



Linguagem de Programação C

Breve histórico de “C”

- Criada por Dennis Ritchie;
- Em 1972;
- Centro de Pesquisas da Bell Laboratories;
- Para utilização no S.O. UNIX.



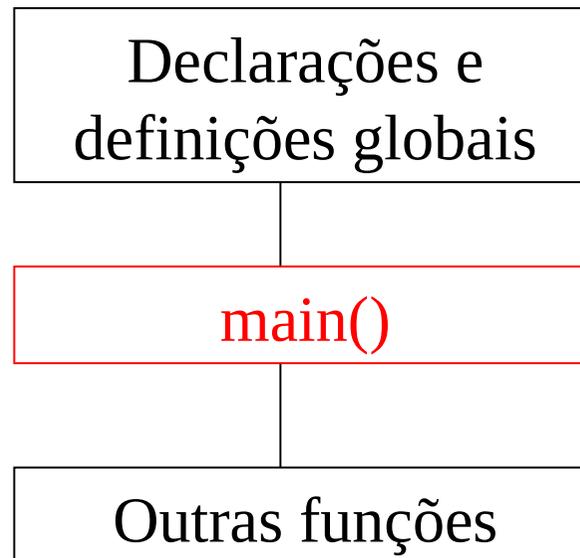
Características Básicas da Linguagem

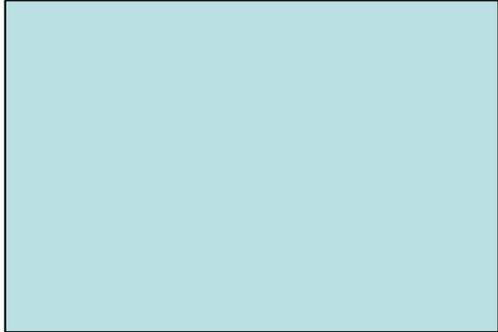
- O C é uma linguagem de propósito geral;
- Case sensitive (sensível ao caso);
- Tipos de dados primitivos: caractere, inteiro e real;
- Possui estruturas de controle de fluxo para viabilizar a programação estruturada;
- Operadores aritméticos, lógicos, relacionais, condicionais e bit a bit;
- Funções de entrada e saída formatadas;
- Todo programa tem uma função chamada *main()*;
- Praticamente, toda linha do programa termina com ";".

ANSI

Em 1983, o **Instituto Norte-Americano de Padrões (ANSI)** formou um comitê, X3j11, para estabelecer uma especificação do padrão da linguagem C. O padrão foi completo em 1989 e ratificado como ANSI X3.159-1989 “Programming Language C” (**C ANSI**).

Estrutura de um programa em C





Conceitos Básicos – Linguagem C

Constantes

Exemplos:

- Decimal (10, -23768)
- Hexadecimal (0x12,0x1fea28)
- Octal (0123)
- Real (2.34, 2.34E+5, 2.14E-9)
- Caractere ('a', '%')

Identificadores

Identificadores são os nomes utilizados para referenciar variáveis, funções ou vários outros objetos definidos pelo programador. Sendo constituídos por:

- letras, dígitos e sublinhado(_);
- não podem começar com dígito;
- não podem ser iguais a uma palavra reservada e nem iguais a um nome de uma função disponibilizada pelo método utilizado para construção do programa.

Palavras Reservadas e Comentários

➤ Palavras Reservadas

asm	const	else	for	sizeof	union
auto	continue	enum	goto	static	unsigned
break	default	extern	if	struct	void
case	do	far	int	switch	volatile
char	double	float	long	typedef	while

➤ Comentários

`/* este trecho não sera considerado pelo processador da linguagem */`

Tipos Primitivos

➤ Caractere

- Definido por char;
- Ocupa 8 bits (1 byte)
- Faixa de valores: -128 à 127

➤ Declaração de variáveis

- Exemplo de declaração de uma variável do tipo primitivo caractere:

char letra;

letra ⊕ 'A';

Operador de atribuição

Tipos Primitivos

➤ Inteiro

- Definido por int;
- Ocupa 16 bits (2 bytes)
- Faixa de valores: -32768 à 32767
- Exemplo:

```
int num = 13;
```

```
num = num - 73;
```

Tipos Primitivos

▶ Ponto flutuante e ponto flutuante de precisão dupla

- ▶ float → 4 bytes

 - ▶ Seis dígitos de precisão

- ▶ double → 8 bytes

 - ▶ Dez dígitos de precisão

- ▶ faixa mínima de um valor em ponto flutuante

 - ▶ $1E-37$ a $1E+37$

- ▶ Exemplo: float a,b,c=2.34;

 - double x=2.38,y=3.1415,z;

Operadores

Operadores Aritméticos

Unários: +, -, ++, --

Exemplos: +1

-5

a=-b;

a++; \Leftrightarrow a=a+1;

a--; \Leftrightarrow a=a-1;

b=a++; \Leftrightarrow b=a;

a=a+1;

b=++a; \Leftrightarrow a=a+1;

b=a;

Obs.: a e b são variáveis numéricas.

Operadores

Operadores Aritméticos

Binários: +, -, *, /, %

Não existe o operador \ (trabalhado em Introdução a Algoritmos)

10/3=? 3
10/3.0=? 3.3333
10.0/3=? 3.3333
10.0/3.0=? 3.3333

Precedência (Hierarquia nas operações)

Hierarquia	Operação
1	Parênteses
2	Função
3	++,--
4	- (menos unário)
5	*, /, %
6	+, -

Operadores

► Operadores de Atribuição

=, +=, -=, *=, /=, %=

Exemplos:

a=5;

a+=5; \Leftrightarrow a=a+5;

a*=5-2; \Leftrightarrow a=a*(5-2);

Operadores

Operadores Lógicos

Operador	Ação
&&	e
	ou
!	não

&&	V	F
V	V	F
F	F	F

	V	F
V	V	V
F	V	F

!V == F

Operadores

Operadores Relacionais

Operador	Ação
>	maior que
>=	maior ou igual a
<	menor que
<=	menor ou igual
==	igual a
!=	diferente de

Operadores

Precedência (Hierarquia dos operadores relacionais e lógicos)

Hierarquia	Operação
1	!
2	>, >=, <, <=
3	==, !=
4	&&
5	

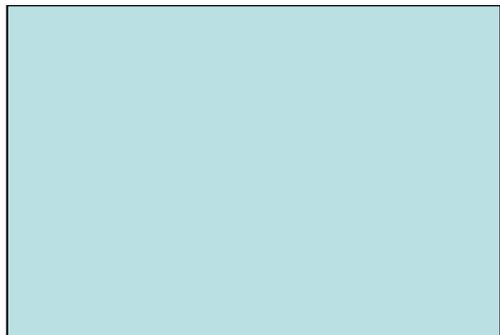
Operadores Lógicos bit a bit

Obs.: Aplicados a char e int.

Operador	Ação
&	AND(e)
	OR(ou)
^	XOR(ou exclusivo)
~	NOT(não)
>>	Desloca os bits à direita
<<	Desloca os bits à esquerda

Exemplos:

	11000001	193 em binário
	01111111	127 em binário
&	<u> </u>	AND bit a bit
	01000001	65 em binário
	10000000	128 em binário
	00000011	3 em binário
	<u> </u>	OR bit a bit
	10000011	131 em binário
	01111110	126 em binário
	01111001	121 em binário
^	<u> </u>	XOR bit a bit
	00000111	7 em binário



$$\begin{array}{cccccccc}
 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 = \\
 2^7 * 1 + 2^6 * 1 + 2^5 * 0 + 2^4 * 0 + 2^3 * 0 + 2^2 * 0 + 2^1 * 0 + 2^0 * 1 = \\
 128 + 64 + 0 + 0 + 0 + 0 + 0 + 1 = 193
 \end{array}$$

5 && 2 == 5 & 2
?

·00101100 byte original (44 em binário)
 ·11010011 após o 1º complemento
 00101100 após o 2º complemento

Operadores Lógicos bit a bit

Exemplos:

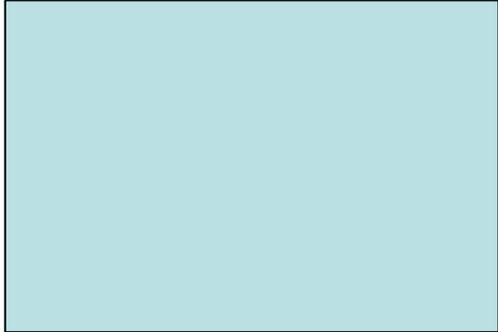
➤ Deslocamento à esquerda:

`variavel2 = variavel1 << num_de_deslocamentos`

➤ Deslocamento à direita

`variavel2 = variavel1 >> num_de_deslocamentos`

unsigned char x;	x na base binária a cada execução da sentença	Valor de x na base decimal a cada execução da sentença
<code>x=7;</code>	00000111	7
<code>x=x<<1;</code>	00001110	14
<code>x=x<<3;</code>	01110000	112
<code>x=x<<2;</code>	11000000	192
<code>x=x>>1;</code>	01100000	96
<code>x=x>>2;</code>	00011000	24



Primeiro Programa na Linguagem C
Funções de Entrada e Saída Formatada

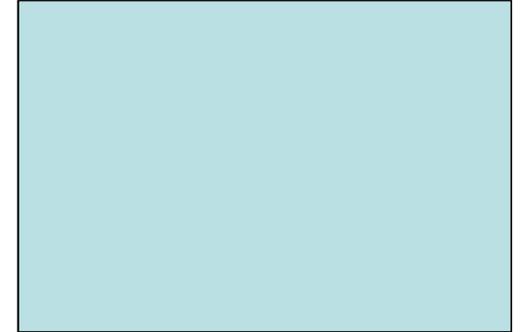
Primeiro programa na linguagem C



➔ Estrutura mínima

```
main()  
{  
  ...  
}
```

Primeiro programa na linguagem C



➔ Estrutura mínima

```
main()
{
    int x = -1, y;
    y = x + 7;
}
```

Funções de Entrada e Saída Formatada

Conceito de função:

Ação, ato ou efeito.

Exemplo:

Dentre as funções dos vigias da UNIVASF, está a *função* de abrir a porta das salas de aula.

Contudo, para que um dos vigias abra a porta de uma das salas de aula temos que solicitar ao mesmo e informar, durante a solicitação, a data, o horário e o número da sala a ser aberta.

Computacionalmente, em uma análise inicial, uma função representa uma sequência de instruções que será executada para efetuar uma certa tarefa. Denominamos como *parâmetros* da função os dados necessários para que esta execute.

Funções de Entrada e Saída Formatada

Diretiva do pré-processador

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

- ✦ std → standard (padrão)
- ✦ io → input/output (entrada/saída)

printf ()

- ✦ Forma geral:

```
printf (string_de_controle, lista_de_argumentos);
```

Funções de Entrada e Saída Formatada

▶ printf (continuação)

▶ *string_de_controle*

- ▶ descrição de tudo que a função apresentará na tela;
- ▶ indica os caracteres;
- ▶ indica as variáveis e expressões, cujos valores serão apresentados, e em quais posições específicas aparecerão. Isso é feito, usando os códigos de controle, com a notação %.

Funções de Entrada e Saída Formatada

Tabela de códigos de formato (%)

Código	Formato
%c	Um caractere (char)
%d	Um número inteiro decimal (int)
%i	O mesmo que %d
%e	Número em notação científica com o "e" minúsculo
%E	Número em notação científica com o "E" maiúsculo
%f	Ponto flutuante decimal
%g	Escolhe automaticamente o melhor/menor entre %f e %e
%G	Escolhe automaticamente o melhor/menor entre %f e %E
%o	Número octal
%s	String
%u	Decimal "unsigned" (sem sinal)
%x	Hexadecimal com letras minúsculas
%X	Hexadecimal com letras maiúsculas
%%	Imprime um %

Funções de Entrada e Saída Formatada

➤ printf (continuação)

➤ *lista_de_argumentos*

Para cada código % contido na string de controle, temos um argumento correspondente na *lista_de_argumentos*

Funções de Entrada e Saída Formatada

➤ printf (continuação)

Vamos ver alguns exemplos:

Código	Imprime
<code>printf ("Um %%%c indica %s",'c',"char");</code>	Um %c indica char
<code>printf ("%X %f %e",107,49.67,49.67);</code>	6B 49.670000 4.967000e+001
<code>printf ("%d %o",10,10);</code>	10 12

Funções de Entrada e Saída Formatada

➔ printf (continuação)

/* Estrutura geral de um programa em C que utiliza a função de saída formatada */

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

```
main ()
```

```
{
```

```
    ...
```

```
    printf("...",...);
```

```
    ...
```

```
}
```

Funções de Entrada e Saída Formatada

➔ printf (continuação)

/* Estrutura geral de um programa em C que utiliza a função de saída formatada */

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

```
main ()
```

```
{
```

```
    char c;
```

```
    c = 'A';
```

```
    printf("%c",c);
```

```
}
```

Funções de Entrada e Saída Formatada

Exercício

Construa um programa, na linguagem de programação C, que escreva a string “juros de ”, o inteiro 10 e o caractere ‘%’ na tela, constituindo a seguinte frase:
juros de 10%