

Engenharia da Computação

Disciplina: DCExt Programação Imperativa

Aula: Introdução à Linguagem de Programação C

Prof. Dr. Hemir da C. Santiago
hcs2@poli.br

Aula 02.1:



- Linguagem de Programação C
- Dados
- Tipos de dados
- Variáveis
- Constantes
- Entrada e saída de dados
- Exercícios

- Quando? Em 1972
- Onde? Bell Telephone Laboratories
- Quem? Dennis Ritchie
- Por que? Para (re)escrever o UNIX



UNIX foi criado por
Kenneth Thompson e
Dennis Ritchie.

https://pt.wikipedia.org/wiki/Ken_Thompson

https://pt.wikipedia.org/wiki/Dennis_Ritchie

[https://pt.wikipedia.org/wiki/C_\(linguagem_de_programa%C3%A7%C3%A3o\)](https://pt.wikipedia.org/wiki/C_(linguagem_de_programa%C3%A7%C3%A3o))

Linguagem de Programação C

- É uma linguagem de programação de nível médio;
- Uma **IDE** (*Integrated Development Environment*) consiste de um conjunto de ferramentas que oferecem suporte às atividades de desenvolvimento de programas.
- O **compilador** é um dos programas da IDE encarregado de ler o programa ou código fonte, escrito na linguagem C, e de convertê-lo em programa ou código executável.

Linguagem de Programação C

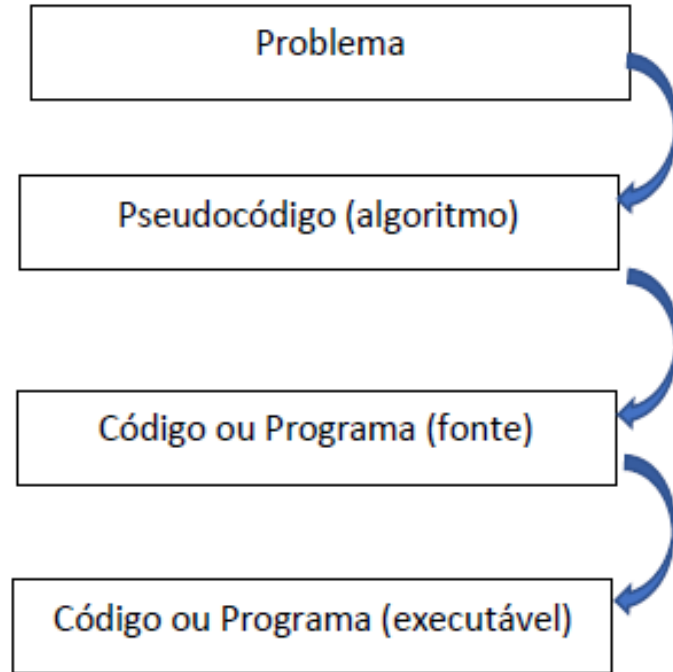


Figura 1. Etapas para desenvolvimento de programas

Linguagem de Programação C

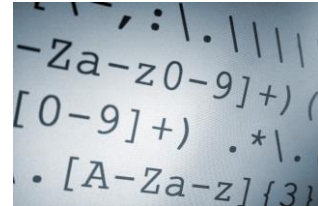
1. É uma **unidade básica de informação**, como: nome, sexo, data de nascimento, grau de instrução, idade.

Exemplos: Carlos, masculino, 30/01/2018, graduação, 50

2. É a representação **numérica**, **alfanumérica**, **gráfica** ou **sonora** de uma determinada realidade.

3. Pode ser manipulado (processado) e gerar novo dado.

Exemplo: $400/8 = 50$



Dados

- Dados numéricos

- São divididos em apenas duas classes: INTEIROS e REAIS.

- ❖ Dados numéricos **inteiros**:

São aqueles que **não** possuem componentes decimais ou fracionários, podendo ser positivos ou negativos (conj. dos \mathbb{N} e \mathbb{Z}). Exemplos:

24: número inteiro positivo	<i>int numeroA = 24;</i>
0: número inteiro	<i>int numeroB = 0;</i>
-12: número inteiro negativo	<i>int numeroC = -12;</i>

Tipos de dados

❖ Dados numéricos reais

São aqueles que podem possuir componentes decimais ou fracionários, também podem ser positivos ou negativos. Exemplos:

24.01: número real positivo com duas casas decimais	<i>float numeroA = 24.01;</i>
144: número real positivo com zero casas decimais	<i>float numeroB = 144;</i>
-13.3: número real negativo com uma casa decimal	<i>float numeroC = -13.3;</i>
0.0: número real com uma casa decimal	<i>float numeroD = 0.0;</i>
0: número real com zero casas decimais	<i>float numeroE = 0;</i>

Tipos de dados

■ Dados literais

- São constituídos por uma sequência de caracteres contendo letras, dígitos e/ou símbolos especiais.
- Muitas vezes chamados de alfanuméricos (cadeia de caracteres), são declarados (na linguagem C) como: *char nome_da_string[tamanho];*
- Exemplos:
 - `char nome1[] = "Fulano";` `// Inicializa nome1`
 - `char nome2[9] = "Beltrano";` `// Inicializa nome2`
 - `char *nome3 = "Sicrano";` `// Inicializa nome3`

Tipos de dados

- Dados lógicos (*booleanos*)
 - Usados para representar os dois únicos valores lógicos possíveis: VERDADEIRO (TRUE) e FALSO (FALSE).
 - Muitas linguagens de programação possuem um tipo booleano que armazena os valores *true* ou *false*. Enquanto o C++ possui o tipo *bool*, na linguagem C, um valor *true* é qualquer valor diferente de 0, incluindo números negativos:

C	C++
<pre>int pode = 1; //verdadeiro int nao_pode = 0; //falso</pre>	<pre>bool nomeBool1 = true; bool nomeBool2 = false;</pre>

Tipos de dados

- Resumo:
 - ❖ Os dados numéricos dividem-se em duas classes:
 - **inteiros**: não possuem parte fracionária e podem ser positivos ou negativos;
 - **reais**: podem possuir parte fracionária e podem ser positivos ou negativos.
 - ❖ Os dados do tipo literal podem conter sequências de letras, dígitos ou símbolos especiais, delimitados por aspas (“ ”).
 - ❖ Os dados do tipo lógico só possuem dois valores possíveis (*true*: != 0 e *false*: 0).

Tipos de dados

Tipo	Num de bits	Intervalo	
		Início	Fim
char	8	-128	127
unsigned char	8	0	255
signed char	8	-128	127
int	16	-32.768	32.767
unsigned int	16	0	65.535
signed int	16	-32.768	32.767
short int	16	-32.768	32.767
unsigned short int	16	0	65.535
signed short int	16	-32.768	32.767
long int	32	-2.147.483.648	2.147.483.647
signed long int	32	-2.147.483.648	2.147.483.647
unsigned long int	32	0	4.294.967.295
float	32	3,4E-38	3.4E+38
double	64	1,7E-308	1,7E+308
long double	80	3,4E-4932	3,4E+4932

Tipos de dados

▪ Definição de variáveis

- ❖ Basicamente, uma variável possui três atributos: um nome (identificador), um tipo de dado associado e a informação armazenada.
 - Um nome de variável deve começar com uma letra;
 - Um nome de variável não deve conter nenhum caractere especial exceto a sublinha “_”.

salario	=	correto
1ano	=	errado
a casa	=	errado
sal/hora	=	errado
sal_hora	=	correto
_desconto	=	correto

Variáveis

■ Definição de variáveis

❖ Todas as variáveis devem ser definidas antes de serem utilizadas.

➤ Isso é necessário para permitir que o compilador reserve um espaço na memória para as mesmas.

❖ Sintaxe (em C):

<tipo_da_variável> <nome_da_variável>;

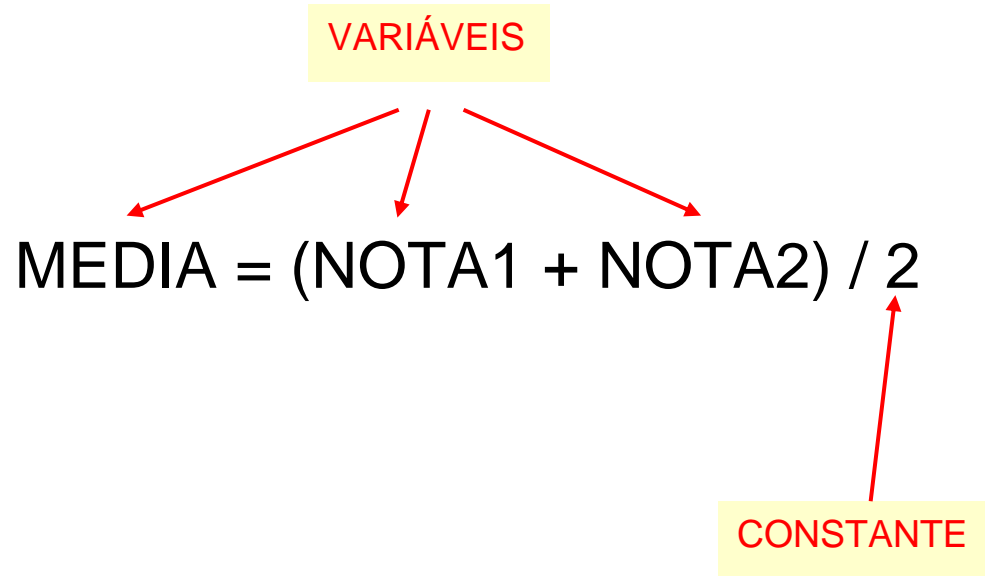
exemplos: int a; float b; char c;

<tipo_das_variáveis> <lista_de_variáveis>;

exemplos: int a, b, c; float d, e, f;

Variáveis

- Durante o processamento de dados o computador manipula dados que podem variar (VARIÁVEIS) e dados que não variam (CONSTANTES):



Constantes

- A **constante** é um dado com **valor fixo** que não se modifica durante o processamento (execução do programa).

1. Pode ser representado diretamente no programa:

MEDIA = (NOTA1 + NOTA2) / **2**

2. Em algumas linguagens pode ser declarado explicita e previamente no programa.

Em C:

1) ***#define PI 3.1415 //antes do início do código***

2) ***const double pi = 3.1415; //em qualquer parte do código***

Constantes


```
#include <stdio.h>
#define MAIORIDADE 18
const int aposentadoria = 65;
const int motorista = 16;

int main()
{
    int idade;
    printf("Digite sua idade: ");
    scanf("%d", &idade);

    if( idade >= MAIORIDADE )
        printf("Voce ja pode se alistar e dirigir.\n");
    else
        if( idade >= motorista )
            printf("Voce pode dirigir, mas nao pode se alistar.\n");
        else
            printf("Voce nao pode dirigir nem se alistar\n");

    if( idade >= aposentadoria )
        printf("Voce ja pode se aposentar!\n");
}
```

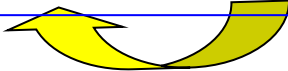
Constantes

- Fornecendo o valor da variável

1. Por atribuição

Sintaxe na linguagem C:

```
<nome_variavel> = <conteúdo>;
```



O conteúdo atribuído pode ser uma constante, uma variável ou uma expressão.

Exemplos:

```
Nome = "José Carlos";
```

```
Nota1 = 6; Nota2 = 5;
```

```
Media = (Nota1 + Nota2) / 2;
```

Entrada de Dados

- Fornecendo o valor da variável

2. Por entrada via teclado

Sintaxe na linguagem C:

```
scanf("%caracteres de controle",  
&lista_de_variaveis);
```

Exemplos:

```
scanf("%d %d", &m, &n);
```

```
scanf("%f", &numero);
```

Entrada de Dados

- Exibindo dados

Durante o processamento o valor de um dado (constante ou variável) pode ser exibido na tela.

Sintaxe na linguagem C:

```
printf("Mensagem a ser escrita na tela");
```

Exemplos:

```
printf("Aprovado");
```

```
float media = 8.5;
```

```
printf("Média do aluno: %f", media);
```

Exemplo do que será exibido no console: *Média do aluno: 8.5*

Saída de Dados

Para iniciar, considere o programa mais simples na linguagem que faz uso de um comando de saída e, simplesmente, exibe a mensagem **Bom dia!**

```
1. // Primeiro programa
int main()
{
    printf("Bom dia!\n");
    return 0;
}
```

* Todo programa em C tem a extensão .c.

Primeiro programa

Exemplos de **função** `main()` :

```
main()  
{  
    printf("Bom Dia!");  
}
```

```
void main()  
{  
    printf("Bom Dia!");  
}
```

```
int main()  
{  
    printf("Bom Dia!");  
    return 0;  
}
```

```
int main()  
{  
    printf("Bom Dia!");  
}
```

- Obter a área de um círculo, cujo o raio tem valor 4.0:

```
const double PI = 3.14159;
```

```
main (){  
    double circulo;  
    double raio = 4.0;  
    circulo = PI*raio*raio;  
    printf("RESULTADO = %f", circulo);  
}
```

Expressões com constantes

- Somar 2 números inteiros

```
void main()
{
    int numA, numB, soma;

    printf("Digite o primeiro numero inteiro: ");
    scanf("%d", &numA);

    printf("Digite o segundo numero inteiro: ");
    scanf("%d", &numB);

    soma = numA + numB;
    printf("Valor da soma = %d", soma);
}
```

Segundo programa

- Sugere-se o uso da IDE **Code::Blocks** pela facilidade de uso. Todavia, outros app's podem ser usados:

<http://www.codeblocks.org> (Recomendado para computador ou notebook)

<https://repl.it> (Recomendado para uso em qualquer dispositivo computacional, exigindo apenas um browser para escrever o código e testar)

<https://www.bloodshed.net/download.html> ou

<https://sourceforge.net/projects/orwelldvcpp/> (Recomendado para computador ou notebook)

IDEs

1. Explique o que está errado nos identificadores incorretos:

<input type="checkbox"/> valor	<input type="checkbox"/> _b248	<input type="checkbox"/> nota*do*aluno
<input type="checkbox"/> a1b2c3	<input type="checkbox"/> 3 x 4	<input type="checkbox"/> Maria
<input type="checkbox"/> km/h	<input type="checkbox"/> xyz	<input type="checkbox"/> nome empresa
<input type="checkbox"/> sala_215	<input type="checkbox"/> "nota"	<input type="checkbox"/> ah!

2. Supondo que as variáveis NT, NA, NMAT e SX sejam utilizadas para armazenar a nota do aluno, o nome do aluno, o número da matrícula e o sexo, declare-as corretamente, associando o tipo adequado ao dado que será armazenado.

Exercícios

Aula 02.2:



- Operador de atribuição
- Operadores aritméticos
- Incremento e decremento
- Prioridades dos operadores
- Trabalhando com percentual
- Exercícios

- A forma geral do operador de atribuição é:

nome_da_variável = expressão;

- onde *expressão* pode ser tão simples como uma única constante ou tão complexa quanto você necessite.
 - O “destino” *nome_da_variável* deve ser uma variável ou um ponteiro, não uma função ou uma constante.
- C permite que você atribua o mesmo valor a muitas variáveis usando atribuições múltiplas em um único comando. Exemplo:

$x = y = z = 0;$

Operador de atribuição

Operador	Objetivo
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto (módulo)
--	Decremento
++	Incremento

Operadores aritméticos

- Quando “/” é aplicado a um inteiro, qualquer resto é truncado. Por exemplo, $5 / 2 = 2$ em uma divisão entre tipos inteiros.
- O operador módulo “%” devolve o resto de uma divisão inteira. Contudo, não pode ser usado nos tipos em ponto flutuante (decimais). Exemplo:

```
int x, y;
```

```
x = 5; y = 2;
```

```
printf("%d", x/y); // mostrará 2
```

```
Printf("%d", x%y); // mostrará 1, o resto da divisão inteira
```

```
X = 1; y = 2;
```

```
printf("%d %d", x/y, x%y); // mostrará 0 1
```

Operadores aritméticos

- O operador ++ soma 1 ao seu operando, e -- subtrai 1:

$x = x + 1; \rightarrow ++x ; \text{ ou } x++ ;$

$x = x - 1; \rightarrow --x ; \text{ ou } x-- ;$

- Quando um operador de incremento ou decremento precede o seu operando, C executa a operação de incremento ou decremento antes de usar o valor do operando. Se o operador estiver após seu operando, C usará o valor do operando antes de incrementá-lo ou decrementá-lo. Exemplo:

$x = 10; y = ++x; \rightarrow y \text{ recebe } 11$

$x = 10; y = x++; \rightarrow y \text{ recebe } 10$

Incremento e decremento

- Conforme regra da aritmética, a ordem de precedência dos operadores básicos são:

1º Multiplicação(*), divisão(/) e resto(%)

2º Adição(+) e subtração(-)

- Exemplos:

$$10 + 20 * 30 = ?$$

$$50 - 8 / 2 = ?$$

$$10 \% 3 + 2 = ?$$

Prioridades dos operadores

- Em expressões com operadores de mesma prioridade, executa-se da esquerda para a direita:

- Exemplos:

$$10 + 20 - 30 = ?$$

$$50 * 8 / 2 = ?$$

$$20 + 10 * 150 - 60 / 3 = ?$$

Prioridades dos operadores

- A ordem de prioridade pode ser alterada através de parênteses.

- Exemplos:

$$(10 + 20) * 30 = ?$$

$$(50 - 8) / 2 = ?$$

$$20 + 10 * (150 - 60) / 3 = ?$$

Prioridades dos operadores

- Exemplo: código na linguagem C para realizar as operações a seguir:

$$(10 + 20) * 30 = 900$$

$$(50 - 8) / 2 = 21$$

$$20 + 10 * (150 - 60) / 3 = 320$$

Prioridades dos operadores

- Para achar o valor a partir do percentual de um número:

30 → 100%

$$X * 100 = 30 * 15$$

X → **15%**

$$X = (30 * 15) / 100$$

$$\text{VALOR DESEJADO} = \text{VALOR TOTAL} * \text{PERCENTUAL} / 100$$

Trabalhando com percentual

- Para achar o percentual a partir do valor de um número:

30 → 100%

$$X\% * 30 = 100 * 7$$

7 → X%

$$X\% = (100 * 7) / 30$$

$$\text{PERCENTUAL DESEJADO} = 100 * \text{VALOR CORRESPONDENTE} / \text{VALOR TOTAL}$$

Trabalhando com percentual

3. Utilizando como referência o 2º programa, escrever um novo programa para calcular a média aritmética a partir de duas notas.
4. Elaborar um programa que solicita ao usuário digitar o valor da temperatura em *Fahrenheit* (F) e a converte em *Celsius* (C), mostrando o resultado.

* Usar o Code::Blocks

Exercícios

5. Uma loja virtual decidiu oferecer vários descontos na *Black Friday*. Escreva um programa na linguagem C para ler o preço e o percentual de desconto para um produto. O programa deve exibir: o valor do desconto e o valor final a ser pago. Exemplo:

ENTRADA:

Preço: R\$100,00

Percentual de desconto: 10%

SAÍDA:

Valor do desconto: R\$10,00

Preço final: R\$90,00

Exercícios

	DATA	AULA
1	22/08/2024	Apresentação da disciplina Introdução à Programação Imperativa
2	29/08/2024	Introdução à Linguagem de Programação C
3	05/09/2024	Conceitos Fundamentais
4	12/09/2024	Tipos de Dados Especiais em C
5	19/09/2024	Estruturas Condicionais e de Repetição
6	26/09/2024	Pré-processamento
7	03/10/2024	Registros/Estruturas de Dados
8	10/10/2024	Ponteiros
9	17/10/2024	1º Exercício Escolar

Plano de Aulas

	DATA	AULA
10	24/10/2024	Arquivos
11	31/10/2024	Acompanhamento de projetos
12	07/11/2024	Acompanhamento de projetos
13	14/11/2024	Acompanhamento de projetos
14	21/11/2024	Acompanhamento de projetos
15	28/11/2024	Apresentação parcial
16	05/12/2024	Apresentação de projetos
17	12/12/2024	Avaliação Final

Plano de Aulas