

## A linguagem C

### ■ Introdução à Linguagem C

- ❑ Estrutura de um programa em C
- ❑ Variáveis, constantes
- ❑ Palavras reservadas
- ❑ Identificadores – regras e convenções
- ❑ Tipos de dados pré-definidos

### ■ Expressões

Operadores: atribuição, relacionais, lógicos, manipulação de bits, incremento e decremento, *sizeof*, conversão de tipos.

## Aspectos Introdutórios

```
/* um programa simples em linguagem C */  
#include <stdio.h>  
int main(void)  
{  
    printf("O meu programa em C.\n");  
    printf("Espero que tudo corra bem!");  
    return 0;  
}
```

Após a execução :



O meu programa em C.  
Espero que tudo corra bem!\_

## Características de um programa

- Tem de possuir a função **main()**
- É uma linguagem "case sensitive".  
(main ≠ Main ≠ MAIN)
- Todas as instruções terminam em ;
- Os blocos de instruções ou instruções compostas são definidos através dos caracteres { e }
- Comentários;
  - /\* ....  
          \*/      Comentários de Bloco
  - //           Comentários de Linha

## Identificadores de Variáveis, Constantes e Funções

Os identificadores devem obedecer às seguintes regras:

- devem começar por uma letra ou sublinha ( \_ );
- os caracteres seguintes podem ser letras, sublinhas ou dígitos;
- os identificadores devem ter relação com as variáveis, constantes ou funções, isto é, estas devem ter nomes significativos (a fim de tornar a programação de mais fácil compreensão e reduzir comentários supérfluos).

## Identificadores Regras e Convenções

Quando se escolhe um identificador existem alguns cuidados e convenções que devem ser tidos em atenção:

- **Não** é aconselhável a utilização de caracteres acentuados em identificadores, pois alguns compiladores não os aceitam;
- Identificadores totalmente em **letras maiúsculas** estão geralmente associados a constantes, esta convenção deve ser respeitada;
- Quando se pretende um identificador com **várias palavras**, estas devem ser separadas por *underscore* ou iniciar as palavras por maiúsculas ou minúsculas, por exemplo contaAlunos.

## Algumas Palavras Reservadas

<i>static</i>	<i>default</i>	<i>extern</i>	<i>int</i>	<i>sizeof</i>
<i>union</i>	<i>break</i>	<i>do</i>	<i>float</i>	<i>long</i>
<i>static</i>	<i>unsigned</i>	<i>case</i>	<i>double</i>	<i>for</i>
<i>register</i>	<i>struct</i>	<i>while</i>	<i>char</i>	<i>else</i>
<i>goto</i>	<i>return</i>	<i>switch</i>	<i>void</i>	<i>continue</i>
<i>entry</i>	<i>if</i>	<i>short</i>	<i>typedef</i>	<i>static</i>

## Variáveis

### Declaração de variáveis

Tipo\_da\_Variável      Lista\_De\_Variáveis ;

Uma variável está associada a uma posição de memória onde são armazenados dados. O seu conteúdo pode variar ao longo da execução do programa.

## Variáveis

### Exemplos:

```
int a, teste, conta1;
```

```
int conta2 = 23, x;
```

```
float tenta, Tenta;    //Correcto mas não aconselhável
```

```
char car;
```

```
unsigned int sem_sinal;
```

```
long double longo;
```

## Constantes

Constantes são identificadores que correspondem a valores **fixos**, não podem ser modificados ao longo da execução do programa.

```
const tipo_constante identificador_constante = valor;
```

Tipo de Dado	Exemplos de Constantes
char	'b' '\n' '\0'
int	2 32000 -130
long int	100000 -467
short int	100 -30
unsigned int	50000 35678
float	0.0 23.7 -12.3e-10
double	12546354334.0 -0.0000034236556

## Constantes

Utilizando a directiva de pré-processamento **#define**

```
#define identif_const valor
```

**Exemplo:**

```
#include <stdio.h>

#define MAX 10

int main (void)
{
    printf("%d\n",MAX);

    return 0;
}
```

## Tipos de Dados Pré-definidos

O **tipo** de uma variável/constante define os valores que ela pode assumir e as operações que podem ser realizadas com ela.

Os **tipos pré-definidos** são:

<b>char:</b>	um byte que armazena o código de um carácter
<b>int:</b>	inteiros com ou sem sinal 32 bits (*)
<b>float,</b>	
<b>double:</b>	números reais (em vírgula flutuante)

*\* Nem sempre o tamanho do int é 4 bytes.*

## Modificadores de Tipo

- Os modificadores de tipo permitem alterar a gama de variação do tipo base, possibilitando adaptações a situações específicas.
- **Exemplos de Modificadores:**
  - **signed:** números com sinal, por omissão (**int** ; **char**).
  - **unsigned:** números positivos (**int** ; **char**).
  - **long:** aumenta a gama de variação (**int** ; **double**).
  - **short:** reduz a gama de variação ( **int** ).

## O Tipo Inteiro

Tipo	Tamanho (bits)	Gama de variação
int	32	-2 147 483 648 ... 2 147 483 647
unsigned int	32	0 ... 4 294 967 295
short int	16	-32768 ... 32767
unsigned short int	16	0 ... 65535
long	32	-2 147 483 648 ... 2 147 483 647
unsigned long	32	0 ... 4 294 967 295

V Vasconcelos

13

## O Tipo Inteiro

### Exemplos:

```
const unsigned int MAIOR = 500010;  
const int MUITO_NEG = -80000;  
const short int PEQUENO=1024;  
unsigned int positivo;  
double grande, maior;
```

V Vasconcelos

14

## Operadores e Funções Disponíveis para Inteiros

Operador	C
Adição	+
Subtração	-
Multiplicação	*
Divisão inteira	/
Resto da Divisão inteira	%
Funções	C
Valor Absoluto  x	abs(x)

## O Tipo Real

Define números reais no formato de vírgula flutuante

Tipo	Tamanho (bits)	Gama de variação
float	32	$-3.4 \times 10^{38}$ .. $-3.4 \times 10^{-38}$ , 0, $3.4 \times 10^{-38}$ .. $3.4 \times 10^{38}$
double	64	$-1.7 \times 10^{308}$ .. $-1.7 \times 10^{-308}$ , 0, $1.7 \times 10^{-308}$ .. $1.7 \times 10^{308}$



O Tipo Real

Exemplos

const float PI = 3.1416;  
const double PI\_PRECISAO = 3.141592654;  
const double VELOCIDADE\_LUZ = 2.99792458E8;  
float media;  
float raio, volume;

V Vasconcelos17

O Tipo Real - Notações

Decimal - parte inteira e fraccionária separadas por um ponto.

Exemplos:0.035; -2532.3; -0.4

Científica - possui a seguinte definição formal:  
<mantissa>E<expoente> ou  
<mantissa>e<expoente>

Exemplos>>>>>>>>

Formato Científico	C
524.4x10 <sup>3</sup>	524.4e3
-12 x 10 <sup>6</sup>	-12E6
33.01 x 10 <sup>-2</sup>	33.01E-2

V Vasconcelos18

## Funções Disponíveis para Reais

#include <math.h>

Funções	C
$ x $	fabs (x)
$\sqrt{x}$	sqrt (x)
$e^x$	exp(x)
$\ln (x)$	log(x)
$\log_{10}(x)$	log10(x)
$\sin(x)$	sin(x)
$\cos(x)$	cos(x)
$\text{tg}(x)$	tan x
$\arctg(x)$	atan(x)
$\arctg(y/x)$	atan2(y,x)
$x^y$	pow(x,y)

## Operadores Disponíveis para Reais

Operador	C
Adição	+
Subtracção	-
Multiplicação	*
Divisão Real	/

## O Tipo Carácter

### *Char*

- O código ASCII permite ao computador armazenar texto (conjunto de caracteres). Este código *ASCII* estabelece uma correspondência unívoca entre caracteres e números (código alfanumérico).
- **Código *ASCII Standard*** usa 7 bits o que permite representar 128 caracteres distintos.
- **Código *ASCII Estendido*** usa 8 bits o que permite representar 256 caracteres distintos. Os primeiros 128 caracteres são os mesmos do código *ASCII standard*; os restantes caracteres são especiais ( 'ç', vogais acentuadas, '≥', 'L', '↑', '©', ...)

## Tipo *char*.

### Exemplos :

```
const char PRIMEIRA = 'a';  
const char ULTIMA = 'Z';  
const char ENTER = '\n';  
const char ESC = '\x1B';  
char letra, inicial;
```

Tabela ASCII

Posição	0	1	2	3	4	5	6	7	8	9
0	#####									
10										
20										
30				!	"	#	\$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	.
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{		}	~		€	
130	,	f		...	†	‡	^	‰	Š	‹
140	œ	□	Ž	□	□	ˆ	˚	°	˚	•
150	—	—	~	™	š	›	œ	□	ž	Ÿ
160	ˆ	ˆ	€	£	¤	¥	¦	§	¨	©
170	ª	«	¬	-	®	¯	°	±	²	³
180	´	µ	¶	·	¸	¹	º	»	¼	½
190	¾	¿	À	Á	Â	Ã	Ä	Å	Æ	Ç
200	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ
210	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û
220	Ü	Ý	Þ	ß	à	á	â	ã	ä	å
230	æ	ç	è	é	ê	ë	ì	í	î	ï
240	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù
250	ú	û	ü	ý	þ	ÿ				

V Vasconcelos

23

RESUMO

Tipo	Nº de bits	Leitura com scanf	Gama de Variação	
			Início	Fim
char	8	%c	-128	127
unsigned char	8	%c	0	255
signed char	8	%c	-128	127
int	32	%d	-2.147.483.648	2.147.483.647
unsigned int	32	%u	0	4.294.967.295
signed int	32	%d	-2.147.483.648	2.147.483.647
short int	16	%hd	-32.768	32.767
unsigned short int	16	%hu	0	65.535
signed short int	16	%hd	-32.768	32.767
long int	32	%ld	-2.147.483.648	2.147.483.647
signed long int	32	%ld	-2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295
float	32	%f	3,4E-38	3,4E+38
double	64	%lf	1,7E-308	1,7E+308

V Vasconcelos

24

## Operador de Atribuição

- O operador de atribuição é o carácter **=** e lê-se **"toma o valor de"**.
- O operador **=** (de atribuição) permite que à variável que surge à sua esquerda seja atribuído o resultado da expressão que se encontra à direita.
- Este é o operador com menor prioridade a seguir ao operador vírgula, dos operadores da linguagem C.

## Operador de Atribuição

### Exemplos

```

...
const int SOMA = 11;
int Z;
Z = 5;                //z toma o valor de 5
Z = SOMA * 2;         //z toma o valor de 22
Z = 35 * 2 - 20;      //z toma o valor de 50
Z = Z + 5;            //z toma o valor de 55
Z = sqrt(4) + 1;      //z toma o valor de 3
Z = pow(Z,2) - 10;    //z toma o valor de -1
...

```

Operações de Atribuição

Formato Abreviado

Formato Normal	Formato Abreviado
<code>x = x + y</code>	<code>x += y</code>
<code>x = x - y</code>	<code>x -= y</code>
<code>x = x * y</code>	<code>x *= y</code>
<code>x = x / y</code>	<code>x /= y</code>
<code>x = x % y</code>	<code>x %= y</code>
<code>x = x   y</code>	<code>x  = y</code>
<code>x = x ^ y</code>	<code>x ^= y</code>
<code>x = x &gt;&gt; y</code>	<code>x &gt;&gt;= y</code>
<code>x = x &lt;&lt; y</code>	<code>x &lt;&lt;= y</code>

V Vasconcelos27

Operadores Relacionais

Permitem comparar expressões sendo o resultado **0** ou **diferente de 0**, consoante o resultado da comparação for, respectivamente, **falso** ou **verdadeiro**.

Operador	C
maior que	<code>&gt;</code>
maior ou igual a	<code>&gt;=</code>
menor que	<code>&lt;</code>
menor ou igual a	<code>&lt;=</code>
igual a	<code>==</code>
diferente de	<code>!=</code>

V Vasconcelos28

# Operadores Relacionais

## Exemplos

Expressão	Resultado
'c' > 'z'	0
123.1 > 10.011	1
101E10 <= 101	0
'a' != 'A'	1
2==2	1

# Operadores Lógicos

É possível construir expressões booleanas utilizando os operadores lógicos. O valor **Falso** é representado pelo inteiro **0** e **Verdadeiro** por um valor inteiro **diferente de zero**.

Operador	C
E	&&
OU	
NEGAÇÃO	!

Operadores Lógicos

Tabela de Verdade

a	b	a&b (conjunção)	a  b (disjunção)	!a (negação)
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

V Vasconcelos31

Operadores de Manipulação de Bits

Realizam operações lógicas bit a bit

Operador	C
E bitwise	&
OU bitwise	
EXOR bitwise	^
NEGAÇÃO bitwise	~
passagem à esquerda	<<
passagem à direita	>>

V Vasconcelos32



## Operadores de Manipulação de Bits

### Exemplos

Variável	Resultado
<b>A</b>	<b>1 0 1 1 0 1 0 0</b>
<b>B</b>	<b>1 1 1 0 0 1 0 1</b>
<b>A &amp; B</b>	<b>1 0 1 0 0 1 0 0</b>
<b>A   B</b>	<b>1 1 1 1 0 1 0 1</b>
<b>A ^ B</b>	<b>0 1 0 1 0 0 0 1</b>
<b>~A</b>	<b>0 1 0 0 1 0 1 1</b>
<b>A &lt;&lt; 3</b>	<b>1 0 1 0 0 0 0 0</b>
<b>B &gt;&gt; 3</b>	<b>0 0 0 1 1 1 0 0</b>

V Vasconcelos

33

## Operadores de Incrementação e Decrementação

Os operadores de **incrementação** (**++**) e **decrementação** (**--**), permitem incrementar/decrementar uma variável de uma unidade.

### Sintaxe:

<b>ident_var++</b>	<b>pós-incrementação</b>
<b>++ident_var</b>	<b>pré-incrementação</b>

<b>ident_var --</b>	<b>pós-decrementação</b>
<b>--ident_var</b>	<b>pré-decrementação</b>

V Vasconcelos

34

## Operadores de Incrementação e Decrementação

Se o operador é colocado depois da variável, esta só vê o seu valor alterado depois de ter sido utilizado na expressão.

**soma = x++;**

**Equivale a :**      **soma = x;**  
                         **x++;**

Se o operador é colocado antes da variável então é realizada em primeiro lugar a incrementação/decrementação e só depois é que o valor corrente da variável é utilizado na expressão.

**soma = ++x;**

**Equivale a :**      **x++;**  
                         **soma = x;**

## Pós e Pré Incrementação e Decrementação Exemplo

**int x = 20, y;**  
**y = x++;**

*/\* y é agora 20 e x é 21 \*/*

**y = ++x;**

*/\* y é agora 22 e x é 22 \*/*

**x = 1;**  
**y = 10 \* x++;**

*/\* y é agora 10 e x é 2 \*/*

**x = 1;**  
**y = 10 \* ++x;**

*/\* y é agora 20 e x é 2 \*/*

## Operador sizeof

O operador **sizeof** permite obter o número de bytes que ocupa, um tipo de dados ou uma variável.

### Exemplo:

```

...
double k;
int tam1, tam2, tam3;
char letra;
tam1 = sizeof(int);    /* tam1 toma o valor 4 */
tam2 = sizeof(k);      /* tam2 toma o valor 8 */
tam3 = sizeof(letra);  /* tam3 toma o valor 1 */
...

```

V Vasconcelos

37

## Operador de Conversão de Tipos (type\_cast)

- O C aceita o seguinte modo de conversão de tipos:  
(type\_cast)expressão

### Exemplo:

```

...
int x = 2;
float y, z;
y = (float)x; // o valor de x é convertido num float e a seguir atribuído a y
...

```

V Vasconcelos

38

## Expressões

- Uma expressão é um conjunto de operandos (valores, constantes, valor corrente de variáveis e resultados de funções) agrupados por operadores. As regras usadas na avaliação da expressão são:
  - ❑ **Operadores com a mesma prioridade**, a expressão é avaliada da esquerda para a direita;
  - ❑ **Operadores com prioridades distintas**, são efectuados em primeiro lugar as operações que contêm os operadores de maior prioridade;
  - ❑ A ordem de avaliação da expressão pode ser **alterada** com a utilização de **parênteses**.

## Prioridades Relativas dos Operadores

Operador	Precedência
() [] -> .	MAIOR
! ~ ++ -- - (type_cast) * & sizeof	
* / %	
+ -	
<< >>	
< <= > >=	
== !=	
&	
^	
&&	
? : (operador ternário)	
= += -= *= /= etc...	
,(vírgula)	MENOR

## Exercício

```
int x, y, z;
```

```
x = 4;
```

```
y = 25;
```

```
z = 25%4;
```

```
x /= 2;
```

```
--y ;
```

```
y = 3 + z;
```

Quais os valores de

**x = ??**

**y = ??**

**z = ??**

**Solução:**

**x = 2**

**y = 4**

**z = 1**

V Vasconcelos

41

## Exercício

```
float x, y, z;
```

```
x = 10.0;
```

```
y = 3.0;
```

```
z = pow(x,2.0) + x++;
```

```
x += y + y + z;
```

```
y--;
```

Quais os valores de

**x = ??**

**y = ??**

**z = ??**

**Solução:**

**x = 127.0**

**y = 2.0**

**z = 110.0**

V Vasconcelos

42

## Exercício

```
float x, y, z;
```

```
x = 22.0;
```

```
y = 5.0;
```

```
z = 25.0;
```

```
x /= sqrt(z) * y ;      //Operador /= tem menor prioridade que o *
```

```
y -= 23 - x + z++;
```

Quais os valores de

**x = ??**

**y = ??**

**z = ??**

**Solução:**

**x = 0.88**

**y = -42.12**

**z = 26**

*"Conte-me e eu esqueço.*

*Mostre-me eu lembro-me.*

*Deixe-me fazer e eu aprendo."*

(Confúcio)