Vetores Numéricos

DCC 119 – Algoritmos





Programa que lê as notas de 4 alunos e calcula a sua média.

```
int main()
  int i;
  float nota, media, soma = 0;
  for (i = 0; i < 4; i++)
    printf("Digite uma nota:");
    scanf ("%f", &nota);
    soma += nota;
 media = soma / 4;
  printf("Media = %f", media);
  return 0;
```



Programa que lê as notas de 4 alunos e calcula a sua média.

```
int main()
  int i;
  float nota, media, soma = 0;
  for (i = 0; i < 4; i++)
    printf("Digite uma nota:");
    scanf ("%f", &nota);
    soma += nota;
 media = soma / 4;
 printf("Media = %f", media);
  return 0;
```

Assuma que seja necessário imprimir, no final do programa, o número de notas acima da média calculada.



Programa que lê as notas de 4 alunos e calcula a sua média.

```
int main()
  int i;
  float nota, media, soma = 0;
  for (i = 0; i < 4; i++)
    printf("Digite uma nota:");
    scanf ("%f", &nota);
    soma += nota;
 media = soma / 4;
 printf("Media = %f", media);
  return 0;
```

Assuma que seja necessário imprimir, no final do programa, o número de notas acima da média calculada.

Que modificações são necessárias no programa?



Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
int main()
  int i;
  float nota, media, soma = 0;
  for (i = 0; i < 4; i++)
   printf("Digite uma nota:");
    scanf ("%f", &nota);
    soma += nota;
 media = soma / 4;
 printf("Media = %f", media);
  return 0;
```

É necessário verificar que notas estão acima da média.

Isso só pode ser feito, após o cálculo da média!



Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
int main()
  int i;
  float nota, media, soma = 0;
  for (i = 0; i < 4; i++)
    printf("Digite uma nota:");
    scanf ("%f", &nota);
    soma += nota;
 media = soma / 4;
 printf("Media = %f", media);
  return 0;
```

Duas opções:

- Ler o valor de cada nota outra vez
- Ler apenas uma vez cada valor, armazenando cada nota em uma variável distinta



Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
// Opcao 1: le duas vezes cada valor
int main()
  int i, cont=0;
  float nota, media, soma = 0;
  for (i = 0; i < 4; i++)
   printf("Digite uma nota:");
   scanf ("%f", &nota);
   soma += nota;
 media = soma / 4:
 printf("Digite tudo de novo!");
  for (i = 0; i < 4; i++) {
   printf("Digite uma nota:");
   scanf ("%f", &nota);
   if( nota > media )
   cont++;
 printf ("Media = %f", media);
 printf("%d notas acima", cont);
  return 0;
```

```
//Opcao 2: uma variavel por nota
int main()
 int i, cont=0;
 float n1, n2, n3, n4;
 float media, soma = 0;
 printf("Digite as notas:");
 scanf ("%f %f %f %f",
         &n1, &n2, &n3, &n4);
 media = (n1 + n2 + n3 + n4) /
 printf("Media = %f", media);
 if( n1 > media ) cont++;
 if(n2 > media) cont++;
 if(n3 > media) cont++;
 if(n4 > media) cont++;
 printf("%d notas acima", cont);
 return 0;
```



Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
// Opcao 1: le duas vezes cada valor
int main()
  int i, cont=0;
  float nota, media, soma = 0;
  for (i = 0; i < 4; i++)
   printf("Digite uma nota:");
   scanf ("%f", &nota);
   soma += nota;
 media = soma / 4;
  printf("Digite tudo de novo!");
  for (i = 0; i < 4; i++) {
   printf("Digite uma nota:");
   scanf ("%f", &nota);
   if( nota > media )
   cont++;
 printf ("Media = %f", media);
 printf("%d notas acima", cont);
  return 0;
```

```
//Opcao 2: uma variavel por nota
int main()
 int i, cont=0;
 float n1, n2, n3, n4;
 float media, soma = 0;
 printf("Digite as notas:");
 scanf ("%f %f %f %f",
         &n1, &n2, &n3, &n4);
 media = (n1 + n2 + n3 + n4) /
 printf("Media = %f", media);
 if( n1 > media ) cont++;
 if(n2 > media) cont++;
 if(n3 > media) cont++;
 if( n4 > media ) cont++;
```

Na opção 1, o usuário vai ter o trabalho de redigitar os valores.



Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
// Opcao 1: le duas vezes cada valor
int main()
  int i, cont=0;
  float nota, media, soma = 0;
  for (i = 0; i < 4; i++)
   printf("Digite uma nota:");
    scanf ("%f", &nota);
   soma += nota;
 media = soma / 4:
 printf("Digite tudo de novo!");
  for (i = 0; i < 4; i++) {
   printf("Digite uma nota:");
    scanf ("%f", &nota);
    if( nota > media )
   cont++;
  Na opção 2, o programador
  precisa repetir os comandos para
```

cada variável.

```
//Opcao 2: uma variavel por nota
int main()
 int i, cont=0;
  float n1, n2, n3, n4;
 float media, soma = 0;
 printf("Digite as notas:");
 scanf ("%f %f %f %f",
          &n1, &n2, &n3, &n4);
 media = (n1 + n2 + n3 + n4) /
 printf("Media = %f", media);
 if( n1 > media ) cont++;
 if(n2 > media) cont++;
 if( n3 > media ) cont++;
 if( n4 > media ) cont++:
 printf("%d notas acima", cont);
 return 0;
```



Programa que lê notas de 4 alunos, calcula sua média e imprime o número de notas acima da média.

```
// Opcao 1: le duas vezes cada valor
int main()
  int i, cont=0;
  float nota, media, soma = 0;
  for (i = 0; i < 4; i++) {
   printf("Digite uma nota:");
    scanf ("%f", &nota);
   soma += nota;
 media = soma / 4:
 printf("Digite tudo de novo!");
  for (i = 0; i < 4; i++)
   printf("Digite uma nota:");
    scanf ("%f", &nota);
   if( nota > media )
   cont++;
 printf ("Media = %f", media);
 print
```

```
//Opcao 2: uma variavel por nota
int main()
 int i, cont=0;
 float n1, n2, n3, n4;
 float media, soma = 0;
 printf("Digite as notas:");
 scanf ("%f %f %f %f",
         &n1, &n2, &n3, &n4);
 media = (n1 + n2 + n3 + n4) /
 printf("Media = %f", media);
 if( n1 > media ) cont++;
 if(n2 > media) cont++;
 if(n3 > media) cont++;
 if( n4 > media ) cont++;
```

ima", cont);



```
// Opcao 1: le duas vezes cada valor
int main()
 int i, cont=0;
 float nota, media, soma = 0;
 for (i = 0; i < 4; i++) {
   printf("Digite uma nota:");
   scanf ("%f", &nota);
   soma += nota;
 printf("Digite tudo de novo!");
 for (i = 0; i < 4; i++)
   printf("Digite uma nota:");
   scanf ("%f", &nota);
   if nota > media )
   cont ++;
 printf("Media = %f", media);
    O usuário não vai gostar
    de redigitar 10 valores
    (ou 50, ou 150, ou 1000)
```

```
//Opcao 2: uma variavel por nota
int main()
 int i, cont=0;
 float n1, n2, n3, n4, n5,
       n6, n7, n8, n9, n10;
 float media, soma = 0;
 printf("Digite as notas:");
 scanf ("%f %f %f %f %f",
        &n1, &n2, &n3, &n4, &n5);
 scanf ("%f %f %f %f %f",
        &n6, &n7, &n8, &n9, &n10);
 media = (n1 + n2 + n3 + n4 + n5)
        + n6 + n7 + n8 + n9 + n10) / 10;
 printf("Media = %f", media);
 if( n1 > media ) cont++;
 if(n2 > media) cont++;
 if(n3 > media) cont++;
 if(n4 > media) cont++;
 if(n5 > media) cont++;
 if( n6 > media ) cont++;
 if(n7 > media) cont++;
 if(n8 > media) cont++;
 if(n9 > media) cont++;
 if(n10 > media) cont++;
 printf("%d notas acima", cont);
 return 0;
```



```
// Opcao 1: le duas vezes cada valor
int main()
  int i, cont=0;
  float nota, media, soma = 0;
  for (i = 0; i < 4; i++) {
    printf("Digite uma nota:");
    scanf ("%f", &nota);
    soma += nota;
  media = soma / 4;
  printf("Digite tudo de novo!");
  for (i = 0; i < 4; i++)
    printf("Digite uma nota:");
    scanf ("%f", &nota);
    if( nota > media )
    cont++;
 printf("Media = %f", media);
  printf("%d notas acima", cont);
```

O programador pode cometer muitos erros ao replicar 10 variáveis (ou 50, ou 150...)

```
//Opcao 2: uma variavel por nota
int main()
 int i, cont=0;
 float n1, n2, n3, n4, n5,
       n6, n7, n8, n9, n10;
 float media, soma = 0;
 printf("Digite as notas:");
 scanf ("%f %f %f %f %f",
        &n1, &n2, &n3, &n4, &n5);
 scanf ("%f %f %f %f %f",
        &n6, &n7, &n8, &n9, &n10);
 media = (n1 + n2 + n3 + n4 + n5)
        + n6 + n7 + n8 + n9 + n10
 if( n1 > media ) cont++;
 if(n2 > media) cont++;
 if(n3 > media) cont++;
 if(n4 > media) cont++;
 if(n5 > media) cont++;
 if( n6 > media ) cont++;
 if(n7 > media) cont++;
 if(n8 > media) cont++;
 if(n9 > media) cont++;
 if( n10 > media ) cont++;
 printf("%d notas acima", cont);
 return 0;
```



```
//Opcao 2: uma variavel por nota
// Opcao 1: le duas vezes cada valor
                                           int main()
int main()
                                             int i, cont=0;
  int i, cont=0;
                                             float n1, n2, n3, n4, n5,
 float nota, media, soma = 0;
                                                  n6, n7, n8, n9, n10;
 for (i = 0; i < 4; i++) {
                                             float media, soma = 0;
   printf("Digite uma nota:");
                                             printf("Digite as notas:");
   scanf ("%f",
                                                         %f %f %f",
                       Vetores permitem
                                                        n2, &n3, &n4, &n5);
   soma += nota;
                                                         %f %f %f",
                          resolver esse
                                                        n7, &n8, &n9, &n10);
 media = soma /
                                                         n2 + n3 + n4 + n5
 printf("Digite
                                                         n7 + n8 + n9 + n10) / 10;
                     problema de forma
 for (i = 0; i <
                                                        = %f", media);
   printf ("Digit
                                                        a ) cont++;
                          conveniente.
   scanf ("%f",
                                                        a ) cont++;
   if( nota > me
                                                        a ) cont++;
                                             if(n4 > media) cont++;
   cont++;
                                             if(n5 > media) cont++;
                                             if( n6 > media ) cont++;
 printf ("Media = %f", media);
                                             if(n7 > media) cont++;
 printf("%d notas acima", cont);
                                             if(n8 > media) cont++;
 return 0;
                                             if(n9 > media) cont++;
                                             if(n10 > media) cont++;
                                             printf("%d notas acima", cont);
                                             return 0;
```

Um vetor...



- é uma estrutura que permite armazenar uma sequência de dados de um mesmo tipo.
- permite que dados sejam armazenados e estruturados de forma simples.
- permite que cada um dos dados armazenados seja acessado diretamente.

Declarando um vetor



- Ao criar um vetor, precisamos informar o tamanho deste, isto é, precisamos informar quantos valores serão armazenados.
- Ao saber o número de valores, o compilador separa o espaço necessário na memória.
- A sintaxe para criação de um vetor é:

```
tipo nomeDoVetor[ TAMANHO ];

Exemplo:
    dados
int dados[ 5 ];
```

Declarando um vetor



- Ao reservar a memória para o programa, antes de iniciar a execução, é necessário saber quanto espaço de memória precisa ser separado.
- O tamanho deve ser especificado através de um valor constante.

```
int main()
  int matricula[50]; //matricula de cada aluno
  float notaP1[50]; //nota de cada aluno na la prova
  float notaP2[50]; //nota de cada aluno na 2a prova
  float notaP3[50]; //nota de cada aluno na 3a prova
  float mediaTurma[3];//media das notas em cada prova
```

Acessando elementos



Não é possível fazer uma operação com todo o conjunto dos dados armazenados no vetor de uma só vez!

- Cada dado precisa ser acessado individualmente, seja para consultar ou para alterar seu valor.
- Para permitir o acesso a cada elemento de um vetor, a sequência de elementos é numerada. Assim, um elemento específico pode ser identificado pelo seu índice.

	_	
	0	?
	1	?
4	2	?
	2 3 4	?
	4	?

Acessando elementos



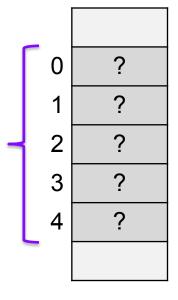
- O primeiro elemento do vetor sempre recebe o índice 0 (zero).
- Logo, em um vetor com N elementos, os índices são numerados de 0 a N-1.
- O acesso ao elemento é feito com

nomeDoVetor [indice]

0	?
1	?
2	?
3	?
4	?
	0 1 2 3 4



```
int main() {
   int potencias[5]; //vetor com 5 elementos
```





```
int main(){
   int potencias[5]; //vetor com 5 elementos
   potencias[0] = 1;
```

O primeiro elemento (índice 0) do vetor tem seu valor modificado.
O comando de atribuição altera somente o elemento correspondente ao índice indicado.

0	1
1	?
2	?
2 3 4	?
4	?



```
int main(){
   int potencias[5]; //vetor com 5 elementos
   potencias[0] = 1;
   scanf("%d", &potencias[1]);
```

O elemento do vetor na posição 1 tem seu valor modificado.

A função **scanf** armazena o valor digitado pelo usuário (por exemplo, 3) no elemento correspondente ao índice indicado.

	0	1
	1	3
_	2	?
	2 3 4	?
	4	?



```
int main(){
  int potencias[5]; //vetor com 5 elementos
  potencias[0] = 1;
  scanf("%d", &potencias[1]);

  potencias[2] = pow(potencias[1],2);
```

O terceiro elemento do vetor tem seu valor modificado em função do segundo.

	0	1
	0 1	3
_	2	9
	2 3 4	?
	4	?



```
int main(){
   int potencias[5]; //vetor com 5 elementos
   potencias[0] = 1;
   scanf("%d", &potencias[1]);
   potencias[2] = pow(potencias[1], 2);
   potencias[3] = pow(potencias[1],3);
   potencias[4] = pow(potencias[1],4);
     Da mesma forma, os elementos
     seguintes são modificados.
```



```
int main(){
   int potencias[5]; //vetor com 5 elementos
   potencias[0] = 1;
   scanf("%d", &potencias[1]);
   potencias[2] = pow(potencias[1],2);
   potencias [3] = pow(potencias[1], 3);
   potencias[4] = pow(potencias[1], 4);
   printf("%d^%d=%d", potencias[1],
           4, potencias[4]);
                                               27
     Da mesma forma, os elementos
     podem ser acessados durante a
                                              81
     impressão.
```

Exercícios



```
vet 1 2 4 7 4 2 8 9 0 6 5 4 3
```

- 1) Quais são os elementos do vetor referenciados por: vet[3]? vet[0]? vet[13]?
- 2) Quais as modificações feitas no vetor em:

```
vet[1] = vet[3] * vet[1] ?
vet[10] = pow( vet[2], 2 ) + vet[0]?
vet[9] = vet[3] < vet[4] ?
vet[6] = vet[vet[5]] ?</pre>
```

3) Qual é a diferença entre os números "3" das duas instruções abaixo ?

```
int vet[3];
vet[3] = 5;
```

Acessando elementos



O acesso ao elemento é feito com

nomeDoVetor [indice]

 Neste caso, o índice não precisa ser um valor constante: pode ser um número, uma variável ou uma expressão, desde que o valor resultante esteja no intervalo entre 0 e N-1.

	0	?
	1	?
4	2	?
	2 3 4	?
	4	?



```
int main(){
   int i, potencias[15]; //vetor com 15 elementos
   potencias[0] = 1;
   scanf("%d", &potencias[1]);
   for( i=2; i<15; i++) {
       potencias[i] = pow(potencias[1],i);
       printf("%d^%d=%d", potencias[1],
                i, potercias[i]);
            Com a repetição, o valor da variável i
            muda. Para cada valor de i no interior
            da repetição, o elemento na posição i
            do vetor é modificado.
                                                   27
                                                   81
                                                   243
```

720

Definindo constantes



- A diretiva de compilação #define permite que uma constante seja definida no código.
- A utilização de constantes ajuda na correção e manutenção do programa.

```
#define NUM_ALUNOS 50
int main()
{
  int matricula[NUM_ALUNOS]; //matricula de cada aluno
  float notaP1[NUM_ALUNOS]; //nota na la prova
  float notaP2[NUM_ALUNOS]; //nota na 2a prova
  float notaP3[NUM_ALUNOS]; //nota na 3a prova
  float mediaTurma[3]; //media das notas em cada prova
  ...
```

Com o uso de vetores, o programa do início da aula pode ser implementado de forma simples.

```
#define NUM ALUNOS 10
int main()
  int i, cont=0;
  float nota[NUM ALUNOS];
  float media, soma = 0;
  for (i = 0; i < NUM ALUNOS; i++)</pre>
    printf("Digite uma nota:");
    scanf ("%f", &nota[i]);
    soma += nota[i];
  media = soma / NUM ALUNOS;
  for (i = 0; i < NUM ALUNOS; i++)
    if( nota[i] > media ) cont++;
 printf("Media = %f", media);
  printf("%d notas acima", cont);
  return 0;
```

```
int main()
 int i, cont=0;
 float n1, n2, n3, n4, n5,
       n6, n7, n8, n9, n10;
 float media, soma = 0;
 printf("Digite as notas:");
 scanf ("%f %f %f %f %f",
        &n1, &n2, &n3, &n4, &n5);
 scanf ("%f %f %f %f %f",
        &n6, &n7, &n8, &n9, &n10);
 media = (n1 + n2 + n3 + n4 + n5)
        + n6 + n7 + n8 + n9 + n10
        / 10;
 printf("Media = %f", media);
 if( n1 > media ) cont++;
 if(n2 > media) cont++;
 if(n3 > media) cont++;
 if( n4 > media ) cont++;
 if(n5 > media) cont++;
 if( n6 > media ) cont++;
 if(n7 > media) cont++;
 if( n8 > media ) cont++;
 if(n9 > media) cont++;
 if(n10 > media) cont++;
 printf("%d notas acima", cont);
 return 0;
```

```
#define NUM ALUNOS 10
int main()
  int i, cont=0;
  float nota[NUM ALUNOS];
  float media, soma = 0;
  for (i = 0; i < NUM ALUNOS; i++)</pre>
    printf("Digite uma nota:");
    scanf ("%f", &nota[i]);
    soma += nota[i];
 media = soma / NUM ALUNOS;
  for (i = 0; i < NUM ALUNOS; i++)
    if( nota[i] > media ) cont++;
 printf("Media = %f", media);
 printf("%d notas acima", cont);
  return 0;
```

Os inconvenientes anteriores foram resolvidos:

- Não há necessidade de o usuário redigitar os valores.
- O programador pode utilizar laços para acessar cada posição, ao invés de inserir uma linha de código específica para cada valor de uma sequência.

```
#define NUM ALUNOS 10
int main()
  int i, cont=0;
  float nota[NUM ALUNOS];
  float media, soma = 0;
  for (i = 0; i < NUM ALUNOS; i++)</pre>
    printf("Digite uma nota:");
    scanf ("%f", &nota[i]);
    soma += nota[i];
  media = soma / NUM ALUNOS;
  for (i = 0; i < NUM ALUNOS; i++)</pre>
    if( nota[i] > media ) cont++;
  printf("Media = %f", media);
  printf("%d notas acima", cont);
  return 0;
```

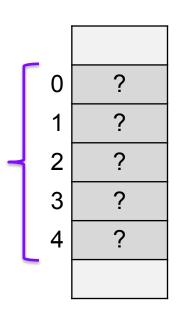
O uso da diretiva #define para especificar o tamanho do vetor facilita o trabalho do programador em caso de necessidade de alteração do tamanho do vetor.

```
#define NUM ALUNOS 10
int main()
  int i, cont=0;
  float nota[NUM ALUNOS];
  float media, soma = 0;
  for (i = 0; i < NUM ALUNOS; i++)
    printf("Digite uma nota:");
    scanf ("%f", &nota[i]);
    soma += nota[i];
 media = soma / NUM ALUNOS;
  for (i = 0; i < NUM ALUNOS; i++)
    if( nota[i] > media ) cont++;
 printf("Media = %f", media);
  printf("%d notas acima", cont);
  return 0;
```



- Assim como ocorre com variáveis, vetores não são automaticamente inicializados na sua declaração, ou seja, após a criação de um vetor, todas as suas posições contém "lixo".
- Para inicializar um vetor, normalmente são utilizadas repetições com comandos de leitura ou atribuição para cada uma das posições.

```
int main()
{
  int i, dados[5];
  // Inicializa todo o vetor com zero
  for( i=0; i<5; i++ )
    dados[i] = 0;
  ...
}</pre>
```





- Assim como ocorre com variáveis, vetores não são automaticamente inicializados na sua declaração, ou seja, após a criação de um vetor, todas as suas posições contém "lixo".
- Para inicializar um vetor, normalmente são utilizadas repetições com comandos de leitura ou atribuição para cada uma das posições.

```
int main()
{
   int i, dados[5];
   //Inicializa com dados lidos do teclado
   for( i=0; i<5; i++ )
      scanf("%d", &dados[i]);
   ...
}</pre>
```



- Outra forma de inicializar vetores com valores constantes é indicar valores para cada posição no momento da declaração do vetor.
- Em geral, esta alternativa só é utilizada para vetores pequenos.

```
int main()
{
    // Declara e inicializa o vetor
    int dados[5]={0,0,0,0,0};

    int main()
    {
        int i, dados[5];
        // Inicializa todo o vetor com zero
        for( i=0; i<5; i++ )
            dados[i] = 0;
        ...</pre>
```



Outro exemplo

```
float valores[5] = {3.2, 5.6, -15.1, 3.0, 2.5};
```

Qual o resultado?

```
float valores[5] = {0.0};
```

Erro de sintaxe

```
float valores[3] = {1.0, 2.0, 3.0, 4.0};
```



O programa a seguir, usa o comando **for** para inicializar com zeros os elementos de um array inteiro **n** de 10 elementos e o imprime sob a forma de uma tabela.

```
#include <stdio.h>
int main()
  int n[10], i;
  for (i=0; i<10; i++)
   n[i] = 0;
  printf("%s%13s\n", "Elemento", "Valor");
  for (i=0; i<10; i++)
   printf("%5d %13d\n", i , n[i]);
  return 0;
```

Elemento	Valor
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0



O programa abaixo inicializa os dez elementos de um array **s** com os valores: 2, 4, 6, ..., 20 e imprime o vetor em um formato de tabela.

```
#include <stdio.h>
#define TAMANHO 10
int main()
  int s[TAMANHO], j;
  for (j=0; j \le TAMANHO - 1; j++)
    s[j] = 2 + 2*j;
 printf("%s%13s\n", "Elemento", "Valor");
  for (j=0; j \le TAMANHO - 1; j++)
   printf("%8d %13d\n", j, s[j]);
  return 0;
```

Elemento	Valor
0	2
1	4
2	6
3	8
4	10
5	12
6	14
7	16
8	18
9	20



O programa abaixo cria um vetor com um valor máximo de tamanho e utiliza apenas uma parte das posições.

```
#include <stdio.h>
#define MAX ALUNOS 100
int main()
  int notas[MAX ALUNOS], j, numAlunos;
 printf("Numero de alunos: ");
  scanf("%d", &numAlunos);
  for (j=0; j < numAlunos; j++)
    printf("Nota do %d. aluno: ", j+1);
    scanf("%d", &notas[j]);
 printf("Fim da leitura das notas");
  /* */
  return 0;
```

```
Numero de alunos: 5
Nota do 1. aluno: 7
Nota do 2. aluno: 9
Nota do 3. aluno: 10
Nota do 4. aluno: 4
Nota do 5. aluno: 7
Fim da leitura das notas
```



O programa abaixo cria um vetor onde cada posição exerce a função de um contador.

```
#include <stdio.h>
#define NUM CANDIDATOS 5
#define NUM VOTOS 25
int main()
  int contador[NUM CANDIDATOS], i, cand;
  for( i=0; i < NUM CANDIDATOS; i++ )</pre>
   contador[i] = 0;
  for( i=0; i < NUM VOTOS; i++ )</pre>
    printf("Entre com %d. voto:",i+1);
    scanf("%d", &cand);
    contador[cand]++;
  for ( i=0; i < NUM CANDIDATOS; i++ )</pre>
   printf("\nCandidato %d: %d votos",
          i, contador[i]);
  return 0;
```

```
Entre com o 1. voto: 3
Entre com o 2. voto: 1
Entre com o 3. voto: 1
Entre com o 4. voto: 2
Entre com o 5. voto: 1
Entre com o 22. voto: 4
Entre com o 23. voto: 0
Entre com o 24. voto: 0
Candidato 0: 5 votos
Candidato 1: 11 votos
Candidato 2: 3 votos
Candidato 3: 2 votos
Candidato 4: 4 votos
```

Exercícios



- 4) Faça um programa que crie três vetores, preencha dois deles com valores lidos do teclado e, ao final, preencha o terceiro vetor armazenando, em cada índice, a soma dos elementos com este mesmo índice nos outros vetores.
- 5) Faça um programa que leia, via teclado, 20 valores do tipo inteiro, determine qual o menor valor existente no vetor e imprima este valor e seu índice no vetor.
- 6) Faça um programa que armazene, no vetor v, 10 valores reais lidos do teclado e calcule sua média. O programa deve copiar os elementos acima da média para o vetor v1 e os abaixo da média para o vetor v2. Ao final, o programa deve imprimir v1 e v2.

Exercícios



7) Um professor resolveu avaliar os trabalhos em grupo de seguinte forma: em cada grupo de 8 alunos, cada estudante dá uma nota de 1 a 10 para os outros 7 integrantes do grupo.

Faça um programa que calcule as notas de um grupo lendo do teclado as notas atribuídas por cada integrante e imprimindo a média das notas que cada aluno recebeu.

Dica: Use um vetor que armazene em cada posição a soma das notas recebidas por cada estudante. Observe que não é necessário armazenar cada nota de um aluno, mas apenas a soma de suas notas.

Exercícios



DESAFIO) Faça um programa com três vetores de números reais v1, v2 e v3 de tamanhos N, N e 2N, respectivamente. Preencha v1 e v2 com valores do teclado, assumindo que cada vetor será preenchido por uma sequência crescente.

Seu programa deverá copiar os elementos de v1 e v2 para v3, de forma que, ao final do processamento, o vetor v3 contenha todos os elementos de v1 e v2 em ordem crescente.

Observação importante: após o preenchimento do vetor, cada vetor só poderá ser percorrido apenas uma vez.



Vetores podem passados como parâmetros para funções!



```
#include <stdio.h>
#define TAMANHO 10
void preencheVetor (int vet[], int tam) {
  int i:
  for (i = 0; i < tam; i++) {
    printf ("Informe o valor da posicao %d: ", i);
    scanf ("%d", &vet[i]);
void imprimeVetor (int vet[], int tam) {
  int i:
  for (i = 0; i < tam; i++)
   printf("Posicao %d => %d\n", i, vet[i]);
int main() {
```

preencheVetor (vetorB, 2*TA) imprimeVetor(vetorA, TAMAN imprimeVetor(vetorB, 2*TAM return 0;

Na **definição da função**, os int vetorA[TAMANHO], vetor colchetes servem para indicar preencheVetor (vetorA, TAMA que o parâmetro em questão é um vetor.



```
#include <stdio.h>
#define TAMANHO 10
void preencheVetor (int vet[], int tam) {
  int i:
  for (i = 0; i < tam; i++) {
    printf ("Informe o valor da posicao %d: ", i);
    scanf ("%d", &vet[i]);
void imprimeVetor (int vet[], int tam) {
  int i:
  for (i = 0; i < tam; i++)
int main() {
  int vetorA[TAMANHO], vetori
 preencheVetor (vetorA, TAMA) Isto permite que a função
  preencheVetor (vetorB, 2*TAI
  imprimeVetor(vetorA, TAMAN
  imprimeVetor (vetorB, 2*TAM de diferentes tamanhos.
  return 0:
```

printf("Posicao %d => %d Normalmente, um parâmetro adicional é utilizado para indicar o tamanho do vetor. possa ser utilizada por vetores



```
#include <stdio.h>
#define TAMANHO 10
void preencheVetor (int vet[], int tam) {
  int i:
                             Na chamada da função,
  for (i = 0; i < tam; i++)
    printf ("Informe o valor
                             quando um vetor é passado
    scanf ("%d", &vet[i]);
                             por parâmetro, os colchetes
                             não são usados. A chamada
void imprimeVetor (int vet[],
                             deve ter apenas o nome do
  int i:
                             vetor.
  for (i = 0; i < tam; i++)
   printf("Posicao %d => %d\n", i, vet[i]);
int main() {
  int vetorA[TAMANHO], vetorB[2*TAMANHO];
 preencheVetor (vetorA, TAMANHO);
 preencheVetor (vetorB, 2*TAMANHO);
  imprimeVetor(vetorA, TAMANHO);
  imprimeVetor(vetorB, 2*TAMANHO);
  return 0;
```



```
#include <stdio.h>
#define TAMANHO 10
void preencheVetor (int vet[], int tam) {
  int i:
                              Observe que a mesma função
  for (i = 0; i < tam; i++)</pre>
    printf ("Informe o valor
                              pode ser utilizada por vetores
    scanf ("%d", &vet[i]);
                              de diferentes tamanhos.
void imprimeVetor (int vet[], int tam) {
  int i:
  for (i = 0; i < tam; i++)
    printf("Posicao %d => %d\n", i, vet[i]);
int main() {
  int vetorA[TAMANHO], vetorB[2*TAMANHO];
 preencheVetor(vetorA, TAMANHO);
  preencheVetor(vetorB, 2*TAMANHO);
  imprimeVetor(vetorA, TAMANHO);
  imprimeVetor(vetorB, 2*TAMANHO);
  return 0;
```





```
#include <stdio.h>
#define TAMANHO 10
void preencheVetor (int vet[], int tam) {
  int i:
  for (i = 0; i < tam; i++) {</pre>
    printf ("Informe o valor da posicao %d: ", i);
    scanf ("%d", &vet[i]);
void imprimeVetor (int vet[], int tam) {
  int i:
  for (i = 0; i < tam; i++)
    printf("Posicao %d => %d\n", i, vet[i]);
int main() {
  int vetorA[TAMANHO], vetorB[2*TAMANHO];
 preencheVetor(vetorA, TAMANHO);
 preencheVetor(vetorB, 2*TAMANHO);
  imprimeVetor(vetorA, TAMANHO);
  imprimeVetor (vetorB, 2*TAMANHO);
  return 0;
```



- A linguagem C não faz uma cópia dos elementos do vetor na chamada da função.
- Quando passamos um vetor como parâmetro, a função chamada recebe uma referência para o vetor.
- Isso significa que a função chamada acessa os elementos armazenados nas posições de memória reservadas pela função que declarou o vetor.



Variáveis



Passagem de parâmetro por valor



Modificações realizadas na variável recebida por parâmetro **não alteram** variável externa à função.

Vetores



Passagem de parâmetro por referência



Modificações realizadas no vetor recebido por parâmetro alteram o vetor externo passado na chamada da função.



```
#include <stdio.h>
#define TAMANHO 4
void zeraParametros(int varB, int vetorB[], int tam) {
  int i;
 varB = 0;
  for (i = 0; i < tam; i++)
    vetorB[i] = 0;
int main() {
  int varA = 12;
  int vetorA [TAMANHO] = \{ 5, 6, 7, 8 \};
  printf("Antes => varA=%d e vetorA[1]=%d\n",
         varA, vetorA[1]);
  zeraParametros(varA, vetorA, TAMANHO);
  printf("Depois => varA=%d e vetorA[1]=%d\n",
         varA, vetorA[1]);
  return 0;
                      Antes => varA=12 e vetorA[1]=6
                      Depois => varA=12 e vetorA[1]=0
```

Exercício resolvido 1



 Problema: Criar uma função que receba um vetor de números reais e seu tamanho e retorne o índice do maior valor contido no vetor. Se houver mais de uma ocorrência do maior valor, retornar o índice do primeiro. Faça um programa principal para testar a função.

 Vamos ver agora o resultado do teste de mesa para o problema.





```
int encontraMaior (float vetor[], int tam)
  int i, indice = 0;
  float maior = vetor[0];
  for (i = 1; i < tam; i++)</pre>
    if (vetor[i] > maior)
      maior = vetor[i];
      indice = i;
  return indice;
int main()
  float vet[6] = \{3.0, 4.3, 5.6, 2.8, 7.9, 3.4\};
  int posicao;
  posicao = encontraMaior(vet, 6);
  printf("Maior valor esta na posicao %d", posicao);
  return 0;
```



```
int encontraMaior (float vetor[], int tam)
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
         indice = i:
10
11
12
13
    return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                      2.8, 7.9, 3.4};
2.0
    int pos;
    pos = encontraMaior ( vet, 6);
2.1
     printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Dado que, com vetores, a quantidade de informação manipulada nos programas vai crescer, usaremos um novo formato de teste de mesa.

Ordem de execução:



```
int encontraMaior (float vetor[], int tam)
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
         indice = i:
11
12
13
     return indice;
14 }
15
16 int main()
17
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                      2.8, 7.9, 3.4};
2.0
    int pos;
2.1
    pos = encontraMaior ( vet, 6);
     printf ("Maior valor na posicao %d'
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

A partir de agora, os testes de mesa vão apresentar:

- a ordem de execução das linhas do programa;
- os valores armazenados em memória (variáveis, parâmetros e vetores);
- as impressões realizadas.

Ordem de execução:



```
int encontraMaior (float vetor[], int tam)
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
         indice = i;
11
12
13
     return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                      2.8, 7.9, 3.4};
20
     int pos;
2.1
     pos = encontraMaior ( vet, 6);
2.2
     printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
      indice = i:
1 1
12
13
   return indice;
14 }
15
16 int main()
17 {
18
    float vet[6] = \{3.0, 4.3, 5.6,
19
                     2.8, 7.9, 3.4};
20
     int pos;
2.1
    pos = encontraMaior ( vet, 6);
     printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam =
i =
indice =
maior =
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21



```
int encontraMajor (float vetor[], int tam)
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
        indice = i:
1 1
12
13
   return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                     2.8, 7.9, 3.4};
20
   int pos;
    pos = encontraMaior ( vet, 6);
2.1
     printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i =
indice =
maior =
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1



```
int encontraMaior (float vetor[], int tam)
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
      indice = i:
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                     2.8, 7.9, 3.4};
20
   int pos;
    pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = ?
indice = 0
maior = 3.0
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4



```
int encontraMaior (float vetor[], int tam)
2
3
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
      indice = i;
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
    float vet[6] = \{3.0, 4.3, 5.6,
19
                     2.8, 7.9, 3.4};
20
   int pos;
   pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 1
indice = 0
maior = 3.0
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
         indice = i:
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                     2.8, 7.9, 3.4};
20
   int pos;
    pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 1
indice = 0
maior = 3.0
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
         indice = i;
10
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                      2.8, 7.9, 3.4};
20
   int pos;
    pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 1
indice = 1
maior = 4.3
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
3
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
      indice = i:
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
    float vet[6] = \{3.0, 4.3, 5.6,
19
                     2.8, 7.9, 3.4};
20
   int pos;
   pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 2
indice = 1
maior = 4.3
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
         indice = i:
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
    float vet[6] = \{3.0, 4.3, 5.6,
19
                     2.8, 7.9, 3.4};
20
   int pos;
    pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 2
indice = 1
maior = 4.3
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
         indice = i;
10
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                      2.8, 7.9, 3.4};
20
   int pos;
    pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 2
indice = 2
maior = 5.6
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7 9 10



```
int encontraMaior (float vetor[], int tam)
2
3
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
      indice = i;
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
    float vet[6] = \{3.0, 4.3, 5.6,
19
                     2.8, 7.9, 3.4};
20
   int pos;
   pos = encontraMaior(vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 3
indice = 2
maior = 5.6
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7 9 10 5



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
         indice = i:
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                      2.8, 7.9, 3.4};
20
   int pos;
    pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 3
indice = 2
maior = 5.6
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7 9 10 5 7



```
int encontraMaior (float vetor[], int tam)
2
3
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
      indice = i;
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
    float vet[6] = \{3.0, 4.3, 5.6,
19
                     2.8, 7.9, 3.4};
20
   int pos;
   pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 4
indice = 2
maior = 5.6
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7 9 10 5 7 5



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
         indice = i:
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                      2.8, 7.9, 3.4};
20
   int pos;
    pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6

i = 4

indice = 2

maior = 5.6
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7 9 10 5 7 5 7



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
         indice = i;
10
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                      2.8, 7.9, 3.4};
20
   int pos;
    pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 4
indice = 4
maior = 7.9
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7 9 10 5 7 5 7 9 10



```
int encontraMaior (float vetor[], int tam)
2
3
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
      indice = i:
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                     2.8, 7.9, 3.4};
20
   int pos;
   pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 5
indice = 4
maior = 7.9
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7 9 10 5 7 5 7 9 10 5



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
         indice = i:
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                      2.8, 7.9, 3.4};
20
   int pos;
    pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 5
indice = 4
major = 7.9
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7 9 10 5 7 5 7 9 10 5 7



```
int encontraMaior (float vetor[], int tam)
2
3
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
      indice = i:
11
12
13
   return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                     2.8, 7.9, 3.4};
20
   int pos;
   pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 6
indice = 4
maior = 7.9
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7 9 10 5 7 5 7 9 10 5 7 5



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
         indice = i:
11
12
13
     return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                      2.8, 7.9, 3.4};
20
   int pos;
    pos = encontraMaior ( vet, 6);
2.1
    printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
24 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 6
indice = 4
maior = 7.9
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7 9 10 5 7 5 7 9 10 5 7 5 13



```
int encontraMaior (float vetor[], int tam)
2
     int i, indice = 0;
     float maior = vetor[0];
     for (i = 1; i < tam; i++)
       if (vetor[i] > maior)
         maior = vetor[i];
10
       indice = i:
11
12
13
    return indice;
14 }
15
16 int main()
17 {
18
     float vet[6] = \{3.0, 4.3, 5.6,
19
                      2.8, 7.9, 3.4};
20
     int pos;
2.1
     pos = encontraMaior ( vet, 6);
     printf ("Maior valor na posicao %d", pos);
2.3
     return 0;
2.4 }
```

Entrada:

vetor de tamanho = 6

Variáveis da Sub-Rotina:

```
tam = 6
i = 6
indice = 4
maior = 7.9
```

i	0	1	2	3	4	5
vet	3.0	4.3	5.6	2.8	7.9	3.4

Maior valor na posição 4

Ordem de execução:

16 18 21 1 3 4 5 7 9 10 5 7 9 10 5 7 5 7 9 10 5 7 5 13 21 22 23

Exercício resolvido 2



 Problema: Criar uma função em C que receba um vetor de números reais e um valor inteiro representando o seu tamanho. Essa função deverá ordenar o vetor em ordem crescente.





```
void ordena(float vet[], int tam)
  int i, j;
  float aux;
  for (i = 0; i \le (tam-2); i++)
    for(j = tam-1; j > i; j--)
      if ( vet[j] < vet[j-1] )
        aux=vet[j];
        vet[j] = vet[j-1];
        vet[j-1]=aux;
```



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam =
i =
j =
aux =
```

	0	1	2	3	4
vet	11.0	22.0	3.0	44.0	5.0

Vamos supor para esse exercício que o vetor de entrada tenha 5 posições e os seguintes valores:

Ordem de execução:



```
void ordena (float vet[], int tam)
3
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
9
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i =
j =
aux =
```

	0	1	2	3	4
vet	11.0	22.0	3.0	44.0	5.0

Ordem de execução:

1



```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for(i = 0; i <= tam-2; i++)
6
        for (j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = ?
j = ?
aux = ?
```

	0	1	2	3	4
vet	11.0	22.0	3.0	44.0	5.0

Ordem de execução:

1 3 4



```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for(i = 0; i <= tam-2; i++)</pre>
6
        for(j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 0
j = ?
aux = ?
```

	0	1	2	3	4
vet	11.0	22.0	3.0	44.0	5.0



Ordem de execução:

1 3 4 5



```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
6
        for(j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1]=aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 0
j = 4
aux = ?
```

	0	1	2	3	4
vet	11.0	22.0	3.0	44.0	5.0



↑ j

Ordem de execução:

1 3 4 5 7



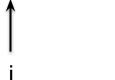
```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
8
9
             if (
                  vet[j] < vet[j-1]
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

	0	1	2	3	4
vet	11.0	22.0	3.0	44.0	5.0
					A



) j

Ordem de execução:

1 3 4 5 7 9



```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[i];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 0
j = 4
aux = 5.0
```

	0	1	2	3	4
vet	11.0	22.0	3.0	44.0	5.0





Ordem de execução:

1 3 4 5 7 9 11



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
                aux=vet[i];
12
                vet[i] = vet[i-1];
13
                vet[j-1]=aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

	0	1	2	3	4
vet	11.0	22.0	3.0	44	44



↑ j

Ordem de execução:

1 3 4 5 7 9 11 12



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
                vet[i] = vet[i-1];
13
                vet[i-1]=aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

	0	1	2	3	4
vet	11.0	22.0	3.0	5.0	44.0
·					<u> </u>





Ordem de execução:

1 3 4 5 7 9 11 12 13



```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for(j = tam-1; j > i; j--
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 0
j = 3
aux = 5.0
```

	0	1	2	3	4
vet	11.0	22.0	3.0	5.0	44.0
	1			↑	

Ordem de execução:

1 3 4 5 7 9 11 12 13 7



```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
8
9
             if (
                  vet[i] < vet[i-1]
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 0
j = 3
aux = 5.0
```

	0	1	2	3	4
vet	11.0	22.0	3.0	5.0	44.0
	1			\uparrow	
i			i		

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9



```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
6
        for (j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 0
j = 2
aux = 5.0
```

	0	1	2	3	4
vet	11.0	22.0	3.0	5.0	44.0
	1				

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7



```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
8
9
                  vet[i] < vet[i-1]
             if (
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 0
j = 2
aux = 5.0
```

	0	1	2	3	4
vet	11.0	22.0	3.0	5.0	44.0
	Î		1		

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9



3

44.0

```
Entrada:
1 void ordena (float vet[], int tam)
                                             vetor de tamanho = 5
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
                                           Variáveis:
                                             tam = 5
        for (j = tam-1; j > i; j--)
                                                i = 0
                                                 = 2
             if ( vet[j] < vet[j-1])
                                             aux = 3.0
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
                                               11.0
                                                    3.0
                                                        22.0
                                                             5.0
                                            vet
13
                vet[j-1]=aux;
14
15
16
17
```

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13



```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for(j = tam-1; j > i; j--
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 0
j = 1
aux = 3.0
```

	0	1	2	3	4
vet	11.0	3.0	22.0	5.0	44.0
		^			



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
9
                  vet[i] < vet[i-1]
             if
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 0
j = 1
aux = 3.0
```

	0	1	2	3	4
vet	11.0	3.0	22.0	5.0	44.0
	↑	↑			



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9



```
Entrada:
 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
                                           Variáveis:
                                             tam = 5
        for (j = tam-1; j > i; j--)
                                                = 0
             if ( vet[j] < vet[j-1])
                                             aux = 3.0
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1]=aux;
14
15
16
17
```

vetor de tamanho = 5

	0	1	2	3	4
vet	3.0	11.0	22.0	5.0	44.0

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for(j = tam-1; j > i; j--
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 0
j = 0
aux = 3.0
```

	0	1	2	3	4
vet	3.0	11.0	22.0	5.0	44.0



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for(i = 0; i <= tam-2; i++)</pre>
6
         for (j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 1
j = 0
aux = 3.0
```

	0	1	2	3	4
vet	3.0	11.0	22.0	5.0	44.0



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for(j = tam-1; j > i; j--
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 1
j = 4
aux = 3.0
```

	0	1	2	3	4
vet	3.0	11.0	22.0	5.0	44.0
					A



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
8
9
             if (
                  vet[j] < vet[j-1]
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 1
j = 4
aux = 3.0
```

	0	1	2	3	4
vet	3.0	11.0	22.0	5.0	44.0
		\uparrow			

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
6
        for(j = tam-1; j > i; j--1
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 1
j = 3
aux = 3.0
```

	0	1	2	3	4
vet	3.0	11.0	22.0	5.0	44.0
		1		1	

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
8
9
             if
                  vet[j] < vet[j-1]
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 1
j = 3
aux = 3.0
```

	0	1	2	3	4
vet	3.0	11.0	22.0	5.0	44.0
		1		1	

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 7



```
Entrada:
  void ordena (float vet[], int tam)
                                              vetor de tamanho = 5
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
                                            Variáveis:
                                              tam = 5
         for (j = tam-1; j > i; j--)
                                                  = 1
             if ( vet[j] < vet[j-1])
                                              aux = 5.0
11
                aux=vet[j];
                                                 0
                                                      1
                                                                    4
12
                vet[j] = vet[j-1];
                                                     11.0
                                                3.0
                                                          5.0
                                                              22.0
                                                                   44.0
                                            vet
13
                vet[j-1] = aux;
14
15
16
17
```

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 11 12 13



```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
6
        for (j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 1
j = 2
aux = 5.0
```

	0	1	2	3	4
vet	3.0	11.0	5.0	22.0	44.0
		↑	↑		



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 7 9 11 12 13 7 9 7 9 7 9 11 12 13 7



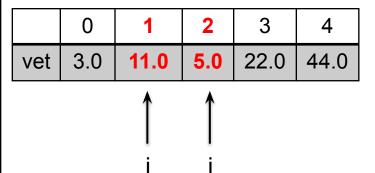
```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
8
9
                  vet[i] < vet[i-1]
             if
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 1
j = 2
aux = 5.0
```



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7 9 11 12 13 7 9



```
Entrada:
  void ordena (float vet[], int tam)
                                              vetor de tamanho = 5
     int i, j;
     float aux;
                                            Variáveis:
     for (i = 0; i \le tam-2; i++)
                                              tam = 5
         for (j = tam-1; j > i; j--)
                                                  = 1
             if ( vet[j] < vet[j-1])
                                              aux = 5.0
10
11
                 aux=vet[j];
                                                               3
                                                 0
                                                                    4
12
                vet[j] = vet[j-1];
                                                3.0
                                                     5.0
                                                         11.0
                                                              22.0
                                                                   44.0
                                             vet
13
                 vet[j-1] = aux;
14
15
16
17
```

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7 9 11 12 13



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
8
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5

i = 1

j = 1

aux = 5.0
```

	0	1	2	3	4
vet	3.0	5.0	11.0	22.0	44.0



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7 9 11 12 13 7



4

44.0

```
Entrada:
  void ordena (float vet[], int tam)
                                              vetor de tamanho = 5
     int i, j;
     float aux;
                                            Variáveis:
     for(i = 0; i <= tam-2; i++)
6
                                              tam = 5
         for (j = tam-1; j > i; j--)
                                                 i = 2
             if ( vet[j] < vet[j-1])
                                              aux = 5.0
10
11
                aux=vet[j];
                                                          2
                                                               3
                                                 0
12
                vet[i] = vet[i-1];
                                                3.0
                                                     5.0
                                                         11.0
                                                              22.0
                                            vet
13
                vet[j-1] = aux;
14
15
16
17
```

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7 9 11 12 13 7 9



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for(j = tam-1; j > i; j--
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 2
j = 4
aux = 5.0
```

	0	1	2	3	4
vet	3.0	5.0	11.0	22.0	44.0



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7 9 7 9 11 12 13 7 9 11 12 13 7 5 7



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
8
9
             if (
                  vet[j] < vet[j-1]
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 2
j = 4
aux = 5.0
```

	0	1	2	3	4
vet	3.0	5.0	11.0	22.0	44.0
		<u> </u>			\uparrow

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for(j = tam-1; j > i; j--
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 2
j = 3
aux = 5.0
```

	0	1	2	3	4
vet	3.0	5.0	11.0	22.0	44.0
			1	\uparrow	

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
8
9
                  vet[i] < vet[i-1]
             if
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 2
j = 3
aux = 5.0
```

	0	1	2	3	4
vet	3.0	5.0	11.0	22.0	44.0
			1	Î	

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7 9 11 12 13 7 5 7 9 7 9



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for(j = tam-1; j > i; j--
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 2
j = 2
aux = 5.0
```

	0	1	2	3	4
vet	3.0	5.0	11.0	22.0	44.0



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 11 12 13 7 9 7 9 7 9 7



```
Entrada:
  void ordena (float vet[], int tam)
                                              vetor de tamanho = 5
     int i, j;
     float aux;
                                            Variáveis:
     for(i = 0; i <= tam-2; i++)
6
                                              tam = 5
         for (j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
                                              aux = 5.0
10
11
                 aux=vet[j];
                                                           2
                                                                3
                                                  0
                                                                     4
12
                 vet[i] = vet[i-1];
                                                 3.0
                                                     5.0
                                                         11.0
                                                              22.0
                                                                   44.0
                                             vet
13
                 vet[j-1] = aux;
14
15
16
17
```

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 7 9 11 12 13 7 9 11 12 13 7 5 7 9 7 9 7 5



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
6
        for(j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 3
j = 4
aux = 5.0
```

	0	1	2	3	4
vet	3.0	5.0	11.0	22.0	44.0



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for (j = tam-1; j > i; j--)
8
9
                  vet[i] < vet[i-1]
             if
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5

i = 3

j = 4

aux = 5.0
```

	0	1	2	3	4
vet	3.0	5.0	11.0	22.0	44.0



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 7 9 11 12 13 7 9 7 9 7 9 7 9 7 9 7 9



```
void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for(j = tam-1; j > i; j--
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[i] = vet[i-1];
13
                vet[j-1] = aux;
14
15
16
17
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 3
j = 3
aux = 5.0
```

	0	1	2	3	4
vet	3.0	5.0	11.0	22.0	44.0



Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 7 9 11 12 13 7 9 11 12 13 7 5 7 9 7 9 7 5 7 9 7



```
Entrada:
  void ordena (float vet[], int tam)
                                              vetor de tamanho = 5
     int i, j;
     float aux;
                                            Variáveis:
     for(i = 0; i <= tam-2; i++)
6
                                              tam = 5
         for (j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
                                              aux = 5.0
10
11
                 aux=vet[j];
                                                          2
                                                                3
                                                  0
                                                                     4
12
                 vet[i] = vet[i-1];
                                                 3.0
                                                     5.0
                                                         11.0
                                                              22.0
                                                                   44.0
                                             vet
13
                 vet[j-1] = aux;
14
15
16
17
```

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 7 9 11 12 13 7 9 11 12 13 7 5 7 9 7 9 7 5 7 9 7 5



```
1 void ordena (float vet[], int tam)
     int i, j;
     float aux;
     for (i = 0; i \le tam-2; i++)
        for(j = tam-1; j > i; j--)
             if ( vet[j] < vet[j-1])
10
11
                aux=vet[j];
12
                vet[j] = vet[j-1];
13
                vet[j-1] = aux;
14
15
16
```

Entrada:

vetor de tamanho = 5

Variáveis:

```
tam = 5
i = 4
j = 3
aux = 5.0
```

Saída: vetor modificado (ordenado)

	0	1	2	3	4
vet	3.0	5.0	11.0	22.0	44.0

Ordem de execução:

1 3 4 5 7 9 11 12 13 7 9 7 9 11 12 13 7 9 11 12 13 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7 9 7



```
void ordena(float vet[], int tam)
  int i, j;
  float aux;
  for(i = 0; i <= (tam-2); i++)</pre>
    for(j = tam-1; j > i; j--)
      if ( vet[j] < vet[j-1] )
        aux=vet[j];
        vet[j] = vet[j-1];
        vet [j-1] = aux;
int main()
  int i;
  float
vet[5] = \{11.0, 22.0, 3.0, 44.0, 5.0\};
  ordena (vet, 5);
  for (i=0; i < 5; i++)
    printf("%.2f\n", vet[i]);
  return 0;
```

- Programa completo.
- Esse algoritmo de ordenação é conhecido como algoritmo da ordenação por bolha (ou bubble sort).

Faça o teste de mesa do programa ao lado.

```
#include <stdio.h>
   #define TAM VET A 2
   #define TAM VET B 5
   void espelhaVetor (int vet[], int tam) {
     int i;
     for (i = 0; i < (tam / 2); i++)
 6
        vet[(tam - 1) - i] = vet[i];
 8
   void imprimeVetor (int vet[], int tam) {
 9
     int i;
10
     for (i = 0; i < tam; i++)
11
       printf("v[%d] => %d\n", i, vet[i]);
12
     printf("----\n");
13
14
   int main() {
15
     int vetorA[TAM VET A] = { 1, 2 };
16
     int vetorB[TAM VET B] = \{ 3, 4, 5, 6, 7 \}
17
18
     imprimeVetor (vetorA, TAM VET A);
19
     imprimeVetor (vetorB, TAM VET B);
20
     espelhaVetor (vetorB, TAM VET B);
21
     imprimeVetor (vetorB, TAM VET B);
22
     return 0;
23
```



Faça o teste de mesa do programa ao lado. Apresente a ordem de execução das linhas do programa; os valores armazenados em memória (variáveis, parâmetros e vetores); as impressões realizadas.

```
1. #include <stdio.h>
2. #define TAM 3
    void funcao(int v[], int tam) {
   int i;
    for(i = 0; i < tam; i++){
    printf("b %d ", v[i]);
     if(v[i] > tam)
       return i+1;
     else
10.
     v[i] = v[i] * v[i];
11.
12.
     return tam;
13. }
14.
    int main(){
15.
     int r=TAM, vet [TAM] = \{2, 1, 3\};
16.
     while(r == TAM) {
17.
       r = funcao(vet, TAM);
18.
       printf("v %d ", r);
19.
20.
     return 0;
21. }
```



9) Elabore duas funções que façam respectivamente a união e a interseção de duas sequências de inteiros. Cada função deverá receber 4 parâmetros: três vetores de valores inteiros, (A, B e C) e um inteiro representando o tamanho dos dois primeiros vetores (tamAB). Assuma que o tamanho do vetor C é apropriado para cada operação.

Desenvolva um programa que crie quatro vetores, sendo v1, v2 e vInter, de tamanho 10, e vUniao, de tamanho 20. O programa deve preencher os vetores v1 e v2 com valores do teclado e chamar as funções acima. Após as chamadas de função, o programa deve imprimir vInter e vUniao.



10) Faça um programa para controlar as vendas de uma loja com no máximo 1000 produtos. O programa deve conter o valor das vendas dos produtos entre os meses de junho e agosto.

Faça funções para:

- a) preencher um vetor com dados do teclado;
- b) imprimir o total vendido de cada produto no período;
- c) retornar o total vendido na loja durante o trimestre;
- d) reajustar o preço de todos os produtos em 10%.

Organize a função principal de modo que os valores das vendas sejam lidos e as funções acima sejam chamadas.



11) Faça um programa que contenha: (1) um vetor com N inteiros, representando códigos de produtos; (2) um vetor com N inteiros, representando a quantidade de cada produto em estoque; (3) um vetor com N reais, representando o valor de cada produto; (4) um vetor com M inteiros, com os códigos dos produtos vendidos no dia (pode haver repetição).

Elabore funções para:

- a) calcular o valor total dos produtos em estoque;
- b) calcular o valor total vendido no dia;
- c) atualizar o estoque, descontando as unidades vendidas de cada produto.



DESAFIO:

- a) Crie uma função que receba um vetor e o seu tamanho e preencha-o com valores digitados no teclado. A função deve descartar valores repetidos durante a leitura, ou seja, sempre que o usuário informar um valor já existente no vetor, a função deve solicitar um novo valor para a mesma posição.
- b) Elabore uma função que receba um vetor e o seu tamanho e reorganize os elementos do vetor de modo que os elementos pares fiquem posicionados à esquerda e os números ímpares à direita.
- c) Faça um programa que crie um vetor, chame as funções acima e imprima o vetor após cada função.

125

Vetores Numéricos

DCC 120 – Lab. de Programação I



Vetores



O vetor é uma estrutura:

- Homogênea
- Estática

Todas as componentes são de um mesmo tipo e seu tamanho permanece o mesmo durante toda a execução do programa.

Vetores: declaração

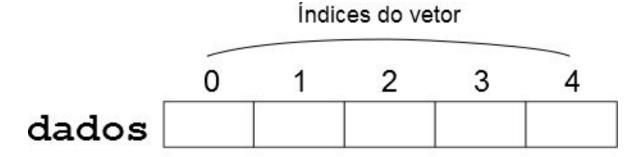


 A sintaxe em C para declaração de variável do tipo vetor é a seguinte:

```
tipoPrimitivo identificador[qtdeElementos];
```

Exemplo:

```
// vetor com 5 elementos do tipo int
int dados[5];
```



5 elementos do tipo int

Vetores: manipulação



 Cada um dos elementos de um vetor é referenciado individualmente por meio de um número inteiro entre colchetes após o nome do vetor.

	0	1	2	3	4
valores	3.6	2.7	4.2	7.9	1.2

Exemplos:

X = valores[1]; //atribui a x o valor da posição 1 do vetor valores Y = valores[4]; //atribui a x o valor da posição 4 do vetor valores valores[0] = 3.2; // a posição zero do vetor valores recebe o valor 3.2

Vetores: exemplo



 Leia um vetor de 10 posições (inteiros) e imprima-o na ordem invertida (da última para a primeira posição).

```
#include <stdio.h>
int main()
  int numeros[10],i;
  for (i=0; i<10;i++)
    printf("Digite valor %d: ",i);
    scanf("%d", &numeros[i]);
  printf("Vetor na ordem invertida:\n");
  for (i=9; i>=0; i--)
    printf("%d\n", numeros[i]);
  return 0;
```



Para cada exercício:

- Leia atentamente o enunciado até que o problema seja completamente entendido;
- Enumere os passos necessários para a solução do problema;
- "Traduza" os passos listados para a linguagem de programação C;
- Compile e corrija eventuais erros de sintaxe;
- Teste seu programa com diferentes entradas.



- 1) Desenvolva um programa que leia um vetor de números reais, um escalar e imprima o resultado da multiplicação do vetor pelo escalar.
- 2) Faça um programa que preencha um vetor de elementos inteiros com valores lidos do teclado e, ao final, imprima somente valores ímpares armazenados nos índices pares.
- 3) Faça um programa que preenche um vetor de inteiros com valores lidos do teclado. O programa deve verificar se há elementos repetidos no vetor e imprimir os índices de todos os pares de elementos repetidos.
- 4) Modifique o programa anterior para que este exiba apenas um par de índices de elementos repetidos, mesmo que haja mais repetições. Ao final, se nenhum par de índices repetidos foi encontrado, imprima a mensagem "não há elementos repetidos no vetor".



5) Faça um programa que crie um vetor com 100 posições e preencha as primeiras posições do vetor até que seja digitado um valor negativo. O programa deve inverter os valores armazenados nas 3 primeiras posições com os valores armazenados nas últimas 3 posições preenchidas e imprimir a sequência obtida ao final.

Por exemplo: Se a sequência digitada for

 $4 \ 6 \ 8 \ 3 \ 6 \ 9 \ 2 \ 0 \ 7 \ -1$

o programa deverá imprimir no final a sequência

7 0 2 3 6 9 8 6 4



- 6) Numa partida de basquete, a cada cesta, são registrados o time que fez a cesta (sendo 1 o time da casa e 2 o time visitante) e o número de pontos da cesta (1, 2 ou 3 pontos). Os valores são armazenados dois vetores de tamanho 200: T que armazena o time de basquete e P que armazena um valor positivo que indica a quantidade de pontos de cada cesta. Os dados de cada cesta são armazenados nos vetores T e P com mesmo índice, indicando qual time (1 ou 2) fez uma quantidade de pontos.
- a) Faça uma função que recebe quatro parâmetros: os vetores T e P, seu tamanho e o identificador de um dos times (1 ou 2). A função deverá retornar a quantidade de pontos que o time indicado fez. Por exemplo, para os vetores abaixo e o time 1, o valor retornado deve ser 8 (3 + 3 + 2).

b) Faça o programa principal que leia as informações necessárias para 20 cestas, utilize a função do item "a" para imprimir a quantidade de pontos do time da casa e do time visitante.

<u>DESAFIO:</u> Faça um programa que insira um elemento no meio de um vetor, de forma que os elementos a partir da posição de inserção sejam deslocados para a direita. Assim:

- (1) crie um vetor de 20 posições;
- (2) preencha cada posição do vetor com seu índice;
- (3) solicite que o usuário indique uma posição x entre 0 e 10;
- (4) desloque os elementos das posições de x a 18 para a direita (o último elemento da sequência será sobrescrito);
- (5) solicite que o usuário indique o valor que será inserido na posição x e armazene-o no vetor;
- (6) imprima a sequência.

DESAFIO: Faça um programa que apague um elemento no meio de um vetor, de forma que os elementos posteriores à posição deletada sejam deslocados para a esquerda. Seu programa deverá ser similar ao exercício anterior (repita os passos 1, 2, 3; no passo 4, desloque os elementos para a esquerda; e imprima o resultado).

Vetores e Funções



 Em C, vetores são passados sempre por referência.

 Ou seja, as modificações feitas no vetor que é um parâmetro no interior de uma função refletem nos dados do vetor passado como parâmetro pela função chamadora.

Vetores e Funções: Exemplo



```
#define TAMANHO 10
// definicao de outras subrotinas
// leVetor, imprimeVetor, maiorElemento
float mediaVetor(int vet[], int tam)
  int i, soma = 0;
  for(i = 0; i < tam; i++) {
    soma = soma + vet[i];
  return soma / (float) tam;
int main()
 int v[TAMANHO];
  leVetor(v, TAMANHO);
  imprimeVetor(v, TAMANHO);
 printf("\nMaior = %d.", maiorElemento(v, TAMANHO));
 printf("\nMédia = %.2f.", mediaVetor(v, TAMANHO));
  return 0;
```

Depuração via impressão



- Depurar (ou debugar) um código é o processo de acompanhar a execução de um programa ou trecho de programa, com o objetivo de verificar sua corretude ou de encontrar um erro.
- Para facilitar este processo, impressões podem ser inseridas no código para mostrar o valor de variáveis ao longo da execução e, assim, facilitar a localização de um eventual problema do código.

Depuração via impressão



- Executando os algoritmos a seguir, é
 possível verificar como as impressões podem
 auxiliar a identificação de erros.
- Observe que o programador é quem define que informações serão impressas. Tais informações devem ser selecionadas de forma a ajudá-lo a identificar o problema, dependendo do algoritmo.

Depuração via impressão: eleição



Problema:

Fazer uma função para apurar os votos de uma eleição com candidatos identificados pelos números 1, 2 e 3. Quaisquer valores diferentes de 1, 2 e 3 deverão ser considerados votos nulos.

Depuração via impressão: eleição



O programa abaixo tem um erro, mas apenas observando o que é impresso fica difícil identificar o problema...

```
#include <stdio.h>
                                               Candidato 1: 3 votos
#define PRIMEIRO CANDIDATO 1
                                                Candidato 2: 10 votos
#define ULTIMO CANDIDATO 3
                                                Candidato 3: 12 votos
#define MAX VOTOS 1000
void apuraVotos(int votos[], int tam)
  int i, cand, numVotos;
  numVotos = 0;
  for ( cand = PRIMEIRO CANDIDATO; cand <= ULTIMO CANDIDATO; cand++ )</pre>
     for( i = 0; i < tam; i++ ) {</pre>
        if( votos[i] == cand )
           numVotos++;
     printf("\nCandidato %d: %d votos", cand, numVotos);
int main()
  int votos[MAX VOTOS] =
               {1,2,3,4,5,6,7,8,3,5,2,5,2,5,2,4,4,6,1,1,2,2,5,4,2};
  apuraVotos(votos, 25);
  return 0;
```

Depuração via impressão: eleição

Com impressões adicionais, identificar o problema é mais fácil...

```
#include <stdio.h>
#define PRIMEIRO CANDIDATO 1
#define ULTIMO CANDIDATO 3
#define MAX VOTOS 1000
void apuraVotos(int votos[], int tam) {
  int i, cand, numVotos;
  numVotos = 0;
  for( cand = PRIMEIRO CANDIDATO; cand <= ULTIMO CANDIDATO; cand++ ) {</pre>
    printf("\n\nApuracao do candidato %d:
                                                             (DEBUG) ", cand);
    for( i = 0; i < tam; i++ ) {</pre>
       if( votos[i] == cand )
          numVotos++;
       printf("\n i=%2d v[i]=%d numVotos=%d
                                                                 (DEBUG) ",
                i, votos[i], numVotos);
    printf("\nCandidato %d: %d votos", cand, numVotos);
int main() {
  int votos[MAX VOTOS] =
               \{1, 2, 3, 4, 5, 6, 7, 8, 3, 5, 2, 5, 2, 5, 2, 4, 4, 6, 1, 1, 2, 2, 5, 4, 2\};
  apuraVotos(votos, 25);
  return 0;
```

142

Depuração via impr Apuração do candidato 1:

Com impressões adicionais, ide

#include <stdio.h>

#define PRIMEIRO CANDIDATO 1

#define ULTIMO CANDIDATO 3

#define MAX VOTOS 1000

void apuraVotos(int votos[], int tal int i, cand, numVotos;

numVotos = 0;

for (cand = PRIMEIRO CANDIDATO; ca printf("\n\nApuracao do cand

for(i = 0; i < tam; i++)</pre> if(votos[i] == cand) numVotos++;

printf("\n i=%2d v[i] i, votos[i], numV

printf("\nCandidato %d: %d voto

int votos[MAX VOTOS] =

apuraVotos(votos, 25);

int main()

return 0;

{1,2,3,4,5,6,7,8,3,5,1

i=22 i=23 i = 24Candidato 1: 3 votos

i=21 v[i]=5 v[i]=4 v[i]=2

Apuracao do candidato 2:

v[i]=1

v[i]=2

v[i]=3

v[i]=1

v[i]=2

v[i]=3

v[i]=4

v[i]=5

v[i]=6

v[i]=7

v[i]=8

v[i]=3

v[i]=5

v[i]=2

v[i]=5

v[i]=2

v[i]=5

v[i]=2

v[i]=4

v[i]=4

v[i]=6

v[i]=1

v[i]=1

i=1

i=2

i = 3

i = 4

i=5

i = 6

i = 7

i=8

i = 9

i = 10

i=11

i=12

i = 13

i = 14

i=15

i=16

i=17

i=18

i = 19

i=20

i = 0

i=1

i=2

v[i]=2v[i]=2 numVotos=3 numVotos=3 numVotos=3

numVotos=3

numVotos=4

numVotos=4

numVotos=1

numVotos=3

numVotos=3 numVotos=3

numVotos=1 numVotos=2 (DEBUG) (DEBUG) (DEBUG)

(DEBUG) (DEBUG)

(DEBUG)

(DEBUG) (DEBUG)

(DEBUG) (DEBUG) (DEBUG) (DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

(DEBUG)

Depuração via impressão: ordenação



Usando o exemplo de ordenação visto no teste de mesa da aula de Algoritmos, podemos ver como a impressão pode ser utilizada para ilustrar (e checar) o funcionamento do algoritmo.

Depuração via impressão: ordenação



```
#include <stdio.h>
void imprime (float vet[], int tam) {
  int i;
  for (i=0; i < tam; i++)</pre>
    printf("%7.1f", vet[i]);
void ordena(float vet[], int tam) {
  int i, j;
  float aux;
 printf("Inicio: vet=>");
  imprime(vet, tam);
  for(i = 0; i <= (tam-2); i++)
    for (j = tam-1; j > i; j--) {
     printf("\ni=%d j=%d vet=>",i,j);
      if ( vet[j] < vet[j-1] )
        aux=vet[j];
        vet[j] = vet[j-1];
        vet [j-1] = aux;
        imprime (vet, tam);
       printf(" (trocou %d e %d)",j-1,j);
      else
        imprime(vet, tam);
int main() {
  float vet[5]={11.0,22.0,3.0,44.0,5.0};
  ordena (vet, 5);
  return 0;
```

```
Inicio:
                          vet=>
                                   11.0
                                          22.0
                                                  3.0
                                                         44.0
                                                                 5.0
                     j=4
                          vet=>
                                   11.0
                                          22.0
                                                  3.0
                                                          5.0
                                                                44.0 (trocou 3 e 4)
                i=0
                     j=3
                         vet=>
                                   11.0
                                          22.0
                                                  3.0
                                                          5.0
                                                                44.0
#include <stdio
                i=0
                     j=2
                          vet=>
                                   11.0
                                           3.0
                                                 22.0
                                                          5.0
                                                                44.0 (trocou 1 e 2)
void imprime(fl
                i=0
                     j=1
                          vet=>
                                    3.0
                                          11.0
                                                 22.0
                                                          5.0
                                                                44.0 (trocou 0 e 1)
  int i;
                     j=4
                          vet=>
                                    3.0
                                          11.0
                                                 22.0
                                                                44.0
                i=1
                                                          5.0
  for (i=0; i
                i=1
                     j=3
                          vet=>
                                    3.0
                                          11.0
                                                 5.0
                                                         22.0
                                                                44.0 (trocou 2 e 3)
    printf("%7
                i=1
                     j=2
                          vet=>
                                    3.0
                                           5.0
                                                 11.0
                                                         22.0
                                                                44.0 (trocou 1 e 2)
                     j=4
                                    3.0
                                                 11.0
                                                         22.0
                                                                44.0
                i=2
                          vet=>
                                           5.0
void ordena(fld
                i=2
                     j=3
                         vet=>
                                    3.0
                                           5.0
                                                 11.0
                                                         22.0
                                                                44.0
  int i, j;
                i=3
                     j=4 vet=>
                                           5.0
                                                 11.0
                                                         22.0
                                                                44.0
                                    3.0
  float aux;
  printf("Inicio:
                   vet=>");
  imprime(vet, tam);
  for(i = 0; i <= (tam-2); i++)
    for(j = tam-1; j > i; j--)
      printf("\ni=%d j=%d vet=>",i,j);
      if ( vet[j] < vet[j-1] )
        aux=vet[i];
        vet[j] = vet[j-1];
        vet [j-1] = aux;
        imprime(vet, tam);
        printf(" (trocou %d e %d)",j-1,j);
      else
        imprime (vet, tam);
int main() {
  float vet[5]={11.0,22.0,3.0,44.0,5.0};
```

ordena (vet, 5);

return 0;

146

Depuração via impressão: ordenação



```
#include <stdio.h>
void imprime (float vet[], int tam) {
  int i;
  for (i=0; i < tam; i++)
    printf("%7.1f", vet[i]);
void ordena(float vet[], int tam) {
  int i, j;
  float aux;
 printf("\nInicio: vet=>",i,j);
  imprime(vet, tam);
  for(i = 0; i <= (tam-2); i++)
    for (j = tam-1; j > i; j--) {
      if ( vet[j] < vet[j-1] ) {
        aux=vet[j];
        vet[j] = vet[j-1];
        vet [j-1] = aux;
        printf("\ni=%d j=%d vet=>",i,j);
        imprime (vet, tam);
        printf(" (trocou %d e %d)",j-1,j);
int main() {
  float vet[5]={11.0,22.0,3.0,44.0,5.0};
 ordena (vet, 5);
 return 0;
```

Depuração via impressão: ordenação

3.0

3.0

22.0

5.0

11.0

44.0

22.0 5.0

5.0

5.0

22.0

22.0

5.0

44.0 (trocou 3 e 4)

44.0 (trocou 1 e 2)

44.0 (trocou 0 e 1)

44.0 (trocou 2 e 3)

44.0 (trocou 1 e 2)



```
Inicio:
                         vet=>
                                  11.0
                                         22.0
#include <stdic
               i=0
                         vet=>
                                  11.0
                                         22.0
                    j=4
void imprime(fl
               i=0
                    i=2 vet=>
                                  11.0
                                        3.0
  int i;
               i=0
                    j=1
                        vet=>
                                3.0
                                         11.0
  for (i=0; i
               i=1
                     i=3 vet=> 3.0
                                         11.0
    printf("%7
                    i=2 vet=>
                                  3.0
                                          5.0
void ordena(float vet[], int tam) {
  int i, j;
  float aux;
  printf("\nInicio: vet=>",i,j);
  imprime(vet, tam);
  for(i = 0; i <= (tam-2); i++)</pre>
    for (j = tam-1; j > i; j--)
      if ( vet[j] < vet[j-1] ) {
        aux=vet[j];
        vet[j] = vet[j-1];
        vet [j-1] = aux;
        printf("\ni=%d j=%d vet=>",i,j);
        imprime (vet, tam);
       printf(" (trocou %d e %d)",j-1,j);
int main() {
  float vet[5]={11.0,22.0,3.0,44.0,5.0};
 ordena (vet, 5);
  return 0;
```



- 6) Faça uma função que inverta a ordem dos elementos de um vetor (exemplo: o vetor {9,2,5,6} ao final da função deve ser alterado para {6,5,2,9}). Faça um programa para testar a função com 3 vetores de tamanhos 5, 9 e 14. Use impressões para mostrar a configuração de um vetor a
- cada passo do algoritmo.
- 7) Faça uma função que receba como parâmetros um vetor de tamanho 5 e um número inteiro entre 0 e 99999, faça uma função que inicialize o vetor com cada dígito do número. Por exemplo, se o número for 4723, o vetor deverá, ao final, conter os valores {0,4,7,2,3}.



- 8) Faça um programa com 3 vetores, P1, P2 e P3, representando notas de alunos em 3 provas. Assuma que a turma tem até 100 alunos. Seu programa irá utilizar as seguintes funções:
- a) preencher um vetor com valores lidos do teclado
- b) imprimir um vetor
- c) imprimir a nota final de cada aluno (=(P1+P2+(2*P3))/4)
- d) imprimir os índices cuja nota < 60 em um vetor
- e) retornar número de notas >= 60 em um vetor

O programa principal deverá chamar a função para preencher os vetores e, em seguida, exibirá um menu, que permita que o usuário selecione o que será realizado.



DESAFIO) O desafio desta semana é implementar várias funções que possibilitarão a ordenação de uma sequência de números inteiros. O algoritmo da bolha já foi ilustrado no teste de mesa da aula de Algoritmos. Neste exercício, você vai implementar outros algoritmos de ordenação.

O objetivo deste desafio não é aprender a ordenar. A ordenação é usada simplesmente por ser um processo em que é fácil identificar se o resultado esperado foi alcançado. O grande desafio aqui é aprender a manipular índices para fazer o que o exercício pede.

Assim, observe a descrição e os exemplos que mostram o funcionamento de cada algoritmo. Depois de entender bem como o algoritmo funciona, tente implementá-lo.



DESAFIO)

Para começar, copie o código ao lado:

Dica: chame a função imprimeVetor em suas repetições para acompanhar o passo a passo de cada algoritmo.

```
#include <stdio.h>
#include <stdlib.h>
#define TAMANHO 10
void preencheVetorAleatoriamente (int vet[], int tam) {
  int i; //o vetor sera preenchido com valores de 0 a
999
  for (i = 0; i < tam; i++)
     vet[i] = rand() % 1000;
void imprimeVetor (int vet[], int tam) {
  int i:
  for (i = 0; i < tam; i++)
   printf(" %4d", vet[i]);
int main() {
  int vetor[TAMANHO];
  printf("\n\n Vetor aleatorio: ");
  preencheVetorAleatoriamente (vetor, TAMANHO);
  imprimeVetor (vetor, TAMANHO);
  printf("\n Vetor ordenado: ");
  //aqui deverá ser chamado o algoritmo de ordenação
  imprimeVetor (vetor, TAMANHO);
  return 0;
```



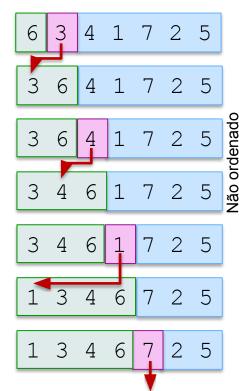
DESAFIO) (continuação)

ORDENAÇÃO POR INSERÇÃO

A ideia do algoritmo é manter e aumentar passo a passo o tamanho da sequência já ordenada (em verde). Para isso, a cada passo, um novo elemento (rosa) é reposicionado na sequência, isto é, o novo elemento é inserido na posição correta da sequência verde de modo que esta permaneça ordenada, como no exemplo:

A sequência ordenada inicialmente contém apenas o primeiro elemento. O próximo elemento a ser inserido é o primeiro elemento da parte não ordenada. Observe que, a cada iteração, os elementos maiores que o valor a ser sinserido na sequência ordenada precisam ser deslocados para a direita.

Faça uma função que receba como parâmetros um vetor e seu tamanho e implemente a ordenação por inserção.





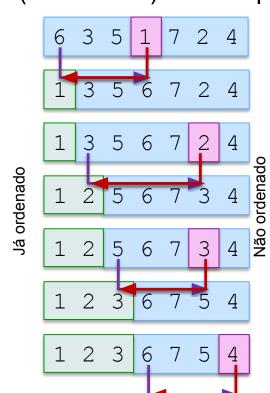
DESAFIO) (continuação)

ORDENAÇÃO POR SELEÇÃO

Este algoritmo também faz com que a sequência ordenada cresça a cada iteração, mas de forma distinta. A cada passo, é procurado o menor elemento na parte não ordenada (em azul) e este elemento é reposicionado na posição seguinte à sequência ordenada (em verde). A sequência ordenada inicialmente está vazia.

Observe que, a cada passo, o elemento da primeira posição da sequência não ordenada (em azul) troca de lugar com o menor elemento encontrado (em rosa).

Faça uma função que receba como parâmetros um vetor e seu tamanho e implemente a ordenação por seleção.



of jf

DESAFIO) (continuação)

ORDENAÇÃO POR CONTAGEM

Este algoritmo é bem diferente dos algoritmos anteriores e só pode ser usado quando os números a serem ordenados são inteiros e estão garantidamente dentro de um intervalo conhecido (ex: notas entre 0 e 100, idades entre 0 e 130, candidatos de uma eleição entre 1 e 999, etc). No exemplo abaixo, todos os valores estão no intervalo entre 0 e 7.

Neste algoritmo, primeiramente, um vetor de contadores é usado para contabilizar cada vez que um número aparece. Depois, cada índice do vetor de contadores é acessado em ordem crescente e o número de ocorrências do índice indica quantas vezes o número deverá aparecer na sequência.

Primeira part	<u>:e: contabilizar</u>	Segunda parte: montar a sequência
	0 1 2 3 4 5 6 7 8 _	0 1 2 3 4 5 6 7 8
5 1 4 5 5 1 7	0 0 0 0 0 0 0 0	0 2 0 0 1 3 0 1 0 ? ? ? ? ? ? ?
5 1 4 5 5 1 7	000001000	0 2 0 0 1 3 0 1 0 1 1 ? ? ? ? ?
5 1 4 5 5 1 7	0 1 0 0 0 1 0 0 0	0 2 0 0 1 3 0 1 0 1 1 ? ? ? ? ?
5 1 4 5 5 1 7	0 1 0 0 1 1 0 0 0	0 2 0 0 1 3 0 1 0 1 1 ? ? ? ? ?
5 1 4 5 5 1 7	0 1 0 0 1 2 0 0 0	0 2 0 0 1 3 0 1 0 1 1 4 ? ? ? ?

155