que seja capaz de armazenar os dados de três pessoas: nome, idade, peso e altura. Ao final, o algoritmo deve mostrar, na tela, o nome e a idade da primeira pessoa e o peso e altura da última pessoa.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 14 - Caso de teste
Digite o nome  1:Jairo
Insira a idade  1:20
Digite o peso  1:78
```

```
Insira a altura  1:1,78
Digite o nome  2:Camila
Insira a idade  2:25
Digite o peso  2:65
Insira a altura  2:1,67
Digite o nome  3:Lúcia
Insira a idade  3:78
Digite o peso  3:69
Insira a altura  3:1,67
Nome da primeira pessoa: Jairo
Idade da primeira pessoa:  20
Peso da última pessoa:  69
Altura da última pessoa:  1.67
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

REGISTROS - Exercício 14 - Solução

```c
#include <stdio.h>
struct pessoa{
    char nome[100];
    int idade;
    float peso;
    float altura;
};
int main(){
    struct pessoa pessoas[3];
    int i;
    for(i=0; i<3; i++){
        printf("Digite o nome %d:\n", i+1);
        scanf("%s", &pessoas[i].nome);
        printf("Insira a idade %d:\n", i+1);
        scanf("%d", &pessoas[i].idade);
        printf("Digite o peso %d:\n", i+1);
        scanf("%f", &pessoas[i].peso);
        printf("Insira a altura %d:\n", i+1);
        scanf("%f", &pessoas[i].altura);
    }
    printf("Nome da primeira pessoa: %s.\n", pessoas[0].no
me);
    printf("Idade da primeira pessoa: %d.\n", pessoas[0].i
dade);
    printf("Peso da última pessoa: %.3f.\n", pessoas[2].pe
so);
    printf("Altura da da última pessoa: %.2f.\n", pessoas[
2].altura);
}
```

15) Faça um programa que seja capaz de armazenar os dados de uma pessoa: nome, idade, peso e altura. Seu programa deve ser capaz de armazenar 5 pessoas. Ao final dos cadastros, o seu programa deve imprimir, na tela, todas as informações de todas as pessoas. Seu programa deve mostrar, também, o nome da pessoa mais magra, nome da pessoa mais baixa e a média das idades de todas as pessoas.

Quer dicas de como seu algoritmo deveria funcionar? Observe o quadro abaixo, no qual você encontra uma simulação da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução – Exercício 15 - Caso de teste
Digite o nome  1:José da Silva
Insira a idade  1:30
Digite o peso  1:89
Insira a altura  1:1,89
Digite o nome  2:Carlos Almeida
Insira a idade  2:40
Digite o peso  2:112
Insira a altura  2:1,80
Digite o nome  3:Camila Oliveira
Insira a idade  3:21
Digite o peso  3:56
Insira a altura  3:1,60
Digite o nome  4:Maria da Silva
Insira a idade  4:24
Digite o peso  4:58
Insira a altura  4:1,56
Digite o nome  5:Mário Lacerda
Insira a idade  5:56
Digite o peso  5:78
Insira a altura  5:1,55
Pessoa mais magra: Camila Oliveira
Pessoa mais alta: Mário Lacerda
Média das idades:  34.2
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

REGISTROS – Exercício 15 – Solução

```c
#include <stdio.h>
#include <string.h>
struct pessoa{
    char nome[100];
    int idade;
    float peso;
    float altura;
};
int main(){
```

```c
    struct pessoa pessoas[5];
    float p_magra, a_baixa, m_idade;
    char n_magra[100], n_baixa[100];
    int i;
    for(i=0; i<5; i++){
        printf("Digite o nome %d:\n", i+1);
        scanf("%s", &pessoas[i].nome);
        printf("Insira a idade %d:\n", i+1);
        scanf("%d", &pessoas[i].idade);
        printf("Digite o peso %d:\n", i+1);
        scanf("%f", &pessoas[i].peso);
        printf("Insira a altura %d:\n", i+1);
        scanf("%f", &pessoas[i].altura);
    }
    p_magra = pessoas[0].peso;
    a_baixa = pessoas[0].altura;
    m_idade = pessoas[0].idade;
    strcpy(n_magra, pessoas[0].nome);
    strcpy(n_baixa, pessoas[0].nome);
    for(i=1; i<5; i++){
        if(pessoas[i].peso < p_magra){
            p_magra = pessoas[i].peso;
            strcpy(n_magra, pessoas[i].nome);
        }
        if(pessoas[i].altura < a_baixa){
            a_baixa = pessoas[i].altura;
            strcpy(n_baixa, pessoas[i].nome);
        }
        m_idade += pessoas[i].idade;
    }
    m_idade /= 5;
    printf("Pessoa mais magra: %s.\n", n_magra);
    printf("Pessoa mais baixa: %s.\n", n_baixa);
    printf("Média das idades: %.2f.\n", m_idade);
}
```

16) Faça um programa que seja capaz de armazenar os dados de um produto: código, nome, valor e quantidade. Seu programa deve ser capaz de armazenar 5 produtos. Ao final dos cadastros, o usuário deve inserir o código de um produto e o seu programa deve imprimir, na tela, as informações daquele produto. Caso o produto não se encontre cadastrado, deve-se informar ao usuário a seguinte mensagem: "código não encontrado".

Quer dicas de como seu algoritmo deveria funcionar? Observe os quadros abaixos, nos quais você encontra simulações da execução do algoritmo. Textos que estão em azul são mensagens geradas pela máquina (operações de saída – `printf` ou `puts`). Já os textos que estão em branco correspondem a dados informados pelo usuário (operações de entrada – `scanf`, `gets` ou `fgets`).

```
Exemplo de execução - Exercício 16 - Caso de teste 1
```

```
Insira o código do produto  1:
10
Insira o nome do produto  1:
Arroz
Insira o valor do produto  1:
2,50
Insira a quantidade do produto  1:
1
Insira o código do produto  2:
20
Insira o nome do produto  2:
Feijão
Insira o valor do produto  2:
5,60
Insira a quantidade do produto  2:
4
Insira o código do produto  3:
30
Insira o nome do produto  3:
Farofa
Insira o valor do produto  3:
1,20
Insira a quantidade do produto  3:
2
Insira o código do produto  4:
40
Insira o nome do produto  4:
Sabonete
Insira o valor do produto  4:
0,98
Insira a quantidade do produto  4:
10
Insira o código do produto  5:
50
Insira o nome do produto  5:
Picanha
Insira o valor do produto  5:
60,00
Insira a quantidade do produto  5:
3
Insira a chave de busca:
30
Nome: Sabonete
Valor:  0.98
Quantidade:  10
```

Exemplo de execução – Exercício 16 - Caso de teste 2

```
Insira o código do produto  1:
10
Insira o nome do produto  1:
Arroz
Insira o valor do produto  1:
2,50
```

```
Insira a quantidade do produto  1:
1
Insira o código do produto  2:
20
Insira o nome do produto  2:
Feijão
Insira o valor do produto  2:
5,60
Insira a quantidade do produto  2:
4
Insira o código do produto  3:
30
Insira o nome do produto  3:
Farofa
Insira o valor do produto  3:
1,20
Insira a quantidade do produto  3:
2
Insira o código do produto  4:
40
Insira o nome do produto  4:
Sabonete
Insira o valor do produto  4:
0,98
Insira a quantidade do produto  4:
10
Insira o código do produto  5:
50
Insira o nome do produto  5:
Picanha
Insira o valor do produto  5:
60,00
Insira a quantidade do produto  5:
3
Insira a chave de busca:
99
Código não encontrado
```

Antes de verificar o gabarito desta questão, limpe sua consciência: Dedique ao MENOS 10 MINUTOS do seu tempo tentando resolver esse exercício. Caso considere que já tenha tentado o suficiente, segue aí uma solução que funciona:

REGISTROS - Exercício 16 - Solução

```c
#include <stdio.h>
#include <stdbool.h>
struct produto {
    int codigo;
    char nome[100];
    float valor;
    float qtde;
};
int main(){
```

```
10      struct produto lista_prod[5];
11      int i, X, p;
12      bool achou;
13
14      for(i=0; i<5; i++){
15          printf("Insira o código do produto %d:\n", i+1);
16          scanf("%d", &lista_prod[i].codigo);
17          printf("Insira o nome do produto %d:\n", i+1);
18          scanf("%s", lista_prod[i].nome);
19          printf("Insira o valor do produto %d:\n", i+1);
20          scanf("%f", &lista_prod[i].valor);
21          printf("Insira a quantidade do produto %d:\n", i+1
    );
22          scanf("%f", &lista_prod[i].qtde);
23      }
24      printf("Insira a chave de busca:\n");
25      scanf("%d", &X);
26      achou = false;
27      i = 0;
28      while((i < 5) && (!achou)){
29          if(lista_prod[i].codigo == X){
30              achou = true;
31              p = i+1;
32          }
33          i++;
34      }
35      if(achou){
36          printf("Nome: %s.\n", lista_prod[i].nome);
37          printf("Valor: %.2f.\n", lista_prod[i].valor);
38          printf("Quantidade: %.3f.\n", lista_prod[i].qtde);
39      } else {
40          printf("Produto não encontrado.\n");
41      }
42  }
```