2013

Decode Team

Instituto Politécnico de tomar

Portugol

Equivalências de estruturas entre Portugol e C

Índice

[Nota Geral: 3](#_Toc358247485)

[Algumas notas sobre C: 3](#_Toc358247486)

[Estrutura Início 3](#_Toc358247487)

[Início: 3](#_Toc358247488)

[Estrutura Fim 4](#_Toc358247489)

[Fim: 4](#_Toc358247490)

[Variáveis 4](#_Toc358247491)

[Equivalência entre TIPOS de variáveis 4](#_Toc358247492)

[Definição e atribuição de variáveis 4](#_Toc358247493)

[Se a variável não estiver definida em memória 4](#_Toc358247494)

[Se a variável estiver definida em memória 4](#_Toc358247495)

[Alguns exemplos de definição e atribuição de variáveis 5](#_Toc358247496)

[Estruturas input/output 7](#_Toc358247497)

[Input – Ler 7](#_Toc358247498)

[Se for a primeira vez a ler do teclado 7](#_Toc358247499)

[Se a variável não estiver definida em memória 7](#_Toc358247500)

[Se a variável já estiver definida em memória 7](#_Toc358247501)

[Output – Escrever 8](#_Toc358247502)

[Estruturas de Decisão 8](#_Toc358247503)

[Condição *“if”* e *“if else”* 8](#_Toc358247504)

[Exemplos práticos 9](#_Toc358247505)

[Condição *“while”* 9](#_Toc358247506)

[Condição *“do while”* 10](#_Toc358247507)

[Exemplos práticos 10](#_Toc358247508)

[Estrutura Conector 11](#_Toc358247509)

[Conector 11](#_Toc358247510)

[Funções 12](#_Toc358247511)

[Definir funções 12](#_Toc358247512)

[Definir função *Exemplo* sem parâmetros de entrada 12](#_Toc358247513)

[Definir função *Exemplo* com parâmetros de entrada 12](#_Toc358247514)

[Chamada de funções 12](#_Toc358247515)

[Estrutura de retorno 13](#_Toc358247516)

[Return 13](#_Toc358247517)

[Operadores 14](#_Toc358247518)

[Aritméticos 14](#_Toc358247519)

[Lógicos 14](#_Toc358247520)

[Relacionais 14](#_Toc358247521)

[ANEXO 15](#_Toc358247522)

[Algoritmo com o uso da condição “if” 15](#_Toc358247523)

[Fluxograma 15](#_Toc358247524)

[Código 15](#_Toc358247525)

[Esquema detalhado 16](#_Toc358247526)

[Algoritmo com o uso da condição “if else” 17](#_Toc358247527)

[Fluxograma 17](#_Toc358247528)

[Código: 17](#_Toc358247529)

[Esquema detalhado 18](#_Toc358247530)

[Algoritmo com o uso da condição “while” 19](#_Toc358247531)

[Fluxograma 19](#_Toc358247532)

[Código 19](#_Toc358247533)

[Esquema detalhado 20](#_Toc358247534)

[Algoritmo com o uso da condição “ do while” 21](#_Toc358247535)

[Fluxograma 21](#_Toc358247536)

[Código 21](#_Toc358247537)

[Esquema detalhado 22](#_Toc358247538)

[Algoritmo com o uso de uma função 23](#_Toc358247539)

[Fluxogramas 23](#_Toc358247540)

[Código 23](#_Toc358247541)

[Esquema detalhado 24](#_Toc358247542)

# Nota Geral:

Devido à especificação da linguagem, a tradução só é possível depois de ser efetuado o fluxograma.

# Algumas notas sobre C:

* Não é case sensitive, embora não convém colocar letras maiúsculas e minúsculas a vontade do fragues.
* Usa o ponto e virgula (;) para terminar uma linha de código.
* As funções não podem ser definidas antes do main.
* A linguagem permite fazer a inclusão de bibliotecas através da instrução include. As bibliotecas devem ser incluídas imediatamente antes do main (ver [*estrutura início*](#_Estrutura_Início)).
* A primeira função a ser codificada deve ser o início.

# Estrutura Início

## Início:

Início

#include <stdio.h>

#include <stdlib.h>

void main()

{

# Estrutura Fim

## Fim:

Fim

}

# Variáveis

## Equivalência entre TIPOS de variáveis

|  |  |  |
| --- | --- | --- |
| TIPO | Portugol | C |
| Inteiro | Inteiro | int |
| Real | Real | float |
| Texto | Texto | char txt [tamanho da string] |
| Caracter | Caracter | char |
| Lógico | Logico | int |

Tabela 1 - Tipos de variáveis

**Nota**: Strings em C, são arrays de caracteres.

## Definição e atribuição de variáveis

variavel <- expressao

### Se a variável não estiver definida em memória

**Passo 1**: Avaliar a expressão (VALOR).

**Passo 2**: Calcular Tipo do VALOR.

**Passo 3**: Declarar a variável: TIPO variavel = expressao;

### Se a variável estiver definida em memória

variavel = expressao;

### Alguns exemplos de definição e atribuição de variáveis

Existem duas formas de definir variáveis e proceder à sua atribuição.

* Float
  + *1 – Definir e atribuir variável em passos separados:*
  + Float variavel;
  + variavel=valor;
  + **Nota 2**: O *valor* é um número que pode ser com virgula.
* Int
  + *1 – Definir e atribuir variável em passos separados:*
  + Int variavel;
  + variavel=valor;
  + **Nota 1**: Este tipo de variável também é usado para valores logicos.
* char
  + *1 – Definir e atribuir variável no mesmo passo:*
  + char variavel =’X’;
  + *2 – Definir e atribuir variável em passos separados:*
  + Char variavel;
  + variavel=’X’;
  + **Nota 1**: tem que ser definido com letra minúscula
  + **Nota 2**: *X* é um caracter e deve estar dentro de pelicas.
* String
  + *1 – Definir variavel:*
  + char text[10]; ou char text[valor];
  + **Nota 1**: Onde o valor é um inteiro que pode ser defenido depois*.*
  + **Nota 2**: A string em C é um array de caracteres*.*

# Estruturas input/output

## Input – Ler

variavel

|  |  |
| --- | --- |
| Tipo | C |
| Real | %f |
| Texto | %c |
| Lógico | %i |
| INT | %i |
| Char | %c |

Tabela 2 - Tipo de variáveis para leitura

**Exemplo:** scanf ("tipo",&variavel) ou variavel = gets(string);

**Nota 1**: O "&" quer dizer que é para a "variavel "que vai ser guardado o valor inserido*.***Nota1:** No caso de ser uma string a instrução utlizada para ler é gets(string);

### Se a variável não estiver definida em memória

**Passo 1:** Identificar o tipo (TIPO) de dados que foi lido.

**Passo 2:** Definir a variável:

TIPO variavel;

**Passo 3:** scanf ("tipo",&variavel);

### Se a variável já estiver definida em memória

**Passo 1:** Realizar apenas o ***Passo 3*** do ponto anterior.

## Output – Escrever

expressao

**Para escrever no ecrã:** printf ("tipo",expressao); ou, no caso ser um array de chars ou seja string, vai ser: puts (expressao);.

# Estruturas de Decisão

## Condição *“if”* e *“if else”*

condição

FALSE

TRUE

Instruções 2

Instruções 1

Para TRUE, escrever:

If (*condição*) {

Instruções 1

Para FALSE:

Se Instruções 2 for igual a (conector) não fazer nada.

Senão, escrever:

} else {

Instruções 2

### Exemplos práticos

#### Condição “if”

if (n%2==0) {

printf("Par");

}

#### Condição “if else”

if (n % 2 == 0) {

printf("Par");

} else {

printf("Impar");

}

## Condição *“while”*

condição

true

false

instruções

While (*condição*){

Instruções

}

## Condição *“do while”*

instruções

TRUE

condição

FALSE

Instruções

}while(*condição*);

### Exemplos práticos

#### Condição “while”

while(i<=10){

printf("%f",i);

i++;

}

#### Condição “do while”

do{

scanf("%f",&i);

}while(i<0);

# Estrutura Conector

## Conector

Se for uma condição “*do while*” escrever:

do {

Senão, escrever:

}

# Funções

## Definir funções

Exemplo( a , b, . . .)

**Nota:** Depois da função ser executada pelo menos uma vez, o tipo de retorno das função RETURN\_TIPO e o TIPOx dos parametros pode ser identificado:

void NOME( TIPO1 a , TIPO2 b, . . .. )

{

### Definir função *Exemplo* sem parâmetros de entrada

void NOME () {

### Definir função *Exemplo* com parâmetros de entrada

TIPO\_DE\_RETORNO NOME (PARAMETROS) {

**TIPO** – Executa a função e calcula o tipo de retorno.

Consultar *tabela 1* no ponto [*Equivalência entre TIPOS de variáveis*](#_Equivalência_entre_TIPOS).

**NOME** – Nome dado à função.

**PARAMETRO** – Variável utilizada pela função para auxiliar o cálculo.

## Chamada de funções

variavel = NOME(PARAMETRO)

variavel = NOME(PARAMETRO);

**Nota:** A variavel em que vai ser guardado o retorno da função tem que ser do mesmo tipo que a função.

# Estrutura de retorno

## Return

expressao

return expressao;

# Operadores

### Aritméticos

|  |  |  |
| --- | --- | --- |
| Nome | Portugol | C |
| Adição | a + b | a + b |
| Subtração | a – b | a - b |
| Divisão | a / b | a / b |
| Multiplicação | a \* b | a \* b |
| Resto da divisão inteira | a % b | a % b |
| Potenciação | a ^ b | pow(base,expoente); |
| Concatenação de texto | , | , |

Tabela 3 - Equivalência de operadores aritméticos

### Lógicos

|  |  |  |
| --- | --- | --- |
| Nome | Portugol | C |
| Disjunção | a E b | a && b |
| Conjunção | a OU b | a || b |
| Negação | NAO a | ! a |

Tabela 4 - Equivalência de operadores lógicos

### Relacionais

|  |  |  |
| --- | --- | --- |
| Nome | Portugol | C |
| Igual | a = b | a == b |
| Diferente | a =/= b | a != b |
| Maior | a > b | a > b |
| Maior ou igual | a >= b | a >= b |
| Menor | a < b | a < b |
| Menor ou igual | a <= b | a <= b |

Tabela 5 - Equivalência de operadores relacionais

# ***ANEXO***

Para uma compreensão mais abrangente do uso das estruturas, ficam alguns exemplos mais extensivos, com o uso de várias estruturas em algoritmos completos.

## Algoritmo com o uso da condição “if”

**Problema:** Verificar se um número introduzido pelo utilizador é par.

### Fluxograma

Inicio

n

n%2=0

TRUE

FALSE

“Par”

Fim

### Código

#include <stdio.h>

int main()

{

int n;

scanf("%d", &n);

if (n%2==0){

printf("Par");

}

}

### Esquema detalhado

Fim

TRUE

“Par”

n%2=0

n

Inicio

FALSE

## Algoritmo com o uso da condição “if else”

**Problema:** Verificar se um número introduzido pelo utilizador é par ou ímpar.

### Fluxograma

### 

TRUE

Fim

“Par”

n%2=0

n

Inicio

FALSE

“Impar”

### Código:

#include <stdio.h>

int main()

{

int n;

scanf("%d", &n);

if (n%2==0){

printf("Par");

}else{

printf("Impar");

}

return 0;

}

### Esquema detalhado

Fim

“Impar”

FALSE

TRUE

“Par”

n%2=0

n

Inicio

## Algoritmo com o uso da condição “while”

**Problema:** Escrever um número de 1 a 10.

### Fluxograma

Inicio

i=1

i <= 10

FALSE

TRUE

i

i=i + 1

Fim

### Código

#include <stdio.h>

int main()

{

int i=1;

while(i<=10){

printf("%d",i);

i=i+1;

}

return 0;

}

### Esquema detalhado

TRUE

i=i + 1

i=1

Fim

i

i <= 10

Inicio

FALSE

## Algoritmo com o uso da condição “ do while”

**Problema:** Pedir um número positivo.

### Fluxograma

Inicio

i = 0

i

TRUE

i < 0

FALSE

Fim

### Código

#include <stdio.h>

int main()

{

int i=0;

do{

scanf("%d",&i);

}while(i<=10);

return 0;

}

### Esquema detalhado

TRUE

i

i = 0

Fim

i < 0

Inicio

FALSE

## Algoritmo com o uso de uma função

**Problema:**  Factorial de um número.

### Fluxogramas

**Código principal**

**Função fact(k)**

Inicio

fact(k)

TRUE

FALSE

k > 2

i = 5

j = fact ( i )

k \* fact(k - 1)

k

j

Fim

Fim

### Código

#include <stdio.h>

int fact(int k){

if (k>2){

k=k\*fact(k-1);

}

return k;

}

int main()

{

int i=5;

int j=fact(i);

printf ("%d",j);

return 0;

}

### Esquema detalhado

k \* fact(k - 1)

Inicio

Fim

Fim

**Função fact(k)**

k

k > 2

fact(k)

TRUE

FALSE

**Código principal**

j

j = fact ( i )

i = 5