

# Guia de Referência (resumo) para Linguagem de Programação C

(Prof. Bruno B. Boniati – Colégio Agrícola de Frederico Westphalen – Universidade Federal de Santa Maria)

## Estrutura básica de uma aplicação

```
/* Estrutura básica de uma aplicação */
```

```
#include <stdio.h> //standard input-output (biblioteca padrão de entrada/saída)
#include <stdlib.h> //standard library (biblioteca de propósito geral)
```

```
int a; //declaração de variáveis globais
```

```
float funcao_exemplo() {
    return 10;
}
```

```
void procedimento_exemplo (float f) {
    f = 10;
}
```

```
int main (int argc, char *argv[]){
    //corpo principal do programa
}
```

## Tipos Primitivos

Tipo	Tamanho	Valores válidos
char	1 byte	-128 a 127
unsigned char	1 byte	0 a 255
short int	2 bytes	-32.768 a 32.767
unsigned short int	2 bytes	0 a 65.535
long int	4 bytes	-2.147.483.648 a 2.147.483.647
unsigned long int	4 bytes	0 a 4.294.967.295
float	4 bytes	$10^{-38}$ a $10^{38}$
double	8 bytes	$10^{-308}$ a $10^{308}$

## Operadores

Tipo	Operador	Propósito	Exemplo
Aritméticos	+	Adição	a = 4 + 1; // 5
	-	Subtração	a = 4 - 1; // 3
	*	Multiplicação	a = 2 * 4; // 8
	/	Divisão	a = 8 / 2; // 4
	%	Módulo (resto da divisão)	a = 5 % 2; // 1
Atribuição	=	Atribuição simples	a = 50;
Lógicos	&&	"e" lógico	(a > 1) && (b < 1)
		"ou" lógico	(a > 1)    (b < 1)
	!	não (inversão)	!(a > 2)
Relacionais (Comparação)	==	igual a	(a == 0)
	!=	diferente de	(a != 0)
	<	menor que	(a < 0)
	>	maior que	(a > 0)
	<=	menor ou igual a	(a <= 0)
	>=	maior ou igual a	(a >= 0)
Incremento e Decremento	++	Incremento	a++;
	--	Decremento	a--;
Referência (Apontadores) Operadores utilizados antes do nome de variáveis	%	Retorna o "endereço de"	int a; //variável inteira int *p; //declaração de ponteiro p = &a; //atribui o endereço de a *p = 2; //atribui ao conteúdo //apontado por p o valor 2; //como p aponta para o endereço //de a, então a recebe 2.
	*	Retorna o "conteúdo de"	

## Entrada e Saída

### printf(formato, argumentos);

Função para saída de valores segundo um determinado formato.

```
Ex.: printf("%d %g\n", 33, 5.3);
     printf("Inteiro = %d Real = %f\n", 33, 5.3);
```

### scanf(formato, lista de endereços)

Função para capturar e armazenar valores fornecidos via teclado.

```
Ex.: scanf("%d",&n);
     Scanf("%d:%d",&h,&m);
```

### Especificadores de formato:

%c	char
%d	int
%u	unsigned int
%f	double ou float
%e	double ou float (científico)
%s	cadeia de caracteres
\n	quebra de linha
\t	tabulação
\"	caractere "
\\	caractere \

Comandos da Linguagem		
Comando	Propósito	Sintaxe
Declaração de variável	Declaração de variável	tipo nome_variavel = valor_inicial;
Declaração de constante	Declaração de constante	#define NOME_CONSTANTE valor
Bloco	Marcar um bloco de cód.	{ } //Abre e fecha chaves "{}"
<b>if</b>	Comando condicional	<pre> <b>if</b> (a &gt; b) {     printf("%s", "A é maior que B"); } <b>else</b> {     printf("%s", "A é igual ou menor que B"); } </pre>
<b>switch</b>	Comando condicional	<pre> <b>switch</b> (i) {     <b>case</b> 0 :         printf("%s", "ZERO");         <b>break</b>;     <b>case</b> 1:         printf("%s", "UM");         <b>break</b>;     <b>case</b> 2:         printf("%s", "DOIS");         <b>break</b>; } </pre>
<b>while</b>	Laço com pré validação	<pre> int i = 1; <b>while</b> (i &lt;= 10) {     printf("%d", i++); } </pre>
<b>do</b>	Laço com pós validação	<pre> int i = 1; <b>do</b> {     printf("%d", i++); } <b>while</b> (i &lt;= 10); </pre>
<b>for</b>	Laço simplificado	<pre> <b>for</b> (i=1;i&lt;=10;i++){     printf("\n%d", i); } </pre>
<b>break</b>	Saída de bloco	<b>break;</b>
<b>continue</b>	Reinício de bloco	<b>continue;</b>
Sub-rotinas	Funções	<pre> float area(float altura, float base) {     return altura * base; } </pre>
	Procedimentos	<pre> void area(float altura, float base) {     printf("A area é: %f", altura * base); } </pre>
Vetores	Variáveis unidimensionais	<pre> int v[10]; //Vetor de inteiros //v[0] é o primeiro elemento e v[9] o último </pre>
Matrizes	Variáveis multidimensionais	<pre> float mat[4][3]; //Tabela com 4 linhas //e 3 colunas </pre>
<b>struct</b>	Tipos de dados compostos	<pre> <b>struct</b> ponto {     int x;     int y; }  <b>struct</b> ponto p; p.x = 10; p.y = 20; </pre>
<b>typedef</b>	Definição de novos tipos de dados	<pre> <b>typedef struct</b> ponto {     int x;     int y; } Ponto;  Ponto p; p.x = 10; </pre>

### Funções utilitárias

sizeof(variavel ou tipo) – retorna o número de bytes de um determinado tipo. Ex. int a = sizeof(float); // 4  
 malloc(tamanho) – recebe o número de bytes que se deseja alocar e retorna o endereço inicial da área alocada.  
 free(ponteiro) – recebe como parâmetro um ponteiro e libera sua memória.