

Estruturas de Dados

Aula 1 – Revisão de Algoritmos

Prof. Tales Nereu Bogoni
tales@unemat.br

Estrutura de um programa

```
#include  Diretivas do processador  
#define   Macros do processador
```

Declarações globais

- funções
- variáveis
- protótipos de funções

Função principal main

```
main ( )  
{  
    declarações locais  
    sentenças  
}
```

Definições de outras funções

```
func1 (...)  
{  
    ...  
}  
func2 (...)  
{  
    ...  
}  
...
```

Tipos Primitivos de Dados (variáveis)

- Inteiros

| Tipo | Num de bits | Formato para leitura com scanf | Intervalo | |
|--------------------|-------------|-----------------------------------|----------------|---------------|
| | | | Início | Fim |
| char | 8 | %c | -128 | 127 |
| unsigned char | 8 | %c | 0 | 255 |
| signed char | 8 | %c | -128 | 127 |
| int | 16 | %d | -32.768 | 32.767 |
| unsigned int | 16 | %u | 0 | 65.535 |
| signed int | 16 | %i | -32.768 | 32.767 |
| short int | 16 | %hi | -32.768 | 32.767 |
| unsigned short int | 16 | %hu | 0 | 65.535 |
| signed short int | 16 | %hi | -32.768 | 32.767 |
| long int | 32 | %li | -2.147.483.648 | 2.147.483.647 |
| signed long int | 32 | %li | -2.147.483.648 | 2.147.483.647 |
| unsigned long int | 32 | %lu | 0 | 4.294.967.295 |

Tipos Primitivos de Dados (variáveis)

- Ponto flutuante

| | | | | |
|--------------------------|--------|------------------|------------------------|------------------------|
| <code>float</code> | 32 | <code>%f</code> | <code>3,4E-38</code> | <code>3.4E+38</code> |
| <code>double</code> | 64 | <code>%lf</code> | <code>1,7E-308</code> | <code>1,7E+308</code> |
| <code>long double</code> | 80/128 | <code>%Lf</code> | <code>3,4E-4932</code> | <code>3,4E+4932</code> |

- Lógicos

- `bool` 8 `%d` `true/false`

- String (não é primitivo, mas muito usado em c++)

Entrada de dados pelo teclado

- Entrada padrão em C
scanf("expressão", variáveis);

Exemplo:

```
scanf("%f", &salario);
```

- Entrada padrão em c++
cin >> variável

Exemplo:

```
cin >> nr;
```

- Entrada de string em C
gets(variável);

Exemplo:

```
gets(nome);
```

- Entrada de string em c++
cin.getline(variável[], tamanho);
getline(cin, variavelstring);

Exemplo:

```
cin.getline(nomecompleto, 50);  
getline (cin, nomestring);
```

Saída de dados para o monitor

- Saída formatada (C)

printf("expressão", variáveis);

Exemplo:

```
printf("%d\n%f\n%c\n",a,b,c);
```

- Saída padrão (C++)

cout << dados

Exemplo:

```
cout << "Num" << n << endl;
```

- Saída formatada (C++)

cout << formato << dados

Exemplo:

```
cout << fixed << setprecision(2) << b << endl;
```

Estruturas condicionais (if)

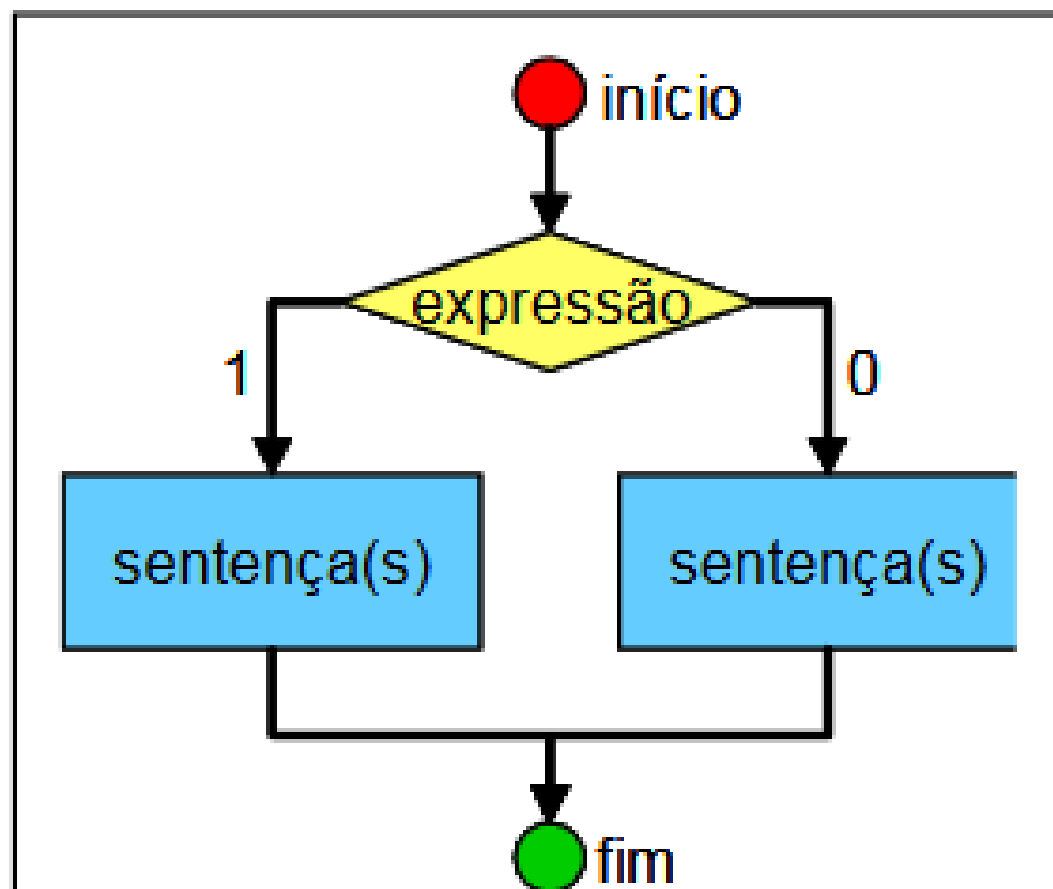


Figura 4 – Fluxo da estrutura condicional if...else

Sintaxe:

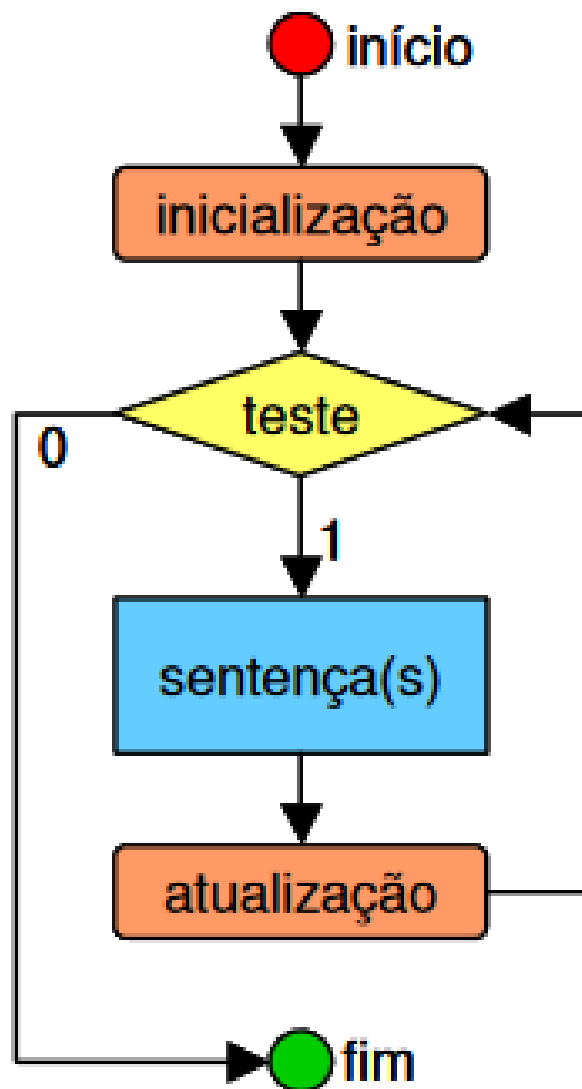
```

if (expressão) {
    sentença;
    sentença;
    ...
} else {
    sentença;
    sentença;
    ...
}
  
```

Figura 5 – Sintaxe da estrutura condicional if...else

Estruturas de repetição (For)

<https://www.ic.unicamp.br/~wainer/cursos/2s2011/Cap06-RepeticaoControle-texto.pdf>

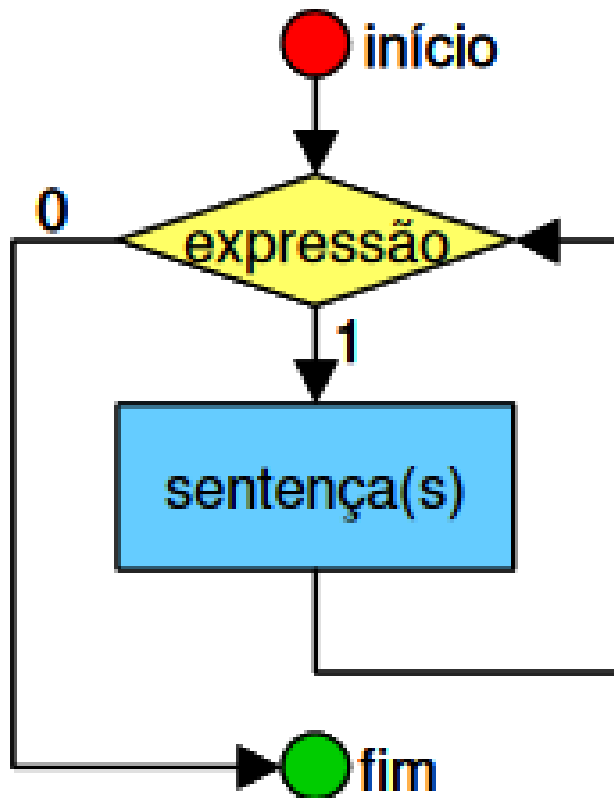


Sintaxe:

```
for (inicialização;  
    teste;  
    atualização) {  
    sentença;  
    sentença;  
    ...  
}
```

Figura 8 – Sintaxe da estrutura de repetição for

Estruturas de repetição (While)

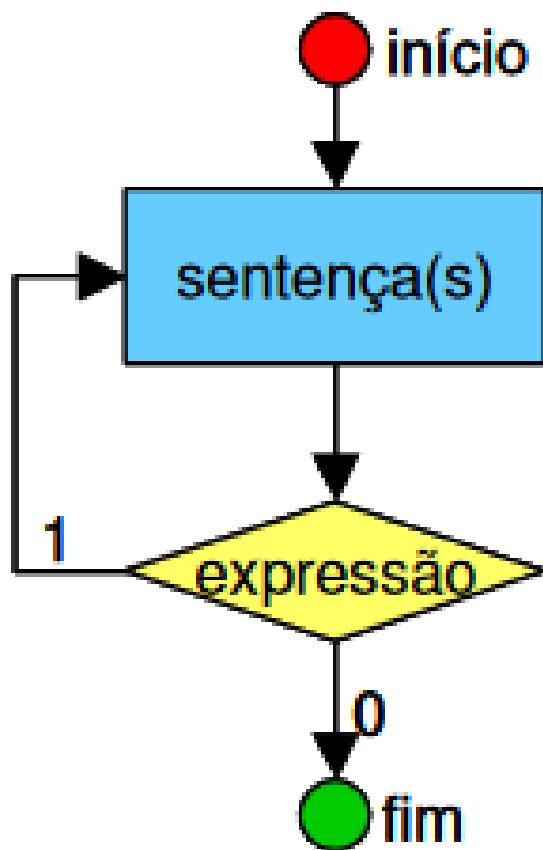


Sintaxe:

```
while (expressão) {  
    sentença;  
    sentença;  
    ...  
}
```

Figura 2 – Sintaxe da estrutura de repetição while

Estruturas de repetição (do ... while)



Sintaxe:

```
do {  
    sentença;  
    sentença;  
    ...  
} while (expressão);
```

Figura 5 – Sintaxe da estrutura de repetição do...while

Arrays unidimensionais (Vetores)

<https://www.ic.unicamp.br/~wainer/cursos/2s2011/Cap07-Vetores-texto.pdf>

`int[10]`

| | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| v_0 | v_1 | v_2 | v_3 | v_4 | v_5 | v_6 | v_7 | v_8 | v_9 |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|

Figura 1 – Exemplo de vetor com 10 elementos

```
int valores[10];  
int indice;
```

```
printf("Escreva 10 números inteiros: ");  
for (indice = 0; indice < 10; indice++) {  
    scanf("%d", &valores[indice] );  
}
```

```
printf("Valores em ordem reversa:\n");  
for (indice = 9; indice >= 0; indice--) {  
    printf("%d ", valores[indice]);  
}
```

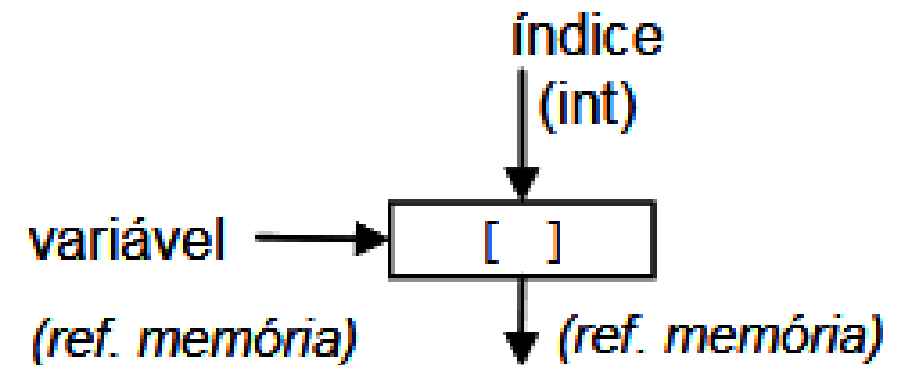


Figura 2 – Operador índice

Arrays multidimensionais (matrizes)

```
int linha, coluna;  
int matriz[4][10];  
...  
for (linha = 0; linha < 4; linha++) {  
    for (coluna = 0; coluna < 10; coluna++) {  
        printf("%d ", matriz[linha][coluna]);  
    }  
    printf("\n");  
}
```

int[4][10]

| | | | | | | | | | | |
|---------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| int[10] | a ₀₀ | a ₀₁ | a ₀₂ | a ₀₃ | a ₀₄ | a ₀₅ | a ₀₆ | a ₀₇ | a ₀₈ | a ₀₉ |
| int[10] | a ₁₀ | a ₁₁ | a ₁₂ | a ₁₃ | a ₁₄ | a ₁₅ | a ₁₆ | a ₁₇ | a ₁₈ | a ₁₉ |
| int[10] | a ₂₀ | a ₂₁ | a ₂₂ | a ₂₃ | a ₂₄ | a ₂₅ | a ₂₆ | a ₂₇ | a ₂₈ | a ₂₉ |
| int[10] | a ₃₀ | a ₃₁ | a ₃₂ | a ₃₃ | a ₃₄ | a ₃₅ | a ₃₆ | a ₃₇ | a ₃₈ | a ₃₉ |

Figura 3 – Exemplo de uma matriz com 4 linhas e 10 colunas

Structs (Dados heterogêneos)

TipoAluno




| | |
|-------------------------------------|----------------------|
| <code>char nome[20]</code> | <input type="text"/> |
| <code>int registro_academico</code> | <input type="text"/> |
| <code>int codigo_curso</code> | <input type="text"/> |

Figura 1 – Exemplo de uma estrutura (dados do aluno)

```
typedef struct {
    double real;
    double imaginario;
} complexo;
```

Acesso aos dados

`aluno.registro_academico = 991122;`

 Variável da estrutura
  Separador
  Variável membro

Funções com retorno de valor

Exemplo:

Tipo do valor de retorno

```
float mediaMelhoresNotas(  
    float nota1, float nota2, float nota3) {  
  
    float media;  
    if ((nota1 <= nota2) && (nota1 <= nota3)) {  
        media = (nota2 + nota3) / 2.0f;  
    } else if ((nota2 <= nota1) && (nota2 <= nota3)) {  
        media = (nota1 + nota3) / 2.0f;  
    } else {  
        media = (nota1 + nota2) / 2.0f;  
    }  
  
    return media;  
}
```

Término da função, indicando qual valor deve ser retornado.

Funções com parâmetros por referência

```
void troca(int *a, int *b){  
    int temp;  
    temp=*a;  
    *a=*b;  
    *b=temp;  
}  
  
int main(){  
    int a=2,b=3;  
    printf("Antes de chamar a função : \na=%d\nb=%d\n",a,b);  
    troca(&a,&b);  
    printf("Depois de chamar a função: \na=%d\nb=%d\n",a,b);  
    return 0;  
}
```

Arquivos de Cabeçalho ("headers")

- Já sabemos que podemos incluir arquivos especiais em um programa usando a diretiva de compilação `#include`.
- Normalmente são usados os símbolos `<` e `>` para indicar o nome do arquivo que será “anexado” na compilação.
- É possível criar suas próprias bibliotecas, para isso deve ser criado um arquivo de *header* “.h”, e anexá-lo no arquivo da programação.
- Neste caso o `#include` deve ser usado com aspas “ ao invés dos sinais de `<` e `>`
- No arquivo Header deve m ser inseridos os cabeçalhos das funções e em um arquivo .cpp deve ser feito o detalhamento de cada função
- Exemplo de chamada de um arquivo Header
 - `#include "constantes.h"`

O arquivo .h

- funcoes.h

```
#ifndef FUNCOES_H
#define FUNCOES_H

//troca dois valores
void troca(int *a, int *b);
//maior entre dois valores
int maior(int a , int b);

#endif
```

O arquivo .cpp

- funcoes.cpp

```
#include "funcoes.h"
```

```
void troca(int *a, int *b) {  
    int temp = *a;  
    *a = *b;  
    *b = temp;  
}
```

```
int maior(int a , int b) {  
    if(a>b)  
        return a;  
    return b;  
}
```

Usando as funções em outras aplicações

- main.cpp

```
#include "funcoes.h"

int main() {
    cout << "O maior valor e " << maior(10,20) << endl;
    int x=23, y=15;
    cout << "No inicio:" << endl << "X=" << x << "    Y=" << y << endl;
    troca(&x, &y);
    cout << "Depois da troca:" << endl << "X=" << x << "    Y=" << y << endl;
}
```