Algoritmo ‘Lógica de’ {

[Programação 1]

}

Unidade 04

Prof. Daniel Caixeta

slidesgo

‘Sumário’ {

}

01 {

[Primeiros passos em C]

< Conceitos, sintaxe, primeiras linhas >

}

O que é uma linguagem de programação?

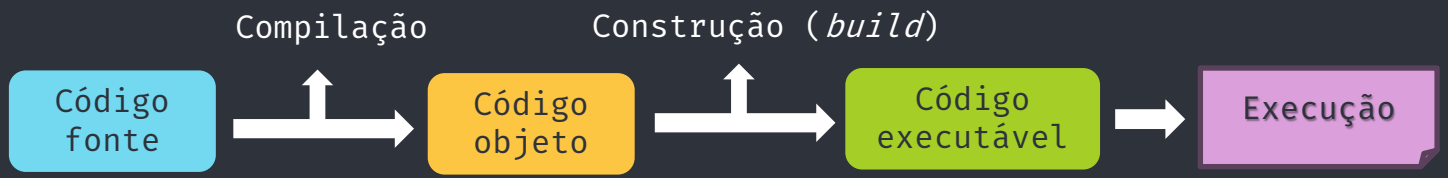
- É uma linguagem artificial e formal usada para comunicar instruções a um computador.
- Assim como uma linguagem natural, possui uma sintaxe, devendo ser rigorosamente seguida, sendo que em qualquer desvio, haverá impossibilidade do computador interpretar o código e executar o programa.
- A partir de exemplos práticos, vamos aprender a sintaxe desta linguagem.

O que é um programa em linguagem C?

- É um texto cujo o conteúdo segue a sintaxe da linguagem C.

1º programa em C – “Hello World!”

```c
#include <stdio.h>
int main()
{
    printf("Hello World!\n");
    return 0;
}
```



Compilação é o processo de tradução de um programa (código-fonte) para uma linguagem de máquina (código-objeto), para que suas instruções sejam executadas pelo processador (código executável).

- O programa HelloWorld.c ou main.c possui o sintaxe/formato básico de um programa em C.

```
#include <stdio.h>
int main()
{
    /* Declaração de variáveis */
    /* Sentenças ou lógica do programa */

    return 0;
}
```

Declaração da biblioteca

Inicia o programa e retorna um inteiro para o S.O.

Corpo do código

return: Encerra a função main retornando para o S.O.

Se retornar 0 (zero) programa bem sucedido, SENÃO, há uma condição de erro no código.

- O ponto e vírgula (;) não é exceção é REGRA!!!

1.2. Tipos de dados e variáveis

- Um programa minimamente útil precisa manipular dados na memória do computador.
- A memória armazena uma imensa sequência de bits, e sua representação é indiferente para o funcionamento da máquina.
- Linguagens como C, entretanto, facilitam a vida do programador disponibilizando tipos de dados em bibliotecas.
- Um tipo de dado é um conjunto de possíveis valores e operações que podem ser efetuadas.
- A linguagem C nos oferece alguns tipos de dados *primitivos*.

Os principais tipos

Tipo	Nome	Número de bits	Intervalo	
			Início	Fim
char()	Caractere	8	-128	127
int()	Inteiro	16	-32.768	-32.767
float()	Flutuante	32	3,4E-38	3.4E+38
double()	Duplo	64	1,7E-308	1,7E+308
void()	Nulo	-	-	-
long int()	Inteiro	32	-2.147.483.648	2.147.483.647
long double()	Duplo	80	3,4E-4932	3,4E+4932

- Na linguagem C é permitido declarar variáveis, que possuem um tipo determinado.
- Declarar uma variável é como dar um nome a um pedaço da memória que guardará um valor de determinado tipo. Daí em diante podemos acessar o valor guardado e manipulá-lo através do nome da variável. Por exemplo:

Exemplo <1>

```
int x, y, soma;

x = 2;
y = 4;

soma = x + y;
```

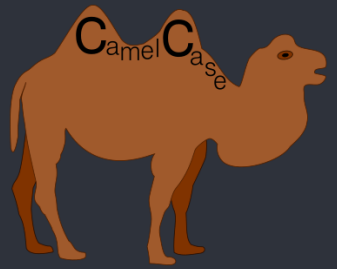
Exemplo <2>

```
int x = 5
int y, z;

y = 1 + 2 * x * (3 - 2 * x);
z = x * y;

printf("y = ", y);
printf("x = ", z);
```

- O nome de uma variável pode ser qualquer sequência de letras e números e alguns símbolos como _ .
- Um nome não pode começar com números e conter espaços. O uso de maiúsculas e minúsculas é permitido. Lembre-se que as linguagens de programação são *sensitive case*, ou seja, MAIÚSCULAS diferentes de minúsculas, de modo que soma e sOma são variáveis diferentes.

Lembre-se ?!?!

- Não pode começar com dígito: use uma letra ou _ .
- Não pode ter espaço em branco.
- Não usar acentos ou til.
- Sugestão: use o padrão “Camel Case“.



Exemplos: Soma, x, y, x12, x_12, soma_abc, nome_bem_grande, NomeBemGrande

1.3. Formatos de leitura de variável

- Os formatos de leitura da funções printf() e scanf() são bem semelhantes. Na tabela abaixo vejam alguns formatos possíveis de leitura e escrita.

Código	Função
%c	Lê um único caractere.
%s	Lê uma série de caracteres.
%d	Lê um número decimal.
%u	Lê um decimal sem sinal.
%ld	Lê um inteiro longo.
%f	Lê um número em ponto flutuante.
%lf	Lê um número double.

Exercício <1>:

- Faça um programa que realiza operações aritméticas com variáveis inteiras (a, b e c) atribuindo valores a essas constantes:

```c
/* Programa que calcula operações aritméticas
básicas. Manipulação de dados na memória.

Autor: dfcaixeta - Data: 06.mai.2024 */

#include <stdio.h>
#include <locale.h>

int main(){

    setlocale(LC_ALL, "");
    int a, b = 5, c = 4;
    a = b + c;
    b = a + 6;
    c = b - a;

    printf("\nOs valores das operações são:\n");
    printf("a = %d\n", a);
    printf("b = %d\n", b);
    printf("c = %d\n\n", c);

    return 0;
}
```

Exercício <2>:

- Qual o valor armazenado na variável a?

```c
/* Programa que calcula o valor da variável a ao final da execução.
Autor: dfcaixeta - Data: 06.mai.2024 */

#include <stdio.h>
#include <locale.h>

int main(){

    setlocale(LC_ALL, "");
    int a, b = 4, c = 2, d = 3;

    d = c + b;
    a = d + 1;
    a = a + 1;

    printf("\nO valor de a é: %d", a);
    return 0;
}
```

Necessário?!?!

Exercício <3>:

- Escreva o código abaixo e indique qual o ERROR!?

```c
/* Qual o erro nesse código? ... Descubra.
Autor: dfcaixeta - Data: 06.mai.2024 */

#include <stdio.h>
#include <locale.h>

int main(void) {
    setlocale(LC_ALL, "");

    int a, b = 4;
    double c = 2.4142, d = 3.1416;
    int g;

    // Cálculos das operações ...
    d = b + 5;
    e = c * d;
    a = b + 1;

    printf("\nO valor de b é: %d", b);
    printf("\nO valor de a é: %d", a);
    printf("\nO valor de e é: %.3f", e);

    return 0;
}
```

Exercício <4>:

- Faça programa que lê um caractere, um número ponto flutuante e por fim um decimal e imprima na tela os dados lidos.

```c
/* Programa que lê um char, float, int
e imprime na tela os dados lidos.
Data: 14.mai.2024
Autor: Daniel Caixeta <@dfcaixeta> */

# include <stdio.h>
# include <stdlib.h>

int main(){
    char c;
    float b;
    int a;

    printf("\nDigite um caractere: ");
    scanf("%c", &c);
    printf("Digite um numero ponto flutuante: ");
    scanf("%f", &b);

    printf("Digite um numero inteiro: ");
    scanf("%d", &a);

    printf("\nOs dados lidos foram: %c, %f, %d\n\n", c, b, a);

    system("pause");
    return 0;
}
```

Exercício <5>:

- Faça um programa que pede ao usuário dois números e depois imprime na tela a média aritmética entre eles:

```c
/* Programa que calcula a media entre dois números.
Autor: @dfcaixeta - Data: 07.mai.2024. */

// Declaração das bibliotecas
# include <stdio.h>
# include <locale.h>

// Corpo do código ..
int main()
{
    setlocale(LC_ALL, "");
    double num1, num2, media;

    printf("\nEntrada dos valores: \n");
    printf("Digite o primeiro número: ");
    scanf("%lf", &num1);
    printf("Digite o segundo número: ");
    scanf("%lf", &num2);

    // Lógica do cálculo ...
    media = (num1 + num2) / 2.0;

    // Imprime os resultados do cálculo ...
    printf("\nResultado: ");
    printf("\nMédia = %.1f\n\n", media);

    return 0;

}
```

Exercício <6>:

- Faça um programa que realiza as operações aritméticas básicas (soma, subtração, multiplicação e divisão):

```c
/* Programa que realiza operações aritméticas
entre dois números.
Autor: @dfcaixeta - Data: 07.mai.2024 */

# include <stdio.h>
# include <locale.h>

int main() {

    setlocale(LC_ALL, "");
    double num1, num2;

    printf("\n ** Entre com os valores: ** \n");
    printf("Digite o primeiro número: ");
    scanf("%lf", &num1);

    printf("Digite o segundo número: ");
    scanf("%lf", &num2);
```

--->

```c
    printf("\n ** Os valores são: **");
    printf("\nSoma = %.2lf\n", num1 + num2);
    printf("Subtração = %.2lf\n", num1 - num2);
    printf("Multiplicação = %.2lf\n", num1 * num2);

    if (num2 != 0) {
        printf("Divisão = %.2lf\n\n", num1 / num2);
    } else {
        printf("\nNão é possível dividir por zero.\n\n");
    }
    return 0;
}
```

02 {

[Expressões comparativas & Lógicas]

< Operadores de comparação e Lógicos >

}

2.1. Expressões de comparação

- Os operadores de comparação são símbolos usados em programação para comparar valores e expressões.
- Retornam um valor booleano (V ou F).
- Esses operadores são fundamentais para controle de fluxo e tomada de decisões em algoritmos/programas.

Operador	Significado
>	Maior
<	Menor
>=	Maior ou igual

Operador	Significado
<=	Menor ou igual
==	Igual
!=	Diferente

Operadores de comparação em Linguagem C.

- Por exemplo:

Expressão	V	F
`5 <= 8`		
`10 > 12`		
`pow(8, 2) == 64`		
`(2 * 4) != (9 - 1)`		

- Os operadores de comparação são amplamente utilizados em estruturas de controle condicional, como instruções *if*, *else if* e *while*, para tomar decisões com base nas relações entre os valores das variáveis.

2.2. Expressões lógicas

- São combinações de símbolos e operadores lógicos que representam relações entre proposições ou valores lógicos.
- Essas expressões são usadas na lógica formal e na programação para representar condições, decisões e raciocínios.
- Os operadores lógicos comuns incluem && (AND/E), || (OR/OU) e ! (NOT/NÃO), que são usados para combinar proposições ou inverter seu valor lógico.
- Por exemplo:

    "Se estiver chovendo || se estiver ensolarado, vou levar um guarda-chuva!“

2.3. Operadores lógicos

Operador	Significado
AND/E (&&)	Verdadeiro se todas as condições forem verdadeiras.
OR/OU (||)	Verdadeiro se pelo menos uma condição for verdadeira.
NOT/NÃO (!)	Verdadeiro se a condição for falsa.

Operador NOT

A	S ou A'
0	1
1	0

Operador AND

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

(A . B)

Operador OR

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

(A + B)

A IDEIA POR TRÁS DO OPERADOR “&&” (AND)

- TODAS as condições devem ser VERDADEIRAS!
- Por exemplo:

  Você pode obter uma habilitação de motorista se:

  1. For aprovado no exame psicotécnico, &&
  2. For aprovado no exame de legislação, &&
  3. For aprovado no exame de direção.

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

EXEMPLOS DE EXPRESSÕES LÓGICAS

- Assumindo X = 3, então:

  1. `(X <= 16) && (X == 9)` Resultado:

  2. `(X > 1) && (X < 6)` Resultado:

  3. `(X < 7) && (X > 2) && (X != 3)` Resultado:

A IDEIA POR TRÁS DO OPERADOR “||” (OR)

- Pelo menos UMA condições deve ser VERDADEIRA!
- Por exemplo:

  Você pode estacionar na vaga especial se:

  1. For idoso(a), ||
  2. For uma pessoa com deficiência, ||
  3. For uma gestante.

A	B	A || B
0	0	0
0	1	1
1	0	1
1	1	1

EXEMPLOS DE EXPRESSÕES LÓGICAS

- Assumindo X = 3, então:

1. `(X <= 5) || (X == 9)` Resultado:
2. `(X > 1) || (X < 6)` Resultado:
3. `(X <= 10) || (X == 6) || (X != 3)` Resultado:
4. `(X <= 3) || (X == 3) && (X != 3)` Resultado:
5. `(X > 2) && (X == 3) || (X <= 4)` Resultado:

Expressões híbridas

A IDEIA POR TRÁS DO OPERADOR “!” (NOT)

- O operador “NOT” inverte a condição.
- Por exemplo:

  Você tem direito a receber uma bolsa de estudos se você:

  - NOT

    Possuir renda maior que R$ 2.500,00

Golias
R$: 2.850,00

V

Davi
R$: 2.300,00

N

A	S ! A’
0	1
1	0

EXEMPLOS DE EXPRESSÕES LÓGICAS

- Assumindo X = 3, então:

  1. `! (X == 10)` Resultado:

  2. `! (X >= 2)` Resultado:

  3. `! ((X <= 8) && (X == 10))` Resultado:

  4. `((X <= 3) || !(X < 5))` Resultado:

03 {

[Estrutura Condicional]

< SE ... ENTÃO ... SENÃO ... FIMSE.
E o pensamento computacional >

}

3.1. Conceito

</1> É uma estrutura de controle que permite definir que um certo bloco de comandos somente será executado dependendo de uma condição.

</2> Portanto, o comando só será executado se a condição for verdadeira. Uma condição é uma comparação que possui dois valores possíveis: verdadeiro ou falso.v

Estrutura condicional SIMPLES

```
SE <condição> ENTAO
.... <comando1>
.... <comando2>
FIMSE
```

REGRA

V: Executa o bloco de comandos.

F: Pula o bloco de comandos.

Observação importante: Reparem na indentação.

Estrutura condicional COMPOSTA

```
SE <condição> ENTAO
.... <comando1>
.... <comando2>
SENAO
.... <comando3>
.... <comando4>
FIMSE
```

REGRA

V: Executa somente o bloco do SE.

F: Executa somente o bloco do SENÃO.

Observação importante: Reparem na indentação.

Operador	Significado
AND/E (&&)	Verdadeiro se todas as condições forem verdadeiras.
OR/OU (||)	Verdadeiro se pelo menos uma condição for verdadeira.
NOT/NÃO (!)	Verdadeiro se a condição for falsa.

3.2. O encadeamento [...].

Estrutura condicional encandeada

```
SE <condição1> ENTAO
.... <comando1>
.... <comando2>
SENAO
     SE <condição2> ENTAO
..... <comando3>
..... <comando4>
     SENAO
..... <comando5>
..... <comando6>
     FIMSE
FIMSE
```

3.3. Estrutura condicional em C

- E finalmente, o C chegou ... Antes, vou apresentar apenas a sintaxe da estrutura condicional simples e composta nessa linguagem de programação.

Estrutura condicional SIMPLES

```
if (condição)
 {
.... comando1;
.... comando2;
 }
```

Em C, torna-se obrigatória a utilização de chaves {} quando existe mais de um comando a executar, sendo que só serão executados se a condição for verdadeira.

Estrutura condicional COMPOSTA

```
if (condição)
 {
.... comando1;
 }
else
 {
.... comando2;
 }
```

04 {

[Incremento, Decremento e Conversão]

< ++a ... a++ ... --a ... a--. E conversões de tipos.>

}

4.1. Incremento & Decremento

</1> É muito comum escrevermos expressões para incrementar/decrementar o valor de uma variável por 1.

</2> Em C, o operador unário ++ é usado para incrementar 1 ao valor de uma variável.

```
a = a + 1; -> a++;
```

</3> O operador unário -- é usado para decrementar de 1 o valor de uma variável. Por exemplo:

```
a = a - 1; -> a--;
```

Portanto:

Incremento -> a++

Decremento -> a--

- Há uma diferença quando estes operadores são usados à esquerda ou à direita de uma variável e fizerem parte de uma expressão maior:

  `++a` : Neste caso o valor de `a` será incrementado antes e só depois o valor de `a` é usado na expressão.

  `a++`: Neste caso o valor de `a` é usado na expressão maior, e só depois é incrementado.

- A mesma coisa acontece com o operador --.

```c

/* Programa que incremente e decrementa os valores nas variáveis.
Autor: Daniel Caixeta <@dfcaixeta> - Data: 25.mai.24 */

# include <stdio.h>

int main() {
    int a = 5, b, c = 5, d, e = 5, f, g = 5, h;

    b = ++a; // ‘a’ é incrementa antes e só depois usado na expressão.
    d = c++; // 'c’ é usado na expressão e depois é incrementado.
    f = --e; // 'e' será decrementado antes e depois usado na expressão.
    h = g--; // 'g’ é usado na expressão e depois é decrementado.

    printf(" b: %d \n", b); // Resultado esperado 6.
    printf(" d: %d \n", d); // Resultado esperado 5.
    printf(" f: %d \n", f); // Resultado esperado 4.
    printf(" h: %d \n", h); // Resultado esperado 5.

    return 0;
}

```

4.2. Atribuições simplificadas

</1> Uma expressão da forma: `a = a + b`, onde ocorre uma atribuição a uma das variáveis da expressão pode ser simplificada da seguinte forma: `a += b`.

Comando	Exemplo	Correspondente a
+=	a += b;	a = a + b;
-=	a -= b;	a = a - b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;
%=	a %= b;	a = a % b;

4.3. Conversões de tipos

</1> É possível converter alguns tipos entre si.

</2> Existem duas formas: Implícita e Explícita:

</3> Implícita:

<p> Capacidade (tamanho) do destino deve ser maior que a origem senão há perda de informação.

Ex_1.: `int a; short b; a = b;`

Ex_2.: `float a; int b = 10; a = b;`

</4> Explícita:

<p> Explicitamente informa o tipo que o valor da variável ou expressão é convertida.

Ex_1. a = (int) ((float)b / (float)c);

<p> Não modifica o tipo “real” da variável, só o valor de uma expressão.

Ex_2. int a; (float)a = 1.0; ← Errado

4.4. Informações extras [...]

- A operação de divisão (/) possui dois modos de operação de acordo com os seus argumentos: inteira ou de ponto flutuante.
- Se os dois argumentos forem inteiros, acontece a divisão inteira. 10 / 3 == 3.
- Se um dos dois argumentos for de ponto flutuante, acontece a divisão de ponto flutuante.
- 1.5 / 3 == 0.5.
- Quando se deseja obter o valor de ponto flutuante de uma divisão (não-exata) de dois inteiros, basta converter um deles para ponto flutuante;:

- Quando se deseja obter o valor de ponto flutuante de uma divisão (não-exata) de dois inteiros, basta converter um deles para ponto flutuante.
- Por exemplo:
- A expressão 10 / (float) 3 tem como valor 3.33333333

05 {

[Expressões e Operações relacionais]

< == ... != ... > ... <. Etc.>

}

5.1. Expressões e Operadores relacionais [...]

</1> Expressões relacionais são aquelas que realizam uma comparação entre duas expressões e retornam:

Zero 0; se o resultado for Falso.

Um 1; ou qualquer outro número diferente de 0, se o resultado for Verdadeiro.

</2> Os operadores relacionais em Linguagem C são:

Comando	Operador	Comando	Operador
==	Igualdade	<	Menor
!=	Diferente	>=	Maior ou igual que
>	Maior	<=	Menor ou igual que

</3> Expressões relacionais:

<p1> expressão == expressão: Retorna verdadeiro quando as expressões forem iguais.

9 == 9 // Resultado é 1, True.

9 == 10 // Resultado é 0, False.

<p2> expressão != expressão: Retorna verdadeiro quando as expressões forem diferentes.

9 != 9 // Resultado é 0, False.

9 != 10 // Resultado é 1, True.

<p3> expressão > expressão: Retorna verdadeiro quando a expressão da esquerda tiver valor maior que a expressão da direita.

```
9 > 5 // Resultado é 1, True.
```

<p4> expressão < expressão: Retorna verdadeiro quando a expressão da esquerda tiver valor menor que a expressão da direita.

```
9 < 5 // Resultado é 0, False.
```

<p5> expressão >= expressão: Retorna verdadeiro quando a expressão da esquerda tiver valor maior ou igual que a expressão da direita.

```
9 >= 5 // Resultado é 1, True.
```

<p6> expressão <= expressão: Retorna verdadeiro quando a expressão da esquerda tiver valor menor ou igual que a expressão da direita.

```
9 <= 5 // Resultado é 0, False.
```

Referência

</1> ASCENCIO, Ana Fernanda G. ARAÚJO, Graziele. S. Estruturas de Dados: algoritmos, análise da complexidade e implementações em Java e C/C++. São Paulo: Pearson Prentice Hall, 2010. (Biblioteca virtual).

</2> PUGA, Sandra. RISSETTE, Gerson. Lógica de Programação e Estrutura de Dados. 2. ed. São Paulo? Pearson Prentice Hall, 2009.

A ‘Obrigado’ Is {

Algoritmo & Linguagem de Programação 1

}