Programação I

</ Linguagem C />

- Sintaxe;
- Definições;
- Declarações
- Funções

TM

2021

TekMind

Linguagem C

⟶ Introdução

A linguagem de programação C foi criada em 1972 por *Dennis Ritchie* no centro de pesquisa da *Bell Laboratories* que tem de propósito geral, uma adequadação à programação estruturada.

⟶ Sintaxe

Regras detalhadas para cada construção na linguagem C. Essas regras estão relacionadas com os **tipos**, as **declarações**, as **funções** e as **expressões**.

Tipos: Definem as propriedades dos dados manipulados em um programa;

Declarações: Expressam as partes do programa, onde é alocado memória, definido o conteúdo inicial e suas funções.

Funções: Especificam as ações que um programa executa quando roda;

Expressões: determinação e alteração de valores regularmente chamadas de funções I/O.

- Todo programa em C inicia sua **execução** chamando a função ***main*()**.
- Os **comentários** no programa são colocados entre **/*** e ***/** (não sendo considerados na compilação).
- Cada instrução encerra com **;** (ponto e vírgula).

A estrutura de um programa em C possui os seguintes elementos:

- definições de pré-processamento
- definições de tipo
- declarações de variáveis globais
- protótipos de funções
- funções

Definição de pré-processamento

Comandos interpretados pelo compilador, que diz respeito a **operações realizadas** pelo compilador para geração de código. Geralmente inicial com uma cerquilha (#) e não são comandos da linguagem C. Entre os mais utilizados, temos:

⟶ #include <stdio.h>

Utilizado para indicar ao compilador que ele deve "colar" as definições do stdio.h no arquivo antes de compilá-lo. Importante para descrever as funções básicas de entrada e saída.

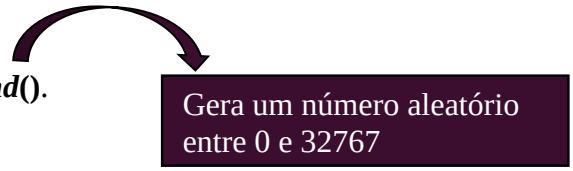
⟶ #include <conio.h>

Quando a função **getch**() é utilizada, em que espera o **pressionamento** de uma tecla pelo usuário. E também, a função ***clrscr***(), que é utilizada para **limpar** a tela.

OBS: a função **getche**() funciona de forma semelhante, porém **exibe** na tela o caracter digitado.

⟶ #include <stdlib.h>

Requerida para usar a ***rand*()**.



⟶ #include <math.h>

Utilizada para realizar **cálculos matemáticos**, como: função trigonométrica, logaritmica, exponencial, entre outras.

⟶ Identificadores

Nomes usados para fazer **referência** aos objetos definidos pelo usuário (Ex.: variáveis, funções, rótulos, etc.).

É um caso *sensitive*, em que as letras maiúsculas se **diferem** das minúsculas.

Definições de Tipo

Quando é declarado um **identificador**, dá a ele um **tipo**. Os tipos principais podem ser colocados dentro da classe do tipo objeto dado. Um tipo de objeto de dados determina como esses valores irão ser **expressados**, **operados** e **representados**.

No C existem cinco tipos de dados básicos:

Tipo	Valor	Número de *Bytes*	Faixa de Dados	Exemplos
Char	*Caracter*	1	-128 a 127	‘a’ ‘s’
Int	Inteiro	2	-32768 e 32768	12 -1900
Float	ponto flutuante (números racionais);	4	6 dígitos de precisão	-0,5 9,56
Double	ponto flutuante de dupla precisão	8	15 dígitos de precisão	44,4565464564
void	Nulo	0	Sem valor	

⟶ Operador de Atribuição

É representado pelo sinal de igual (=), utilizado para **atribuir** um valor a outra variável. A declaração de atribuição é:

variável = **expressão**;

⟶ Operadores Matemáticos

Em C temos dois operadores matemáticos **unários** (agem sobre um único operando) e cinco **binários** (agem sobre dois operandos).

→ Operadores Unários

Operador	Símbolo	Ação	Exemplos
Incremento	++	Incrementa o operando de uma unidade	x++, ++x
Decremento	--	Decrementa o operando em uma unidade	x--, --x

Sendo que:

x++; é equivalente à x = x + 1;
y++; é equivalente à y = y - 1;

Quando o operador for colocado antes da variável, ele **modifica** o conteúdo da variável **antes** dela ser usada.

→ Operadores Binários

A tabela abaixo apresenta os operadores binários de C e as ações executadas pelos operandos x e y:

Operador	Símbolo	Ação	Exemplos
Adição	+	Adiciona dois operandos	x + y
Subtração	-	Subtrai o segundo operando do primeiro	x - y
Multiplicação	*	Multiplica dois operandos	x * y
Divisão	/	Divide o primeiro operando do segundo	x / y
Módulo de Divisão	%	Calcula o resto de uma divisão de inteiros	x % y

→ Operadores Relacionais

Usados para comparar dois valores. São eles:

Operadores	Exemplos
==	igual
!=	Diferente
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a

O valor de uma expressão envolvendo um operador relacional pode ser **Verdadeiro** ou **Falso**.

Por exemplo, se x=5 e y=-4. O valor da expressão x==y é **falso**. Mas se y fosse igual a 5, então resultaria em **verdadeiro**.

→ Operadores Lógicos

Imagine duas expressões quaisquer: x e y. Supondo que elas resultam, falso (F) ou verdadeiro (V), a tabela abaixo mostra os resultados que podem ser obtidos usando dos operadores lógicos:

Operadores	Símbolos
And (e)	&&
Or (ou)	||
Not (negação)	!

Declarações de variáveis globais

As linguagens de programação exigem que o programador defina o **espaço** e o **local da memória** a ser usado por um determinado valor.

Quantidade de espaço: definida quando informamos qual é o tipo de valor a ser armazenado. Por exemplo, quando indicamos um valor do tipo char, um byte é colocado na memória RAM para seu armazenamento.

Local da memória: Locais na memória do computador que podem ser acessados através de **endereços**, portanto, deve nomear em relação ao endereço da memória onde será armazenado determinado valor.

Esse processo de definir um local e uma quantidade de espaço na memória do computador para armazenar os dados, é chamada de **declaração de variáveis**.

→ Algumas regras para criar nome das variáveis

- Pode conter letras, números e caracteres sublinhados;
- O primeiro caractere não pode ser número;
- Letras minúsculas são consideradas diferentes das maiúsculas (Ex.: delta e Delta, são diferentes);
- Os caracteres não podem ter sinais de pontuação (Ex: variável1, cão, etc.);

→ Declaração e Atribuição

Para declarar uma variável em seu algoritmo, escreva o **tipo** de dado que será armazenado na variável, seguido de seu **nome**. Sintaxe:

```
tipo nome_variável;
```

Atribuindo valores às variáveis. Sintaxe:

```
nome_variável = valor;
```

→ Compatibilidade de Tipos

O valor a ser armazenado em uma variável deve ser do **mesmo tipo** da variável.

Exemplo: int a;

a = 5; //correto
a = 5,3; //cuidado: a parte decimal será perdida, pois não há espaço suficiente para armazená-la.

Funções

→ main()

Função principal e obrigatória para **execução** de um programa em C, que contém o código de execução após rodar o programa. Se não for adicionado no algoritmo, será gerado um erro durante o processo de geração do programa.

Estrutura de todo programa C

```
#include<stdio.h>
  main(){
     //corpo do programa
}
```

→ printf()

Função de que permite **escrever caracteres** que serão exibidos na tela juntamente com os códigos de formatação, que indicam o **formato** em que os argumentos devem ser impressos. Cada **argumento** deve ser separado por **vírgula**. Sintaxe:

```
printf("expressão de controle", argumentos);
```

Exemplo: int n = 15;
printf("O valor de n eh: %d", n); //exibe o "O valor de n eh 15".

Note-se que todo formato da string de dados é exibido com o especificador **"%d**", que é substituído pelo valor em formato **inteiro** da variável.

Especificadores de Formato mais utilizados:

%c	Caracteres simples (*char*)
%d	Inteiro (*int*)
%e	Notação científica
%f	Ponto flutuante (float)
%g	%e ou %f (mais curto)
%o	Octal
%s	*String*
%u	Inteiro sem sinal
%x	Hexadecimal
%lf	Tipo *double*
%u	Inteiro não sinalizado (*unsigned int*)
%old	Tipo *long int*

→ scanf()

É o complemento do printf(), que nos permite **ler dados** formatados do teclado. Sintaxe:

scanf (“string de formato”, &var1, &var2,...&varN);

– Operador de endereço “&”

Ele tem a função de **retornar** o endereço do operando.

Exemplo 1:

```
int t;
printf(“Digite um inteiro: “);
scanf(“%d”, &t);
```

/*aguarda a digitação de um número do tipo int. O número digitado é armazenado na variável t quando o usuário digita ENTER */

Exemplo 2:

```
  char carac1;
  int i;
  printf(“Digite um caracter e um int, separados por vírgula: ”);
  scanf(“%c, %d”, &carac1, &i);
```

/*Os especificadores de formato %c e %d estão separados por vírgula, o que significa que o usuário deve digitar os valores também separados por vírgula e na ordem correta*/

→ getchar()

Função original de entrada de caractere dos sistemas baseados em UNIX. Ele armazena a entrada até que ENTER seja pressionado. **Exemplo**:

```
main(){
   char ch;
   ch = getchar();
   printf("%c",ch);
}
```