

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 contém 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.



```
6  #include <stdio.h>
7  int main() {
8      int vetor[10], i;
9      for(i=0; i<10; i++){
10         printf("Digite um numero : ");
11         scanf("%d", &vetor[i]);
12     }
13     for(i=0; i<10; i++){
14         printf("Valor: %d  Indice: %d \n", vetor[i], i);
15     }
16     return 0;
17 }
```





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

```
10  #include <stdio.h>
11  int main() {
12      int vetor[5], i;
13      for(i=0; i<5; i++){
14          printf("Digite um numero : ");
15          scanf("%d", &vetor[i]);
16      }
17      for(i=4; i>=0; i--){
18          printf(" %d ", vetor[i]);
19      }
20      return 0;
21  }
```




Exemplos

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




```
6  int main() {
7      int A[5], B[5], C[5], i;
8      for(i=0; i<5; i++){
9          printf("Vetor A[%d] = ", i);
10         scanf("%d", &A[i]);
11         printf("Vetor B[%d] = ", i);
12         scanf("%d", &B[i]);
13         C[i] = A[i] + B[i];
14     }
15     for(i=0; i<5; i++){
16         printf(" A[%d] + B[%d] = C[%d] \n", A[i], B[i], C[i]);
17     }
18     return 0;
19 }
```






Exemplos

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



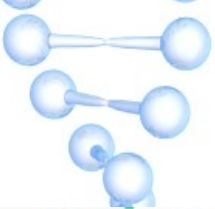
```
5  #include <stdio.h>
6  int main() {
7      int vetor[5], menorElemento, posicao = 0, i;
8      for(i=0; i<5; i++){
9          printf("Vetor[%d] = ", i);
10         scanf("%d", &vetor[i]);
11     }
12     menorElemento = vetor[0];
13     for(i=0; i<5; i++){
14         if(vetor[i] < menorElemento){
15             menorElemento = vetor[i];
16             posicao = i;
17         }
18     }
19     printf("Menor elemento é %d na posição %d do vetor", menorElemento, posicao);
20     return 0;
21 }
```






Exemplos

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



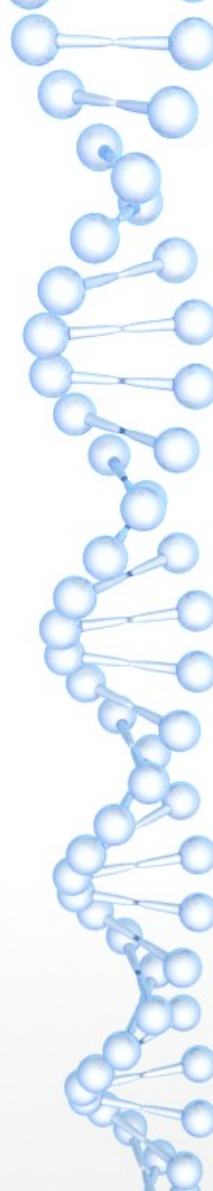
```
6  #include <stdio.h>
7  int main() {
8      int vetor[5], media = 0, i;
9      for(i=0; i<5; i++){
10         printf("Vetor[%d] = ", i);
11         scanf("%d", &vetor[i]);
12         media += vetor[i];
13     }
14     printf("Média : %.2f", (float) media / 5);
15     return 0;
16 }
```





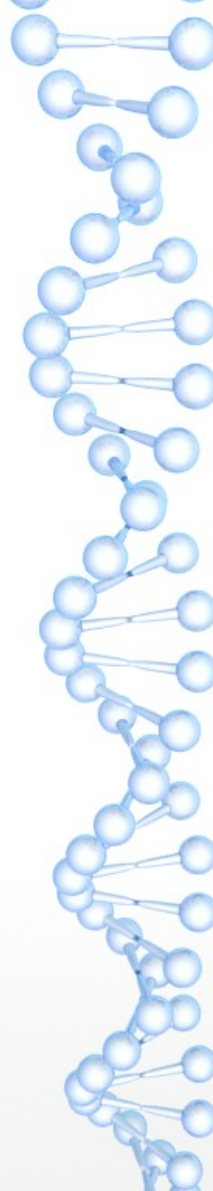
Exercícios

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



Ordenação em Vetores

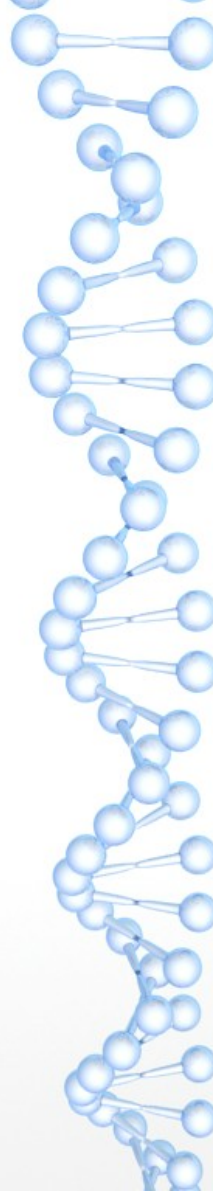
- Ordenar
 - Processo de reorganizar um conjunto de objetos em uma ordem ascendente ou descendente.
- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado.
 - Imagine um lista de telefone desordenada...



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 (Insertion 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

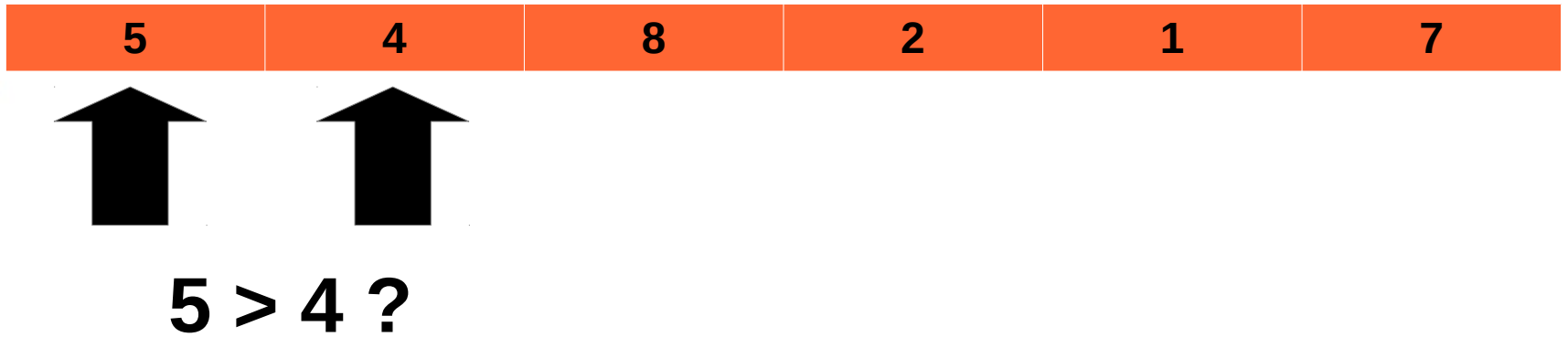
8

2

1

7

Bubble Sort



Bubble Sort



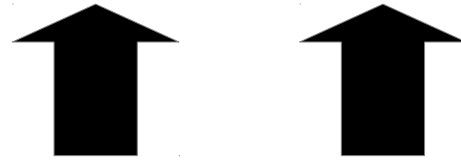
5 > 4 ?
SIM

Bubble Sort



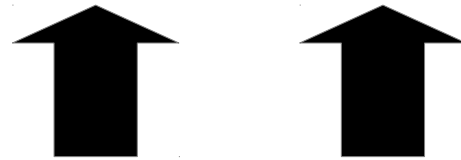
Troca

Bubble Sort



$5 > 8 ?$

Bubble Sort



$5 > 8 ?$

Não

Bubble Sort



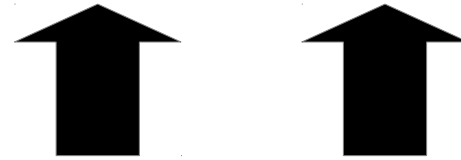
$8 > 2 ?$

Bubble Sort



Troca

Bubble Sort



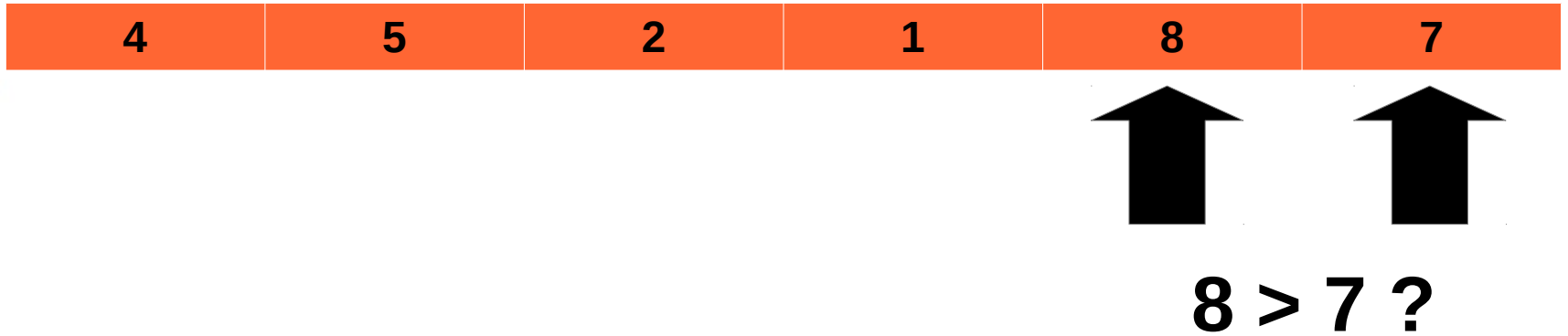
8 > 1 ?

Bubble Sort



Troca

Bubble Sort

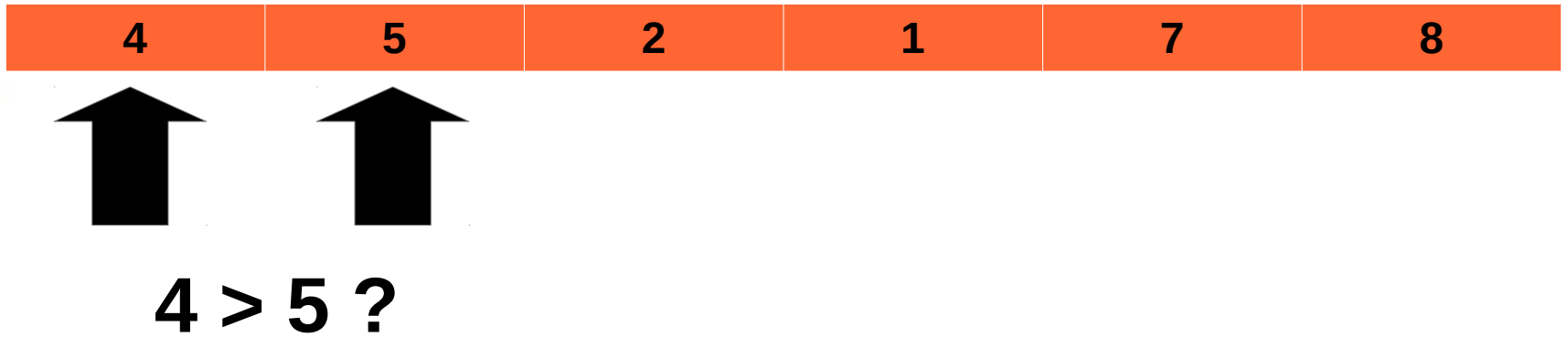


Bubble Sort



Troca

Bubble Sort



Bubble Sort



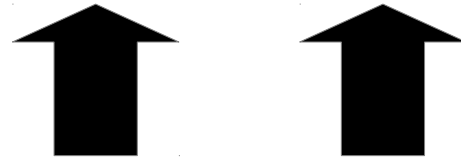
4 > 5 ?
Não

Bubble Sort



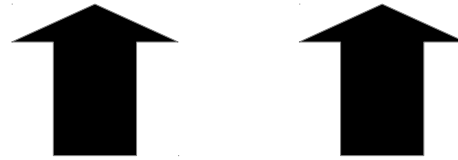
$5 > 2 ?$

Bubble Sort



Troca

Bubble Sort



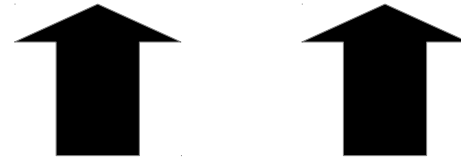
$5 > 1 ?$

Bubble Sort



Troca

Bubble Sort



1 > 7 ?

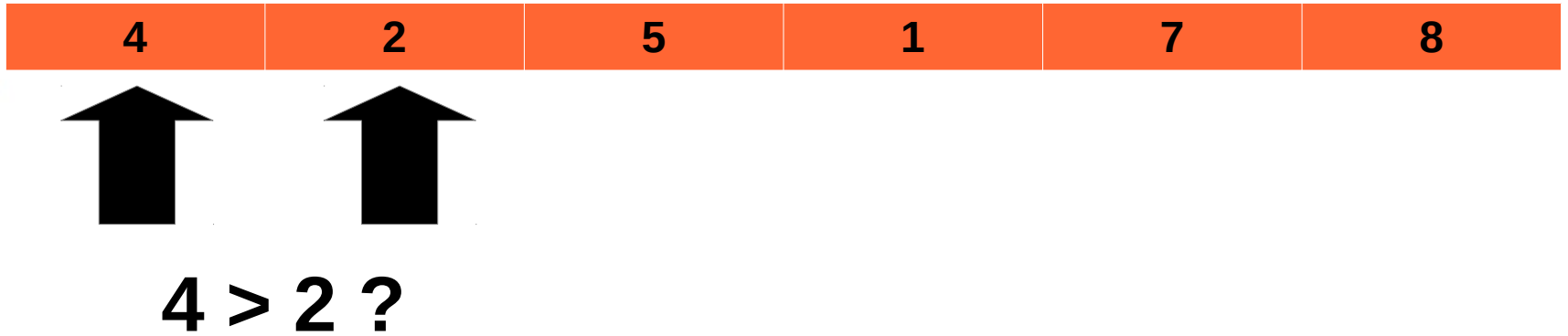
Bubble Sort



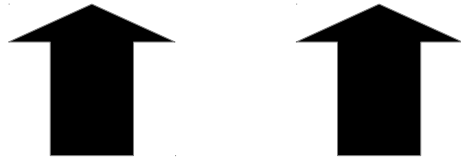
$1 > 7 ?$

Não

Bubble Sort



Bubble Sort



Troca

Bubble Sort



$4 > 5 ?$

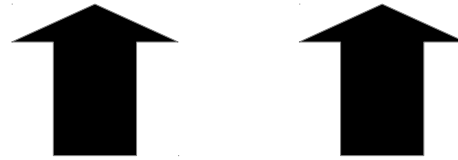
Bubble Sort



$4 > 5 ?$

Não

Bubble Sort



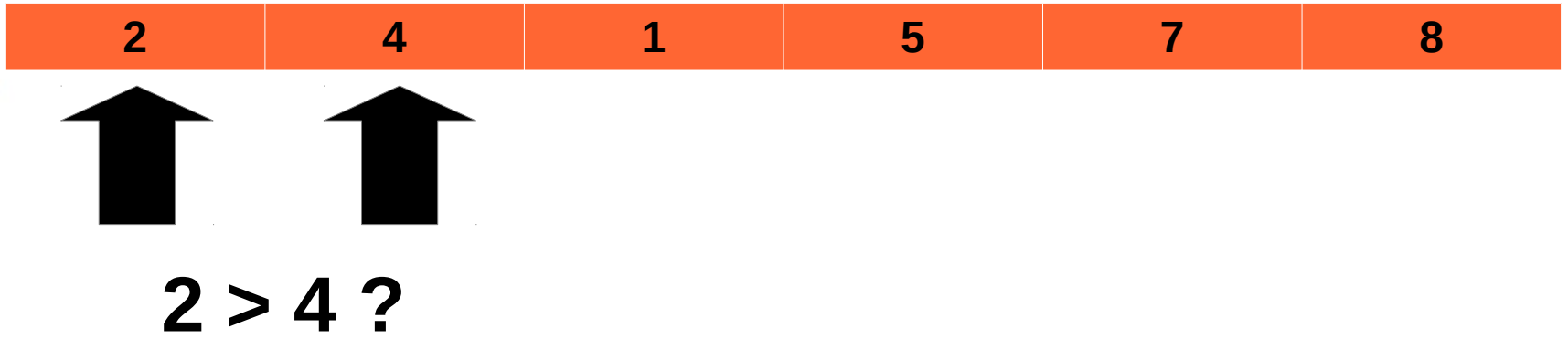
$5 > 1 ?$

Bubble Sort

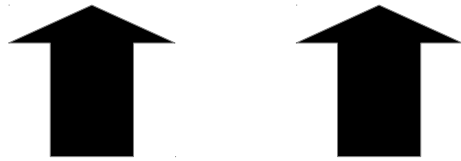


Troca

Bubble Sort



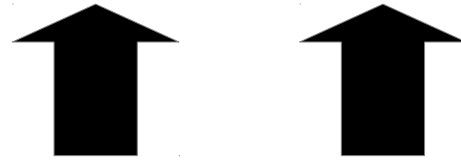
Bubble Sort



$2 > 4 ?$

Não

Bubble Sort



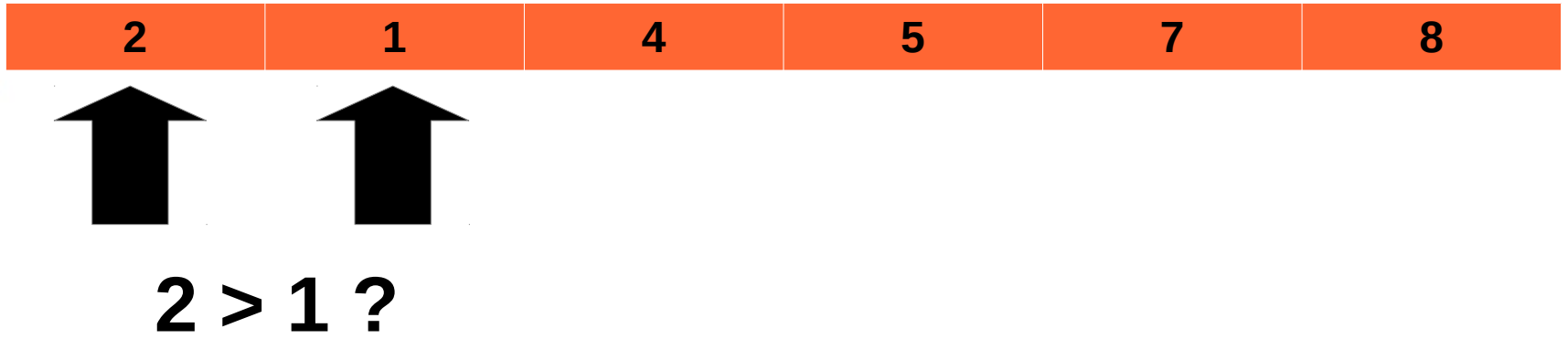
$4 > 1 ?$

Bubble Sort



Troca

Bubble Sort

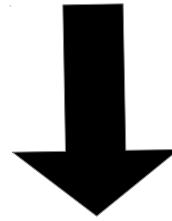


Bubble Sort



Troca

Bubble Sort



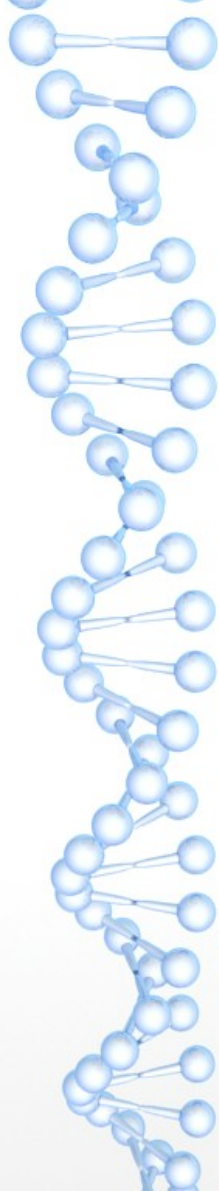


Atividade

- 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.



2020

```
1  #include <stdio.h>
2  main(){
3      int tamanhoVetor, i, vetor[10], controle, aux;
4      for (i=0; i<10;i++){
5          printf("Vetor[%d] : ", i+1);
6          scanf("%d", &vetor[i]);
7      }
8      tamanhoVetor = 10;
9      do{
10         controle = 0;
11         for (i=0; i<tamanhoVetor-1; i++){
12             if (vetor[i] > vetor[i+1]){
13                 aux = vetor[i+1];
14                 vetor[i+1] = vetor[i];
15                 vetor[i] = aux;
16                 controle = 1;
17             }
18         }
19         tamanhoVetor--;
20     }while(controle != 0);
21     for (i=0; i<10;i++){
22         printf(" %d ", vetor[i]);
23     }
24 }
```

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



```
1 | #include <stdio.h>
2 | main(){
3 |     int i, vetor[10], valor;
4 |     for (i=0; i<10;i++){
5 |         printf("Vetor[%d] : ", i);
6 |         scanf("%d", &vetor[i]);
7 |     }
8 |     printf("Digite o valor para procurar : ");
9 |     scanf("%d", &valor);
10 |    for (i=0; i<10;i++){
11 |        if(vetor[i] == valor){
12 |            printf("Vetor[%d] = %d \n", i, vetor[i]);
13 |        }
14 |    }
15 | }
```



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

```

4  #include <stdio.h>
5  int main() {
6      int vetor[3][3], i, j, valor = 10;;
7      for(i=0; i<3; i++){
8          for(j=0; j<3; j++){
9              vetor[i][j] = valor;
10             valor += 10;
11         }
12     }
13     for(i=0; i<3; i++){
14         for(j=0; j<3; j++){
15             printf(" %d ", vetor[i][j]);
16         }
17         printf("\n");
18     }
19     return 0;
20 }

```

10	20	30
40	50	60
70	80	90



Exemplo

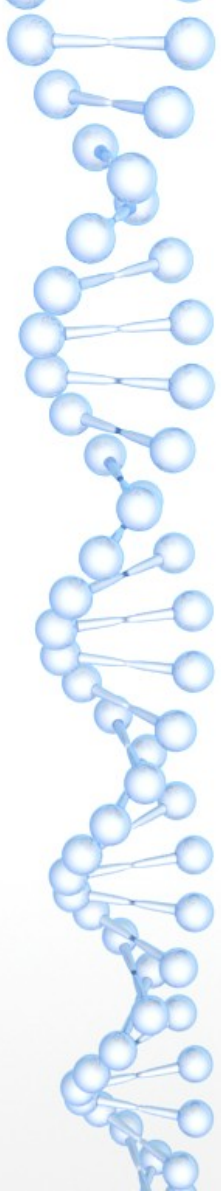
- Crie um programa que leia uma matriz 3x3 de números inteiros e apresente os números do vetor e a sua posição.

```
4  #include <stdio.h>
5  int main() {
6      int vetor[3][3], i, j, valor = 10;;
7      for(i=0; i<3; i++){
8          for(j=0; j<3; j++){
9              printf("Matriz[%d %d] : ", i,j);
10             scanf("%d", &vetor[i][j]);
11         }
12     }
13     for(i=0; i<3; i++){
14         for(j=0; j<3; j++){
15             printf("Matriz[%d %d] = %d ", i,j, vetor[i][j]);
16         }
17         printf("\n");
18     }
19     return 0;
20 }
```

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

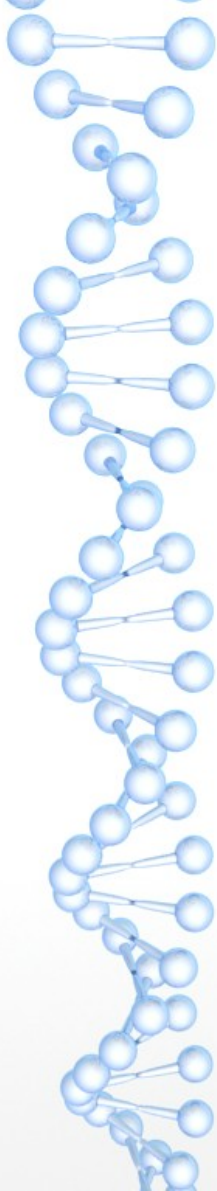


```
4  #include <stdio.h>
5  int main() {
6      int vetor[3][3], i, j, valor = 10;;
7      for(i=0; i<3; i++){
8          for(j=0; j<3; j++){
9              printf("Matriz[%d %d] : ", i,j);
10             scanf("%d", &vetor[i][j]);
11         }
12     }
13     for(i=0; i<3; i++){
14         for(j=0; j<3; j++){
15             if(i == j){
16                 printf(" %d ",vetor[i][j]);
17             }
18         }
19     }
20     return 0;
21 }
```

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

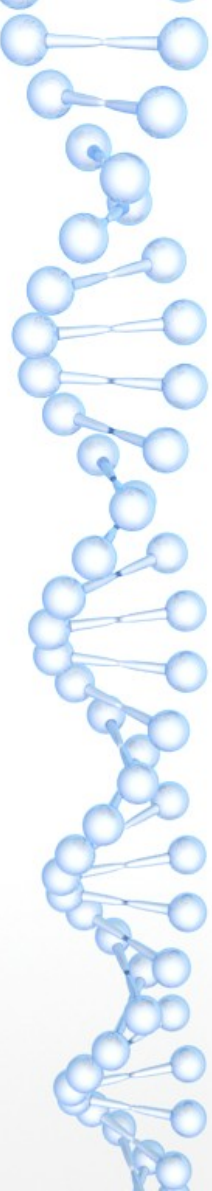


```
4  #include <stdio.h>
5  int main() {
6      int vetor[3][3], i, j, valor = 10;;
7      for(i=0; i<3; i++){
8          for(j=0; j<3; j++){
9              printf("Matriz[%d %d] : ", i,j);
10             scanf("%d", &vetor[i][j]);
11         }
12     }
13     for(i=0; i<3; i++){
14         for(j=0; j<3; j++){
15             if(i < j){
16                 printf(" %d ",vetor[i][j]);
17             }
18         }
19     }
20     return 0;
21 }
```



Exemplo

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

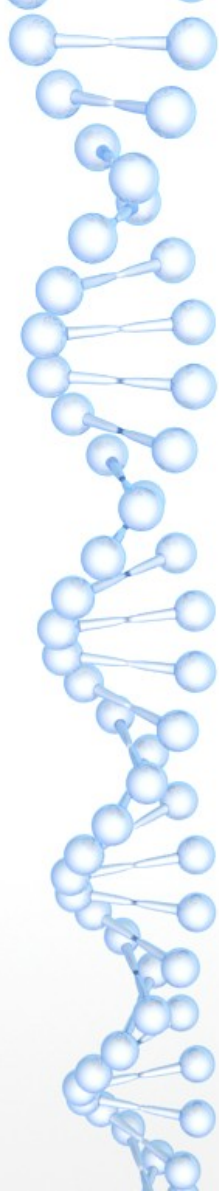


```
5  #include <stdio.h>
6  int main() {
7      int vetor[3][3], i, j, maior, menor;
8      for(i=0; i<3; i++){
9          for(j=0; j<3; j++){
10             printf("Matriz[%d %d] : ", i,j);
11             scanf("%d", &vetor[i][j]);
12         }
13     }
14     maior = vetor[0][0];
15     menor = vetor[0][0];
16     for(i=0; i<3; i++){
17         for(j=0; j<3; j++){
18             if(vetor[i][j] > maior){
19                 maior = vetor[i][j];
20             }
21             if(vetor[i][j] < menor){
22                 menor = vetor[i][j];
23             }
24         }
25     }
26     printf("Maior = %d | Menor = %d \n", maior, menor);
27     return 0;
28 }
```



Exemplos

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



2020

```
5  #include <stdio.h>
6  int main() {
7      int vetor[3][3], i, j, aux;
8      for(i=0; i<3; i++){
9          for(j=0; j<3; j++){
10             printf("Matriz[%d %d] : ", i,j);
11             scanf("%d", &vetor[i][j]);
12         }
13     }
14     for(i=0; i<3; i++){
15         for(j=0; j<3; j++){
16             if ( i == 0){
17                 aux = vetor[i][j];
18                 vetor[i][j] = vetor[2][j];
19                 vetor[2][j] = aux;
20             }
21         }
22     }
23     for(i=0; i<3; i++){
24         for(j=0; j<3; j++){
25             printf(" %d ", vetor[i][j]);
26         }
27         printf("\n");
28     }
29     return 0;
30 }
```



Exercícios

- Vetores

- Crie um vetor A de 10 posições.
- Armazene em A uma sequência 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

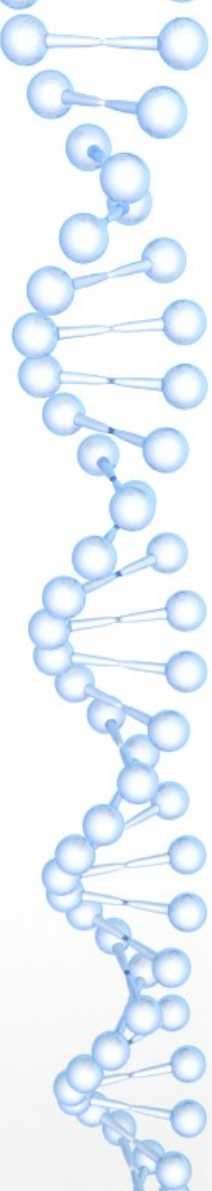
- Resolva 20 problemas propostos do capítulo (Vetores) e 20 problemas propostos do capítulo (Matrizes) do livro “fundamentos da programação de computadores”.



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

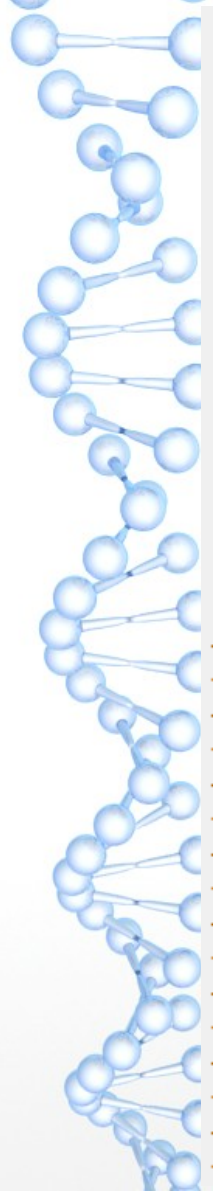


```
struct nome_do_tipo_do_registro {  
    tipo1 campo1;  
    tipo2 campo2;  
    tipo3 campo3;  
    ...  
    tipon campon;  
};  
  
struct funcionario {  
    char nome[50];  
    int idade;  
    float salario;  
};
```



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**




```
1 #include <stdio.h>
2 #include <string.h>
3 struct funcionario {
4     char nome[50];
5     int idade;
6     float salario;
7 };
8 int main() {
9     struct funcionario f;
10    strcpy(f.nome, "Rafael");
11    f.idade = 24;
12    f.salario = 6730;
13    printf("Nome: %s\n", f.nome);
14    printf("Idade: %d\n", f.idade);
15    printf("Salario: %.2f\n", f.salario);
16    return 0;
17 }
```

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`**



```
1 #include <stdio.h>
2 #include <string.h>
3 struct pessoa {
4     char nome[50];
5     int idade;
6 };
7 int main() {
8     struct pessoa p[2];
9     strcpy(p[0].nome, "Rafael");
10    p[0].idade = 25;
11    strcpy(p[1].nome, "Maria");
12    p[1].idade = 35;
13    printf("Nome: %s - Idade: %d\n", p[0].nome, p[0].idade);
14    printf("Nome: %s - Idade: %d\n", p[1].nome, p[1].idade);
15    return 0;
16 }
```




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.

```
1  #include <stdio.h>
2  #include <string.h>
3  struct produto {
4      char descricao[50];
5      float valor;
6  };
7  int main(){
8      struct produto p[5];
9      int i;
10     for(i=0; i<5; i++){
11         printf("Nome do produto : ");
12         scanf("%[^\n]s", p[i].descricao);
13         printf("Valor do produto : ");
14         scanf("%f", &p[i].valor);
15         getchar(); // Limpa o buffer de teclado
16     }
17     for(i=0; i<5; i++){
18         printf("Descrição: %s - Valor R$ %.2f \n", p[i].descricao, p[i].valor);
19     }
20     return 0;
21 }
```



Exercícios

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

