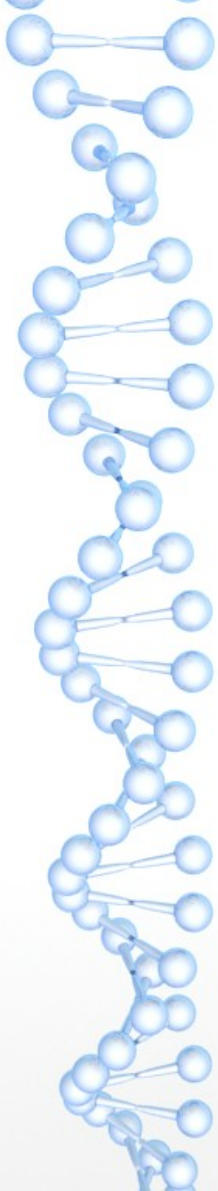


Algoritmos e Lógica de Programação I

Estrutura de Dados Homogêneas e Heterogêneas

Prof. MSc. Rafael Staiger Bressan
rafael.bressan@unicesumar.edu.br

Cronograma

- Introdução
- Vetores
- Ordenação em Vetores
- Pesquisa em Vetor
- Strings
- Matrizes
- Estruturas

Introdução

- Estruturas Homogêneas
  - Vetores e Matrizes
- Estruturas Heterogêneas
  - Structs

Vetores

- Imagine que você tem que receber o nome e a nota de 50 alunos de uma escola, e depois listar o nome de cada um e a média final de cada aluno e a média da turma.
- Agora imagine você na declaração de variáveis, declarando uma a uma, as 50 variáveis para o nome, depois as 50 variáveis para as notas

Vetores

- Vamos imaginar um vetor como sendo uma “caixa” com vários lugares separados para guardar seus documentos, ou seja, uma variável onde se pode armazenar mais de um dado.

Vetores

- Particularidades
  - Índice : Todo vetor contem um índice (número inteiro positivo) para identificar as posições de memória.
    - Em C, a primeira posição sempre será zero, ou seja, índice zero.
  - Declaração :
    - Para declarar um vetor de uma dimensão, adiciona-se colchetes logo após o nome da variável.
    - Exemplo: int vetor[5];
      - Declaração de um vetor de 5 posições, com índices começando em 0 e terminando em 4. (5 posições)

Vetores

int vetor[5];

0	1	2	3	4

Vetores

```
 4   #include <stdio.h>
 5   int main() {
 6       int vetor[5];
 7       vetor[0] = 10;
 8       vetor[1] = 20;
 9       vetor[2] = 30;
10       vetor[3] = 40;
11       vetor[4] = 50;
12       return 0;
13   }
```

0	1	2	3	4
10	20	30	40	50

Vetores

```
4  #include <stdio.h>
5  int main() {
6      int vetor[5], i;
7      for(i=0; i<5; i++){
8          vetor[i] = (i+1) * 10;
9      }
10     for(i=0; i<5; i++){
11         printf(" %d ", vetor[i]);
12     }
13     return 0;
14 }
```

0	1	2	3	4
10	20	30	40	50

Exemplos

- Crie um programa que leia 10 números inteiros, armazene-os em um vetor (A) e apresente os números do vetor e a sua posição.

```c
#include <stdio.h>
int main() {
    int vetor[10], i;
    for(i=0; i<10; i++){
        printf("Digite um numero : ");
        scanf("%d", &vetor[i]);
    }
    for(i=0; i<10; i++){
        printf("Valor: %d  Indice: %d \n", vetor[i], i);
    }
    return 0;
}
```

Exemplos

- Crie um programa que leia 5 números inteiros, armazene-os em um vetor (A) e apresente os números do vetor de maneira inversa a sua inserção.
  - Exemplo:
    - Entrada: 1 2 3 4 5
    - Saída: 5 4 3 2 1

```c
#include <stdio.h>
int main() {
    int vetor[5], i;
    for(i=0; i<5; i++){
        printf("Digite um numero : ");
        scanf("%d", &vetor[i]);
    }
    for(i=4; i>=0; i--){
        printf(" %d ", vetor[i]);
    }
    return 0;
}
```

Exemplos

- Ler dois vetores A e B de 5 elementos, calcule e armazene em um vetor C a soma de A com B.

```c
int main() {
    int A[5],B[5],C[5], i;
    for(i=0; i<5; i++){
        printf("Vetor A[%d] = ", i);
        scanf("%d", &A[i]);
        printf("Vetor B[%d] = ", i);
        scanf("%d", &B[i]);
        C[i] = A[i] + B[i];
    }
    for(i=0; i<5; i++){
        printf(" A[%d] + B[%d] = C[%d] \n", A[i], B[i], C[i]);
    }
    return 0;
}
```

Exemplos

- Desenvolva um programa que armazene 5 números em um vetor, calcule e apresente o menor elemento e sua posição.

```c
#include <stdio.h>
int main() {
    int vetor[5], menorElemento, posicao = 0, i;
    for(i=0; i<5; i++){
        printf("Vetor[%d] = ", i);
        scanf("%d", &vetor[i]);
    }
    menorElemento = vetor[0];
    for(i=0; i<5; i++){
        if(vetor[i] < menorElemento){
           menorElemento = vetor[i];
           posicao = i;
        }
    }
    printf("Menor elemento é %d na posição %d do vetor",menorElemento, posicao);
    return 0;
}
```

Exemplos

- Desenvolva um programa que armazene 5 números em um vetor, calcule e apresente a média dos valores contidos no vetor.

```c
#include <stdio.h>
int main() {
    int vetor[5], media = 0, i;
    for(i=0; i<5; i++){
        printf("Vetor[%d] = ", i);
        scanf("%d", &vetor[i]);
        media += vetor[i];
    }
    printf("Média : %.2f", (float) media / 5);
    return 0;
}
```

Exercícios

- Resolva 10 exercícios “Propostos” do livro “Fundamentos da programação de computadores: algoritmos, Pascal, C/C++ e Java” do capítulo (VETOR)

Ordenação em Vetores

- Ordenar
  - Processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado.
  - Imagine um lista de telefone desordenada...

Ordenação em Vetores
Métodos Simples

- Classificação dos métodos de ordenação:
  - Ordenação interna: arquivo a ser ordenado cabe todo na memória principal.
  - Ordenação externa: arquivo a ser ordenado não cabe na memória principal.
- Diferenças entre os métodos:
  - Em um método de ordenação interna, qualquer registro pode ser imediatamente acessado.
  - Em um método de ordenação externa, os registros são acessados sequencialmente ou em grandes blocos.

Ordenação em Vetores
Métodos Simples

- Método de Seleção (Selection Sort)
- Método de Inserção (Inserction Sort)
- **Método de Troca (Bubble Sort, ou Método de Flutuação / Bolha)**
- ...

Bubble Sort

- Algoritmo de ordenação simples.
  - a ideia é percorrer o vetor várias vezes (geralmente com o número de elementos), e a cada vez, ‘flutuar’ o maior elemento da sequência, ou seja, essa movimentação lembra a forma de como as bolhas em um reservatório de água, procuram seu próprio nível.

Bubble Sort

5	4	8	2	1	7

Bubble Sort

5 > 4 ?

Bubble Sort

5 > 4 ?

SIM

Bubble Sort

Troca

Bubble Sort

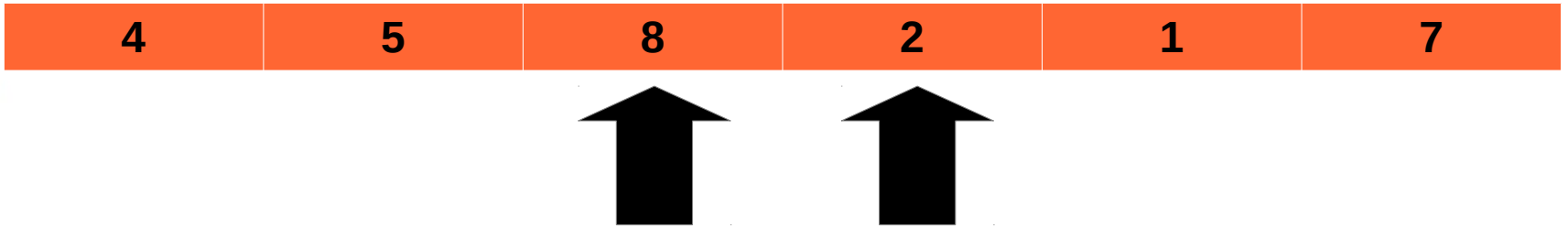
5 > 8 ?

Bubble Sort

5 > 8 ?

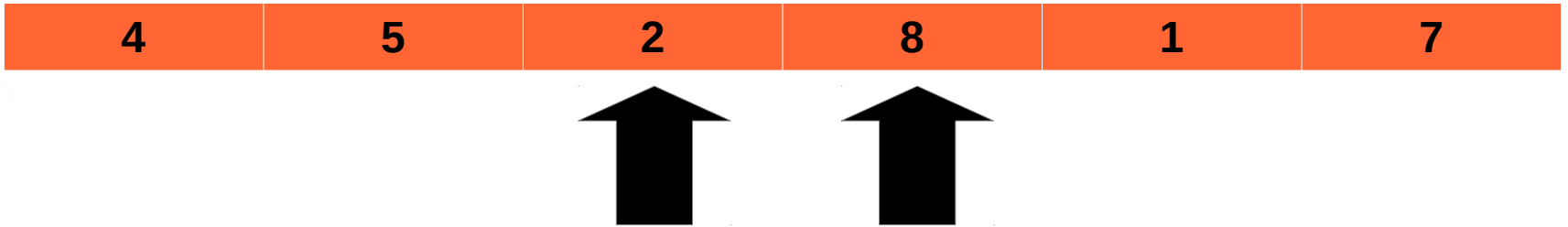
Não

Bubble Sort



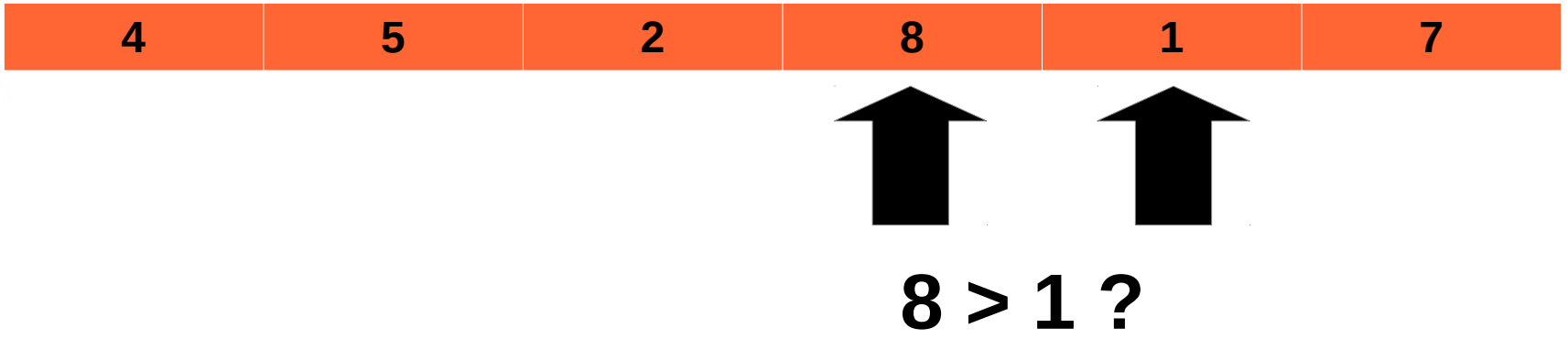
8 > 2 ?

Bubble Sort

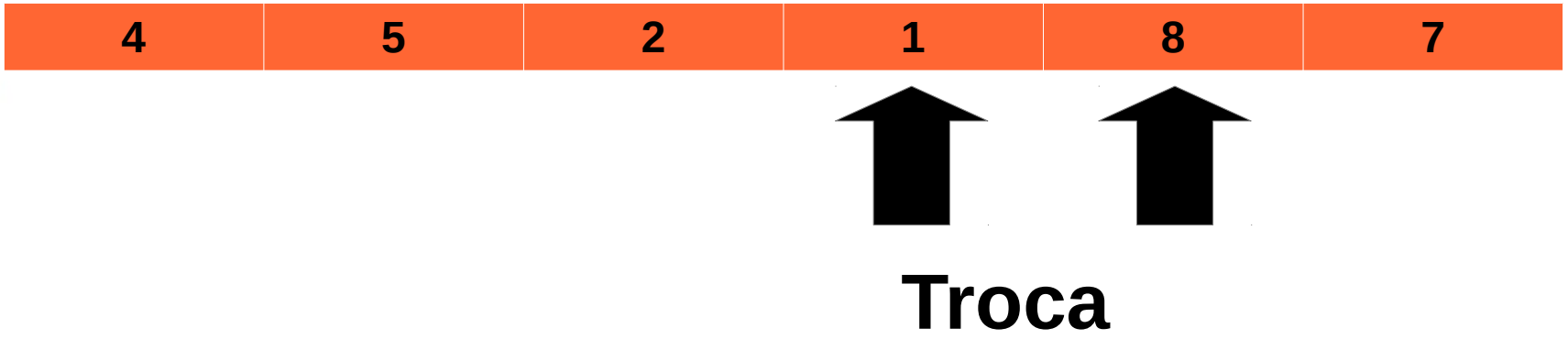


Troca

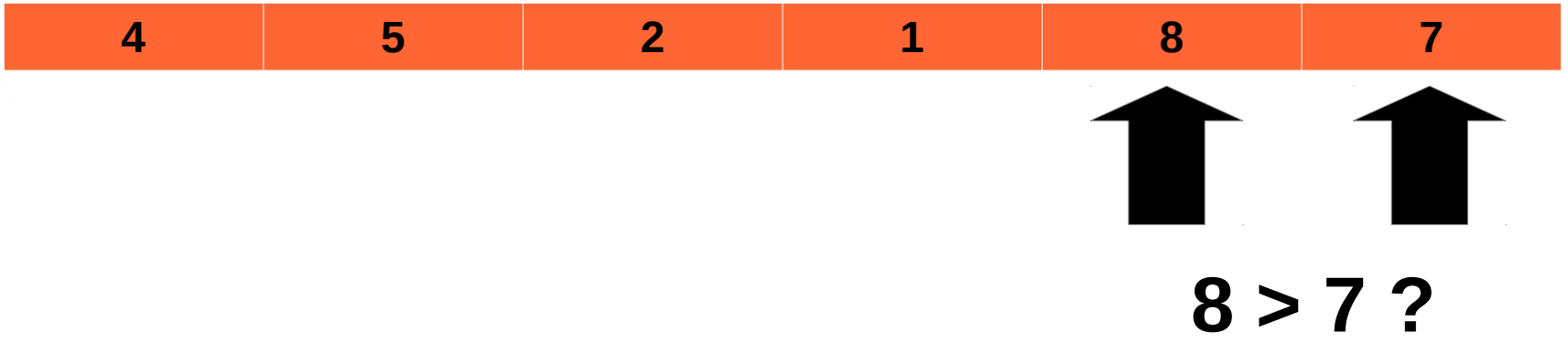
Bubble Sort



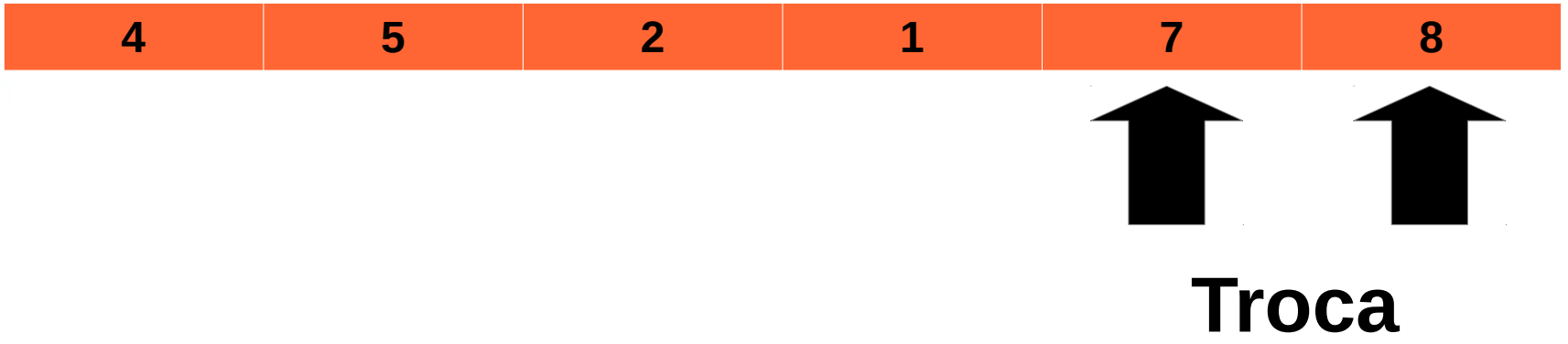
Bubble Sort



Bubble Sort



Bubble Sort



Bubble Sort

4 > 5 ?

Bubble Sort

4 > 5 ?

Não

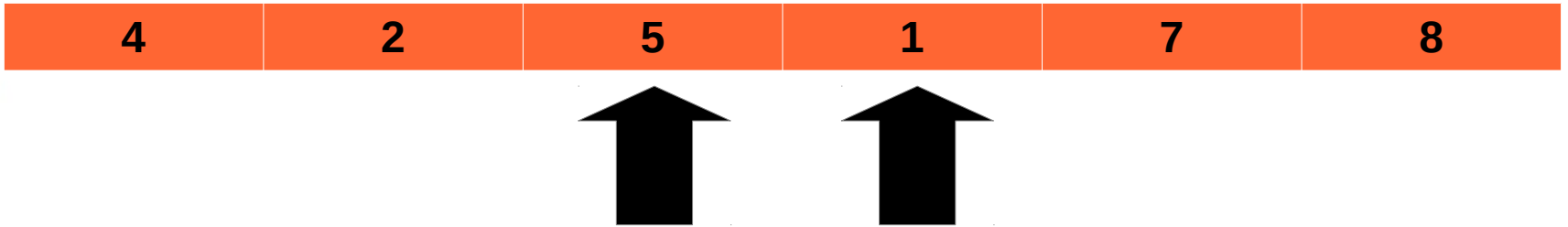
Bubble Sort

5 > 2 ?

Bubble Sort

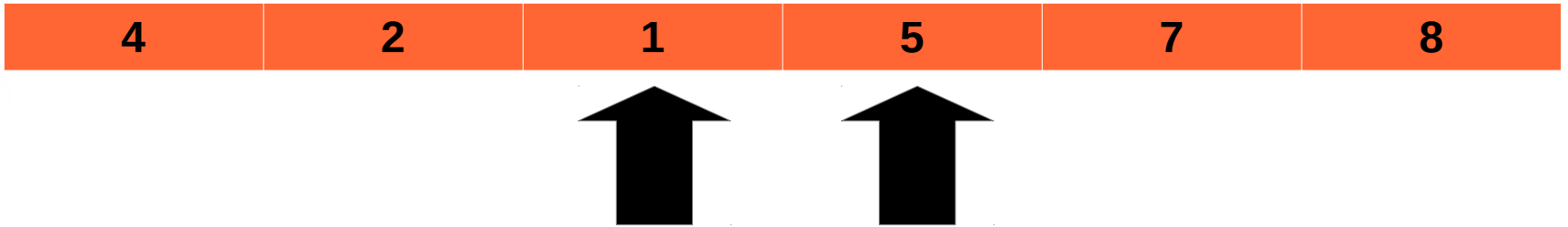
Troca

Bubble Sort



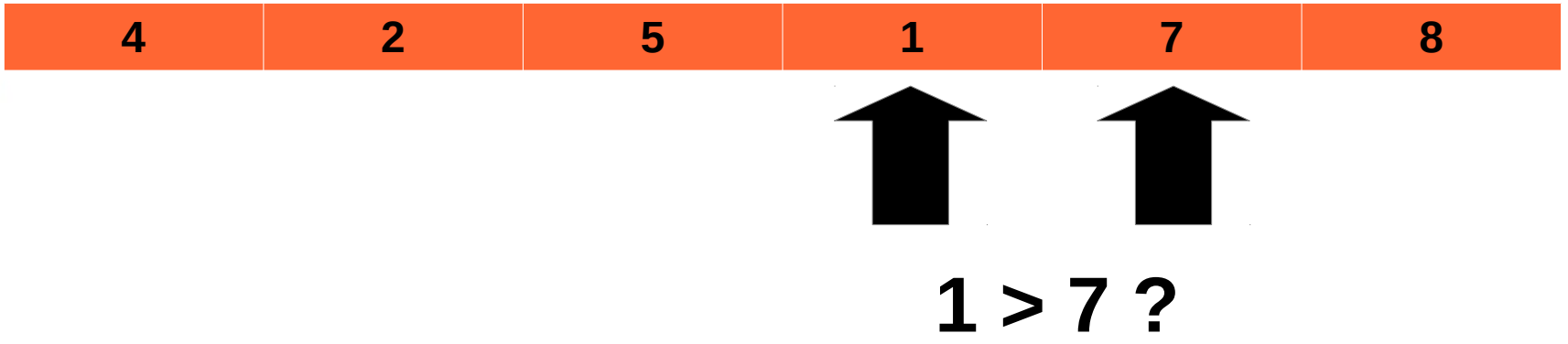
5 > 1 ?

Bubble Sort



Troca

Bubble Sort



Bubble Sort

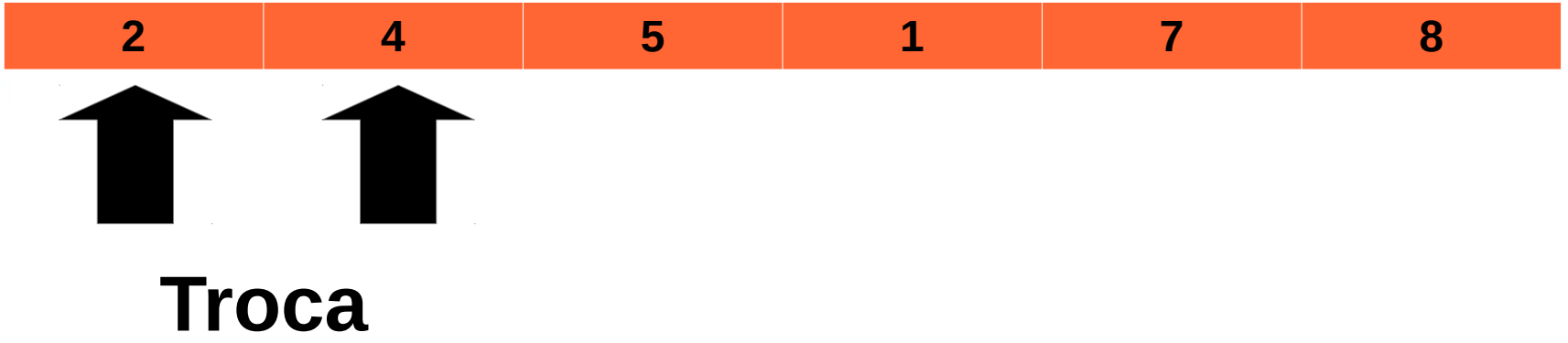
1 > 7 ?

Não

Bubble Sort

4 > 2 ?

Bubble Sort



Bubble Sort

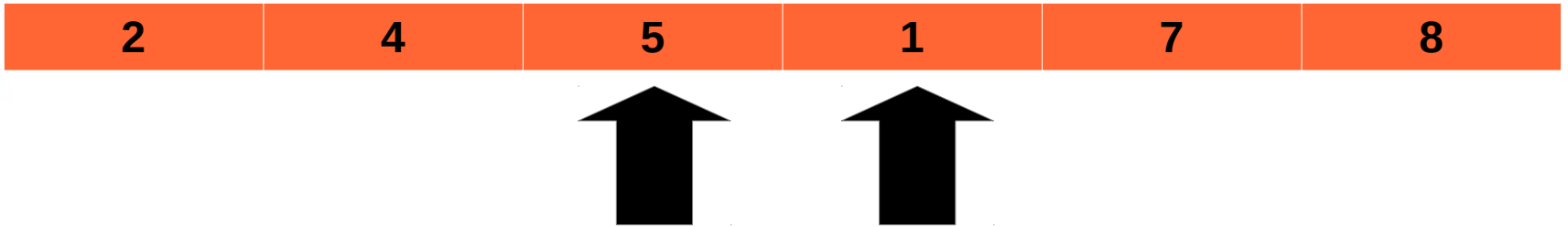
4 > 5 ?

Bubble Sort

4 > 5 ?

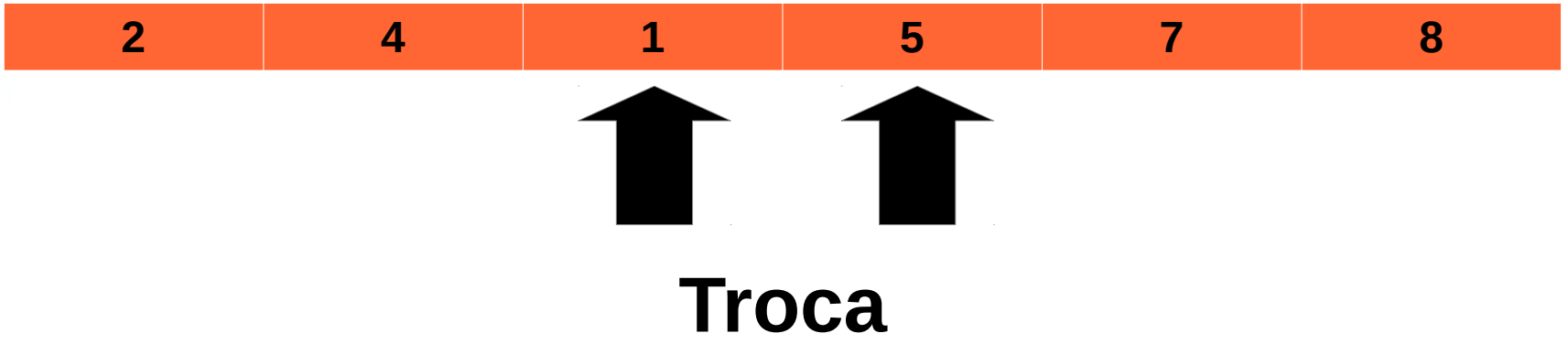
Não

Bubble Sort



5 > 1 ?

Bubble Sort



Bubble Sort

2 > 4 ?

Bubble Sort

2 > 4 ?

Não

Bubble Sort

4 > 1 ?

Bubble Sort

Troca

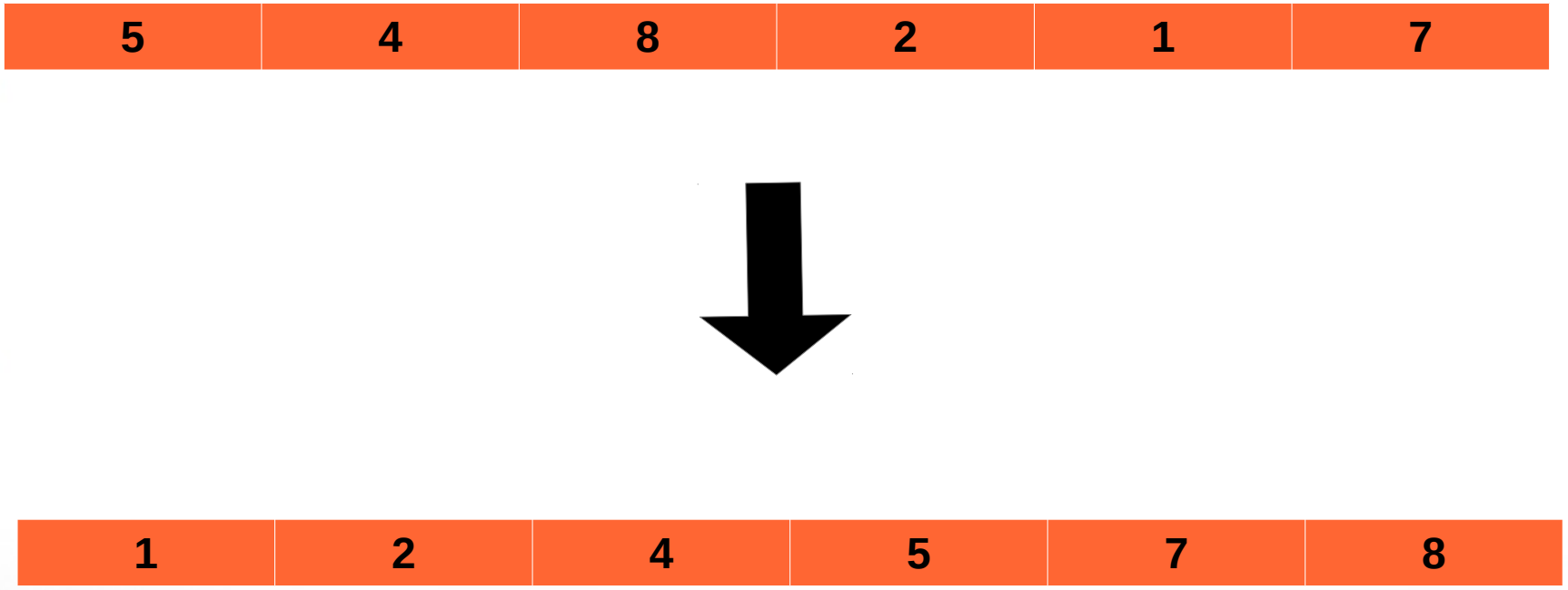
Bubble Sort

2 > 1 ?

Bubble Sort

Troca

Bubble Sort

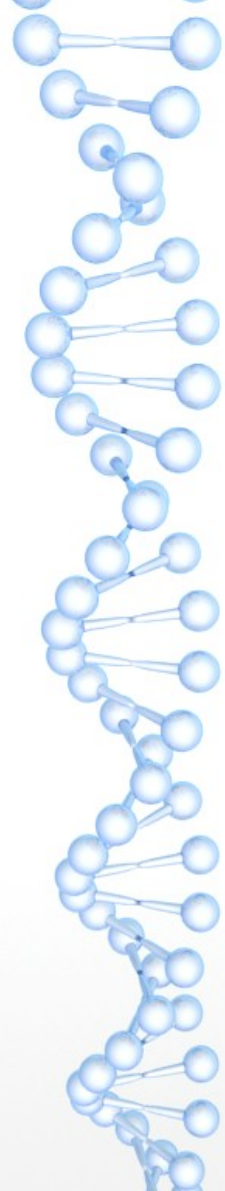


Atividade

- Considere o seguinte vetor

20	8	1	30	10	6

- Aplique o algoritmos *Bubble Sort* e apresente quantas trocas foram necessárias para ordenar o vetor.



```c
#include <stdio.h>
main(){
      int tamanhoVetor, i, vetor[10], controle, aux;
      for (i=0; i<10;i++){
        printf("Vetor[%d] : ", i+1);
        scanf("%d", &vetor[i]);
      }
      tamanhoVetor = 10;
      do{
           controle = 0;
           for (i=0; i<tamanhoVetor-1; i++){
              if (vetor[i] > vetor[i+1]){
                 aux = vetor[i+1];
                 vetor[i+1] = vetor[i];
                 vetor[i] = aux;
                 controle = 1;
              }
           }
           tamanhoVetor--;
      }while(controle != 0);
      for (i=0; i<10;i++){
        printf(" %d ", vetor[i]);
      }
}
```

Pesquisa em Vetor

- Considere o seguinte vetor

30	2	1	10	8	6

- Como verificar se existe o número (1) no vetor?

Pesquisa em Vetor

- Considere o seguinte vetor

30	2	1	10	8	6

- Como verificar se existe o número (1) no vetor?
  - Como o vetor está desordenado, a única saída é percorrer todo o vetor para verificação.
  - Veja o código a seguir

```c
#include <stdio.h>
main(){
      int i, vetor[10], valor;
      for (i=0; i<10;i++){
        printf("Vetor[%d] : ", i);
        scanf("%d", &vetor[i]);
      }
      printf("Digite o valor para procurar : ");
      scanf("%d", &valor);
      for (i=0; i<10;i++){
        if(vetor[i] == valor){
            printf("Vetor[%d] = %d \n", i, vetor[i]);
        }
      }
}
```

Pesquisa em Vetor

- Considere o seguinte vetor

30	2	1	10	8	6

- Como podemos melhorar essa busca?

Pesquisa em Vetor

- Considere o seguinte vetor

30	2	1	10	8	6

- Primeiro passo (Coloco o vetor em ordem do menor para o maior).

1	2	6	8	10	30

Pesquisa em Vetor

- Primeiro passo (Coloco o vetor em ordem do menor para o maior).

1	2	6	8	10	30

- Verifique se a posição aproximada do meio do vetor contem o número procurado, como exemplo, vamos procurar o número (10)

Pesquisa em Vetor

- Verifique se a posição aproximada do meio do vetor contem o número procurado, como exemplo, vamos procurar o número (10)

1	2	6	8	10	30

6 é valor procurado?

Pesquisa em Vetor

- Verifique se a posição aproximada do meio do vetor contem o número procurado, como exemplo, vamos procurar o número (10)

1	2	6	8	10	30

6 é valor procurado?
Não, então verifiquei se o valor procurado é maior ou menor que 6.

Pesquisa em Vetor

- Verifique se a posição aproximada do meio do vetor contem o número procurado, como exemplo, vamos procurar o número (10)

1	2	6	8	10	30

10 > 6 ?
Sim, localize o meio do vetor considerando as posições da direita.
Ou seja, estamos na posição 2 e o vetor tem 6 posições (0 a 5) o meio entre 3 e 5 é 4.

Caso fosse menor, consideraríamos o intervalo a esquerda.

Pesquisa em Vetor

- Verifique se a posição aproximada do meio do vetor contem o número procurado, como exemplo, vamos procurar o número (10)

Atividade

- Considere o seguinte vetor

8	17	58	30	2	1	6

- Aplique o algoritmos *Bubble Sort* e realize a busca do número 18.

Strings

- A biblioteca string.h da linguagem C, contém uma série de funções para manipular strings.
  - strcpy(string_destino, string_origem);
    - Realiza a cópia do conteúdo de uma variável a outra.
  - strncpy(string_destino, string_origem, tamanho);
    - Realiza a cópia do conteúdo de uma variável a outra, porém,
    - deve ser especificado o tamanho a ser copiado.

Strings

- strcat(string_destino, string_origem);
  - Realiza a concatenação do conteúdo de uma variável a outra.
- strncat(string_destino, string_origem, tamanho);
  - Realiza a concatenação do conteúdo de uma variável a outra,
  - porém, deve ser especificado o tamanho a ser concatenado.

Strings

- strlen(string);
  - Determina o tamanho de uma string.
- strcmp(string1, string2);
  - Compara o conteúdo de duas strings;
- strncmp(string1, string2, tamanho);
  - Também faz a comparação do conteúdo de duas strings, porém, deve ser especificado o tamanho a ser comparado;

Matrizes

- Matriz é a uma estrutura de dados do tipo vetor com duas ou mais dimensões.
- Os itens de uma matriz tem que ser todos do mesmo tipo de dado.
- Na prática, as matrizes formam tabelas na memória.

Matrizes

- Exemplo de declaração de matriz com 2 dimensões usando linguagem C.

int matriz[3][3];

0 0	0 1	0 2
1 0	1 1	1 2
2 0	2 1	2 2

Matrizes

```
4   #include <stdio.h>
5   int main() {
6       int vetor[3][3];
7       vetor[0][0] = 10;
8       vetor[0][1] = 20;
9       vetor[0][2] = 30;
10      vetor[1][0] = 40;
11      vetor[1][1] = 50;
12      vetor[1][2] = 60;
13      vetor[2][0] = 70;
14      vetor[2][1] = 80;
15      vetor[2][2] = 90;
16      return 0;
17  }
```

10	20	30
40	50	60
70	80	90

```c
#include <stdio.h>
int main() {
    int vetor[3][3], i, j, valor = 10;;
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            vetor[i][j] = valor;
            valor += 10;
        }
    }
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            printf(" %d ",  vetor[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

10	20	30
40	50	60
70	80	90

Exemplo

- Crie um programa que leia uma matriz 3x3 de números inteiros e apresente os números do vetor e a sua posição.

```c
#include <stdio.h>
int main() {
    int vetor[3][3], i, j, valor = 10;;
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            printf("Matriz[%d %d] : ", i,j);
            scanf("%d", &vetor[i][j]);
        }
    }
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            printf("Matriz[%d %d] = %d ",i,j,vetor[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Exemplo

- Desenvolva um programa que leia uma matriz 3x3 de números inteiros e apresente os valores da diagonal principal.

0 0	0 1	0 2
1 0	1 1	1 2
2 0	2 1	2 2

```c
#include <stdio.h>
int main() {
    int vetor[3][3], i, j, valor = 10;;
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            printf("Matriz[%d %d] : ", i,j);
            scanf("%d", &vetor[i][j]);
        }
    }
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            if(i == j){
                printf(" %d ",vetor[i][j]);
            }
        }
    }
    return 0;
}
```

Exemplo

- Desenvolva um programa que leia uma matriz 3x3 de números inteiros e apresente os valores acima da diagonal principal.

0 0	0 1	0 2
1 0	1 1	1 2
2 0	2 1	2 2

```c
#include <stdio.h>
int main() {
    int vetor[3][3], i, j, valor = 10;;
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            printf("Matriz[%d %d] : ", i,j);
            scanf("%d", &vetor[i][j]);
        }
    }
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            if(i < j){
                printf(" %d ",vetor[i][j]);
            }
        }
    }
    return 0;
}
```

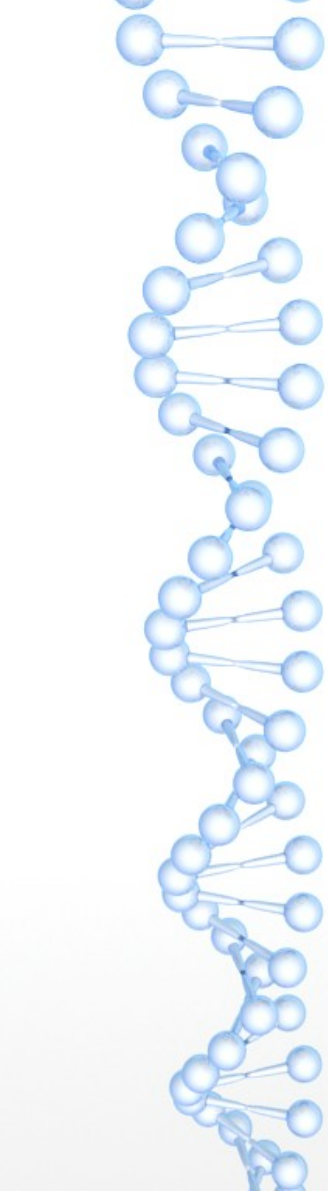
Exemplo

- Desenvolva um programa que leia uma matriz 3x3 de números inteiros, calcule e apresente o maior e menor valor.

```c
#include <stdio.h>
int main() {
    int vetor[3][3], i, j, maior, menor;
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            printf("Matriz[%d %d] : ", i,j);
            scanf("%d", &vetor[i][j]);
        }
    }
    maior = vetor[0][0];
    menor = vetor[0][0];
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            if(vetor[i][j] > maior){
                maior = vetor[i][j];
            }
            if(vetor[i][j] < menor){
                menor = vetor[i][j];
            }
        }
    }
    printf("Maior = %d | Menor = %d \n", maior, menor);
    return 0;
}
```

Exemplos

- Desenvolva um programa que leia uma matriz 3x3 de números inteiros, troque a primeira linha com a última.



```c
#include <stdio.h>
int main() {
    int vetor[3][3], i, j, aux;
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            printf("Matriz[%d %d] : ", i,j);
            scanf("%d", &vetor[i][j]);
        }
    }
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            if ( i == 0){
                aux = vetor[i][j];
                vetor[i][j] = vetor[2][j];
                vetor[2][j] = aux;
            }
        }
    }
    for(i=0; i<3; i++){
        for(j=0; j<3; j++){
            printf(" %d ", vetor[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Exercícios

- Vetores
  - Crie um vetor A de 10 posições.
  - Armazene em A uma sequencia de 1 a 10.
  - Crie um vetor B de 10 posições que recebe o dobro de A.
  - Crie um vetor C de 10 posições que recebe o produto de A com B.
  - Apresente os valores contidos em C.
  - Crie um vetor D de 50 posições que recebe os números pares contidos em A.

Exercícios

- Matriz
  - Crie uma matriz A com as dimensões (4x4) de números inteiros.
  - Preencher a segunda coluna da matriz A com o valor 1 (um).
  - Preencher a diagonal principal com valor zero (0)
  - Crie uma matriz B que contenha o dobro dos valores contidos na matriz A.
  - Apresente na tela a matriz A e a matriz B.

Exercícios

- Resolva 20 problemas propostos do capítulo (Vetores) e 20 problemas propostos do capítulo (Matrizes) do livro “fundamentos da programação de computadores”.

Estruturas - Registros
struct

- São variáveis compostas heterogêneas.
- São conjuntos de dados logicamente relacionados, mas de tipos diferentes (inteiro, real, string, etc.)
- Os elementos dos registros são chamados de campos.
- Exemplo: Dados sobre funcionários de uma empresa:
  - Nome
  - Idade
  - Salário

Declaração

```
struct nome_do_tipo_do_registro {
   tipo1 campo1;
   tipo2 campo2;
   tipo3 campo3;
   ...
   tipon campon;
};
struct funcionario {
   char nome[50];
   int idade;
   float salario;
};
```

Acesso a campos de um registro

- Pode ser realizado através da seguinte sintaxe:
  - **nome_do_registro.nome_do_campo**
- Para uma variável f do tipo funcionario:
  - **struct funcionario f;**
- O campo nome é acessado assim:
  - **f.nome**

```c
#include <stdio.h>
#include <string.h>
struct funcionario {
    char nome[50];
    int idade;
    float salario;
};
int main() {
    struct funcionario f;
    strcpy(f.nome,"Rafael");
    f.idade = 24;
    f.salario = 6730;
    printf("Nome: %s\n", f.nome);
    printf("Idade: %d\n", f.idade);
    printf("Salario: %.2f\n", f.salario);
    return 0;
}
```

Funções especiais:
strcpy(destino, origem);
strcmp(s1, s2);

Vetor de Registros

- Declaração:
  - struct nome_do_registro nome_da_variavel[tamanho_do_vetor];
- Uso:
  - **nome_da_variavel[indice].nome_do_campo**

```c
#include <stdio.h>
#include <string.h>
struct pessoa {
    char nome[50];
    int idade;
};
int main() {
    struct pessoa p[2];
    strcpy(p[0].nome,"Rafael");
    p[0].idade = 25;
    strcpy(p[1].nome,"Maria");
    p[1].idade = 35;
    printf("Nome: %s - Idade: %d\n", p[0].nome, p[0].idade);
    printf("Nome: %s - Idade: %d\n", p[1].nome, p[1].idade);
    return 0;
}
```

Exercícios

- Considerando o registro de um produto de uma loja contendo as seguintes informações: descrição, valor.
- Fazer um programa que, dado o registro de 50 produtos, exiba-os na ordem inversa em que foram digitados.

```c
#include <stdio.h>
#include <string.h>
struct produto {
    char descricao[50];
    float valor;
};
int main(){
    struct produto p[5];
    int i;
    for(i=0; i<5; i++){
        printf("Nome do produto  : ");
        scanf("%[^\n]s", p[i].descricao);
        printf("Valor do produto : ");
        scanf("%f", &p[i].valor);
        getchar(); // Limpa o buffer de teclado
    }
    for(i=0; i<5; i++){
        printf("Descrição: %s - Valor R$ %.2f \n", p[i].descricao, p[i].valor);
    }
    return 0;
}
```

Exercícios

- Desenvolva um código em C que armazene em um registro os dados dos usuários de um sistema.
  - NOME
  - E-MAIL
  - TELEFONE
  - CPF