

# Mutirão de Programação em C

Aula 4 e 5 – Prática de programação em C

# Objetivos

- Revisão aulas C
- Exercícios
- Converter um algoritmo em código C

# Alguma dúvida?

- Exercícios aula 1
  - Variáveis, impressão formatada
- Exercícios aula 2
  - Vetores, matrizes, funções, strings
- Quizzes aula 3
  - Entendimento de ponteiros

# Ponteiros e vetores

- Material da aula 3: ponteiros

# Exercícios

1. O período  $T$  de um pêndulo de comprimento  $L$  é dado por:

$$T = 2\pi \sqrt{\frac{L}{g}}$$

onde  $g = 9,8 \text{ m/s}^2$ . Faça uma função que recebe o comprimento de um pêndulo e retorna seu período de oscilação.

2. Faça uma função que recebe o comprimento de um pêndulo e retorna sua frequência de oscilação. Use a função do item anterior.

# Exercícios

3. Agora faça uma função que recebe o comprimento de um pêndulo e mais dois ponteiros:

- um ponteiro para uma variável que receberá o valor calculado do período de oscilação,
- outro ponteiro para uma variável que receberá a frequência de oscilação.

```
float compute_pendulum_swing(float L, float *T, float *f);
```

Use as funções anteriores.

# Exercícios

## 4. FizzBuzz!

Escreva um programa que imprima os números contidos no intervalo de 1 a 100, com as seguintes observações:

- Quando o número for múltiplo de 3 o programa deve imprimir a palavra "Fizz" ao invés do número.
- Quando o número for múltiplo de 5 o programa deve imprimir a palavra "Buzz" ao invés do número.
- Quando o número for ao mesmo tempo múltiplo de 3 e 5 o programa deve imprimir FizzBuzz ao invés do número

# Exercícios

## 5. Números malucos do Luciano!

Um número de Harshad é um número inteiro que é divisível pela soma de seus dígitos. Exemplos:

- 3 é divisível pela soma de seus dígitos que é 3
- 27 é divisível pela soma dos seus dígitos que é 9
- 204 é divisível por 6

Escreva uma função que

- recebe um número inteiro não-negativo e
- retorna 1 se o número passado é um número de Harshad ou
- retorna 0 caso contrário.

Se o número passado for 0 a função também deve retornar 0.

Escreva também o código de teste para esta função.



# Exercícios

6. Escreva uma função que intercala dois vetores de números inteiros. A função tem a seguinte declaração:

```
void intercala(int a[], int b[], int n, int c[]);
```

onde  $a[]$  e  $b[]$  são vetores de um tamanho desconhecido,  $n$  é o número de elementos “válidos” de  $a[]$  e  $b[]$  (o mesmo valor de  $n$  vale para ambos), e  $c[]$  é o vetor destino. Assuma que  $c[]$  tem espaço reservado para pelo menos  $2n$  elementos.

# Exercícios

(Exercício 6, cont.)

Por exemplo: se

- $a[] = \{1, 2, 3, \text{🗑}, \text{🗑}\}$
- $b[] = \{99, 88, 77, \text{🗑}, \text{🗑}\}$
- $n = 3$
- $c[] = \{\text{🗑}, \text{🗑}, \text{🗑}, \text{🗑}, \text{🗑}, \text{🗑}, \text{🗑}, \text{🗑}, \text{🗑}, \text{🗑}, \text{🗑}\}$

então após a chamada da função intercala o resultado será:

- $c[] = \{1, 99, 2, 88, 3, 77, \text{🗑}, \text{🗑}, \text{🗑}, \text{🗑}\}$

# Exercícios

7. Faça um programa que recebe uma string com uma data no formato americano (mês/dia/ano hora:minuto AM/PM) e imprime a data no formato brasileiro (dia/mês/ano hora:minuto).

Exemplo:

02/29/2016 5:32 PM => 29/02/2016 17:32

# Exercícios

8. Dada uma matriz quadrada  $A_{m \times m}$
- Escreva uma função que recebe A e retorne o maior elemento da diagonal principal
  - Escreva uma função que recebe A e retorne 1 se ela for triangular superior, ou zero caso contrário

# Gerando números (pseudo-)aleatórios

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX_NUMBER 100

int main(void) {
    int i;

    srand(time(NULL));
    i = rand() % MAX_NUMBER;

    printf("%d\n", i);

    return 0;
}
```

# Gerando números (pseudo-)aleatórios

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

```
#define MAX_NUMBER 100
```

```
int main(void) {
```

```
    int i;
```

```
    srand(time(NULL));
```

```
    i = rand() % MAX_NUMBER;
```

```
    printf("%d\n", i);
```

```
    return 0;
```

```
}
```

Truque para inicializar o gerador  
de números aleatórios

# Gerando números (pseudo-)aleatórios

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

```
#define MAX_NUMBER 100
```

```
int main(void) {
    int i;

    srand(time(NULL));
    i = rand() % MAX_NUMBER;

    printf("%d\n", i);

    return 0;
}
```

<http://www.cplusplus.com/reference/cstdlib/rand/>  
Retorna um inteiro entre 0 e RAND\_MAX

# Gerando números (pseudo-)aleatórios

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX_NUMBER 100
```

```
int main(void) {
    int i;

    srand(time(NULL));
    i = rand() % MAX_NUMBER;

    printf("%d\n", i);

    return 0;
}
```

Truque para obter um inteiro aleatório  
entre 0 e (MAX\_NUMBER - 1)



# Exercícios

9. Faça uma função que preenche um vetor com números aleatórios. A função recebe:
  - um vetor de inteiros,
  - um número de posições a preencher (menor que o tamanho do vetor, claro), e
  - um valor máximo para os números aleatórios (também conhecidos como randômicos).
  
10. Faça uma função que verifique se um vetor está ordenado ou não. Quais seriam os argumentos? Qual a melhor maneira de retornar essa informação?

# Algoritmos: de Português para C

- Problema: dado um vetor de inteiros e seu tamanho, ordenar o vetor.

# Algoritmos: de Português para C

“E se a gente fizesse o seguinte: a gente varre o vetor e vê se tem algum parzinho fora de ordem. Se tiver, troca os dois de lugar. Repete isso até que esteja tudo ordenado!”

# Algoritmos: de Português para C

“E se a gente fizesse o seguinte: a gente varre o vetor e vê se tem algum parzinho fora de ordem. Se tiver, troca os dois de lugar. Repete isso até que esteja tudo ordenado!”

# Algoritmos: de Português para C

- Repete isso até que esteja tudo ordenado
  - varre o vetor
    - vê se tem algum parzinho fora de ordem. Se tiver,
      - troca os dois de lugar

# Algoritmos: de Português para C

- Enquanto **não ordenado**
  - Para cada posição do vetor
    - Se posição atual e seguinte estão fora de ordem
      - troca os dois de lugar

# Algoritmos: de Português para C

- Enquanto **não ordenado**
  - Para cada posição do vetor
    - Se posição atual e seguinte estão fora de ordem
      - troca os dois de lugar

↓

IDÉIA: se não houve troca alguma,  
é porque está ordenado!

# Algoritmos: de Português para C


- Marca como não-ordenado
- Enquanto **marcado como não-ordenado**
  - Marca como ordenado (provisoriamente)
  - Para cada posição do vetor
    - Se posição atual e seguinte estão fora de ordem
      - Troca os dois de lugar
      - Marca como não-ordenado



# Algoritmos: de Português para C

- ordenado = 0
- Enquanto ordenado == 0:
  - ordenado = 1
  - Para cada posição 'i' do vetor
    - Se vetor[i] > vetor[i+1]
      - Troca os dois de lugar
      - ordenado = 0

# Algoritmos: de Português para C

- ordenado = 0
  - Enquanto ordenado == 0:
    - ordenado = 1
    - Para cada posição 'i' do vetor **exceto a última**
      - Se vetor[i] > vetor[i+1]:
        - Troca os dois de lugar
        - ordenado = 0
- 

# Algoritmos: de Português para C

Algoritmo ordena

Entrada: vetor  $v[]$ , tamanho  $n$

Saída: nada

Efeito: ordena os  $n$  primeiros elementos de  $v[]$

Procedimento:

- ordenado = 0
- Enquanto ordenado == 0:
  - ordenado = 1
  - Para  $i$  de 0 a  $(n-2)$ :
    - Se  $v[i] > v[i+1]$ :
      - Aux =  $v[i]$
      - $V[i] = v[i + 1]$
      - $V[i + 1] = aux$
      - ordenado = 0

} Troca os dois de lugar

# Resultado

```
void bubblesort(int vec[], int n) {
    int i, ordenado, aux;

    ordenado = 0;
    while (!ordenado) {
        ordenado = 1;
        for (i = 0; i < n - 1; i++) {
            if (vec[i] > vec[i + 1]) {
                aux = vec[i];
                vec[i] = vec[i + 1];
                vec[i + 1] = aux;
                ordenado = 0;
            }
        }
    }
}
```

# Atividade

- Dividir a sala em dois grupos
- Cada grupo irá implementar um algoritmo diferente de ordenação

# Exercício: Selection sort

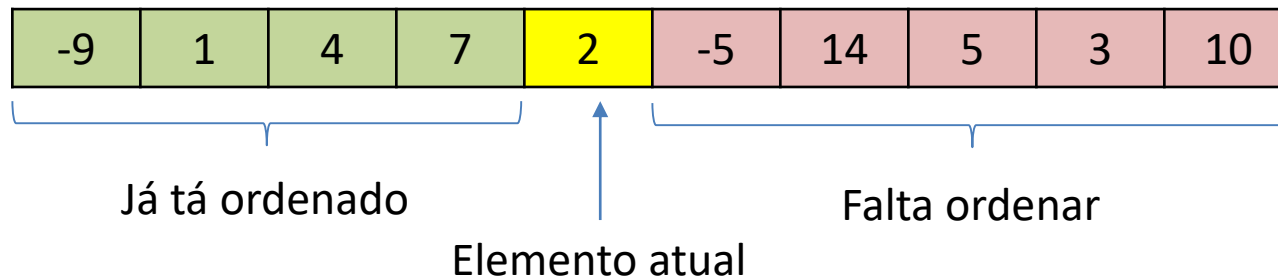
“E se a gente fizesse o seguinte:

- a gente varre o vetor e acha o menor de todos. Troca o menor de lugar com a primeira posição.
- Agora varre da segunda posição em diante procurando o menor. Troca esse novo menor de lugar com a segunda posição.
- Agora varre da terceira posição em diante procurando o menor. Troca esse novo menor de lugar com a terceira posição.
- Etc.

Fica fazendo isso até acabar o vetor!”

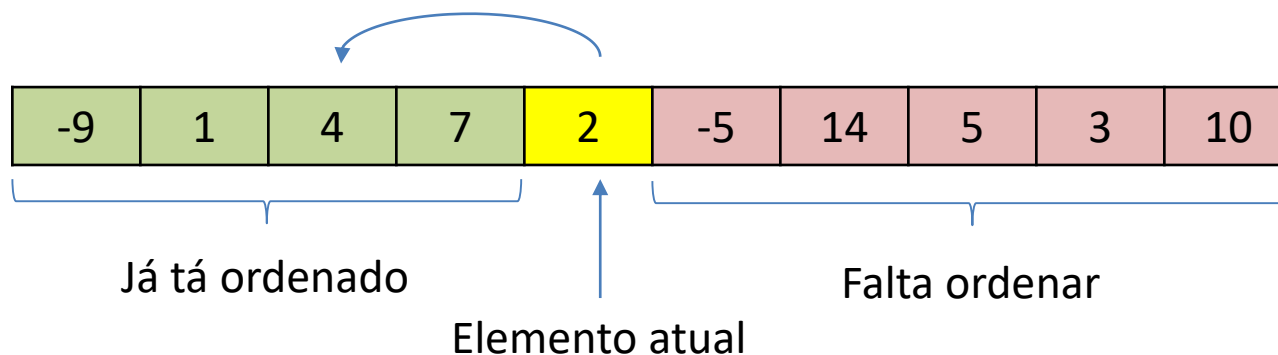
# Exercício: Insertion sort

“E se a gente fizesse o seguinte:”



# Exercício: Insertion sort

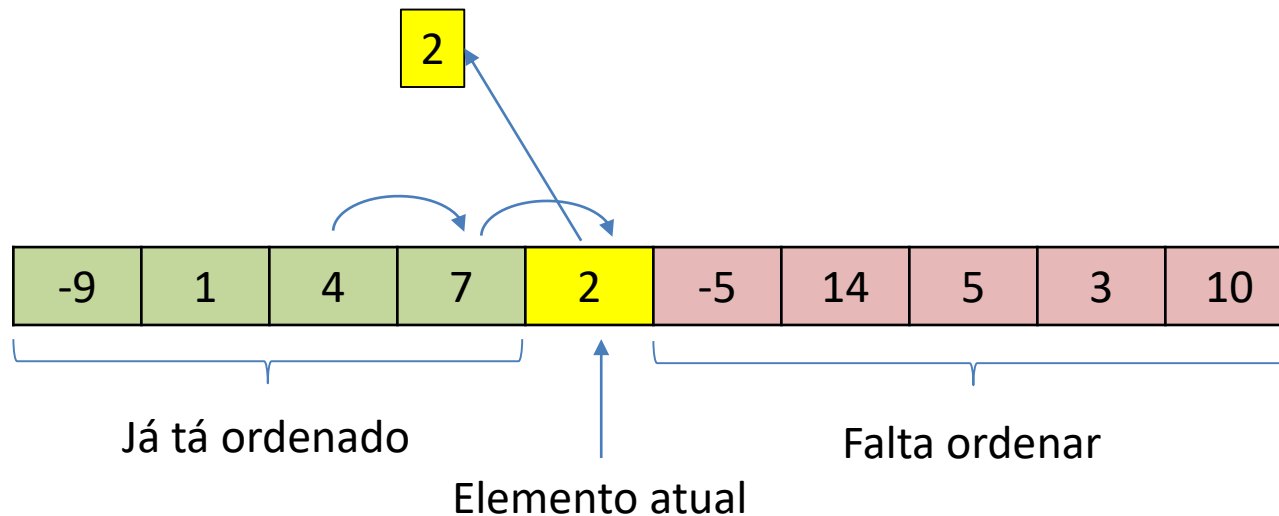
“Aí a gente acha onde inserir o elemento atual...”





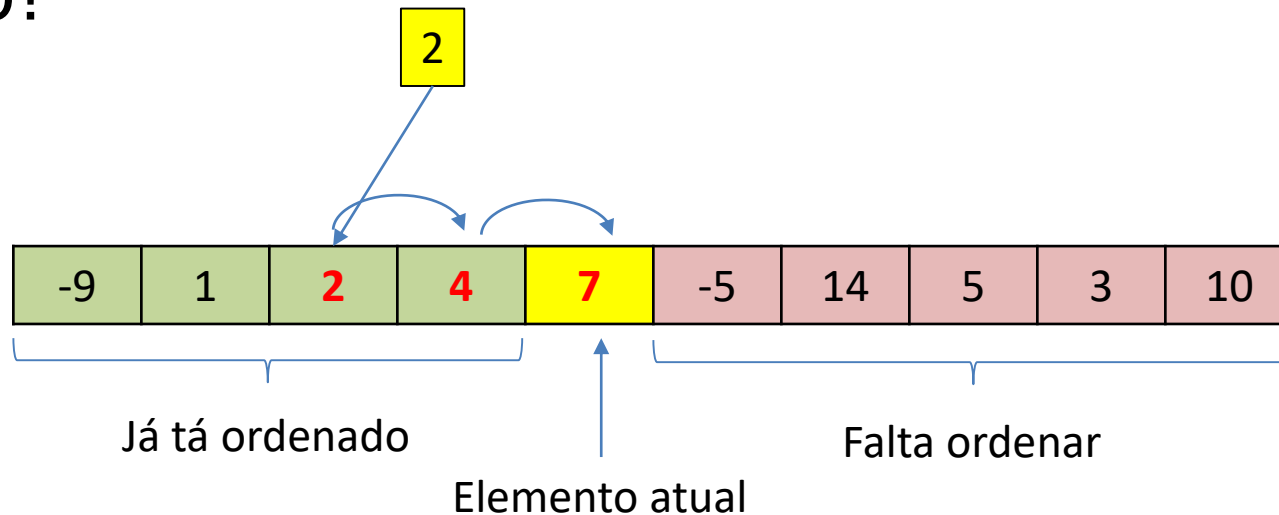
# Exercício: Insertion sort

“Desloca todo mundo para a frente...”



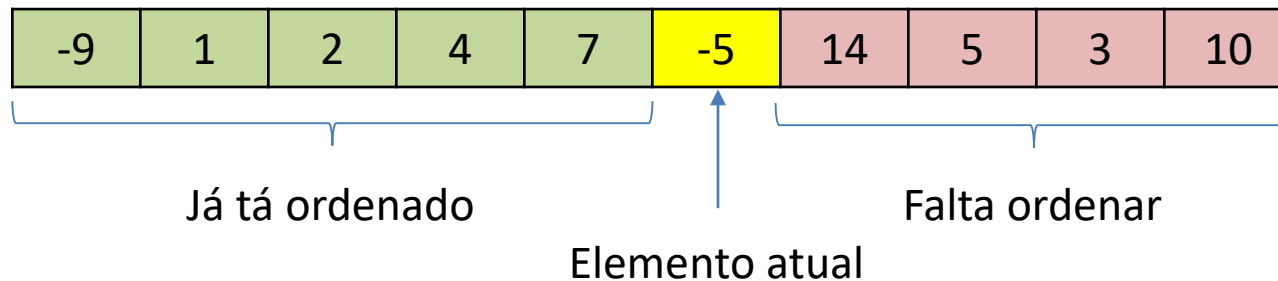
# Exercício: Insertion sort

“E copia o elemento atual na sua posição de direito!”



# Exercício: Insertion sort

“Repete até ficar ordenado!”



# Insper

[www.insper.edu.br](http://www.insper.edu.br)