

Algoritmos

Aula 07 – Linguagem C: elementos básicos

Professora

Laysa Mabel de Oliveira Fontes

mabel.fontes@ufersa.edu.br

Pau dos Ferros/RN

2022

O que é?

É um conjunto de regras detalhadas para cada construção válida em uma linguagem de programação.

Sintaxe

- Considerando a linguagem C, essas regras estão relacionadas com:
 - Diretivas;
 - Constantes;
 - Variáveis;
 - Identificadores;
 - Tipos de dados;
 - Declaração de variáveis;
 - Atribuições;
 - Estruturas de decisão;
 - Estruturas de repetição;
 - Funções;
 - Vetores;
 - Matrizes.

Códigos de Formatação

Esses códigos são utilizados nas funções de entrada (*scanf*) e saída (*printf*) para informar o tipo de dado que está sendo tratado.

Código	Tipo
%c	char
%i	int
%f	float
%lf	double
%s	string

Função de Saída

printf

As informações definidas na função printf são exibidas para o usuário através de um dispositivo de saída de dados, normalmente o monitor.

Função de Saída

```
printf("<texto/código de formatação>", <constante/variável/expressão>);
```

- Exemplo:

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

int main(){
    setlocale(LC_ALL, "Portuguese");
    float f, c;
    printf("Digite a temperatura em Fahrenheit: ");
    scanf("%f", &f);
    c = (f - 32) / 1.8;
    printf("A temperatura em Celsius é: %.1f", c);
    return(0);
}
```

Função de Entrada


scanf

A função scanf permite ler valores fornecidos pelo usuário, via teclado, e armazená-los em variáveis do programa.

Função de Entrada

`scanf("<código de formatação>", &<identificador da variável>);`

- Exemplo:



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

int main(){
    setlocale(LC_ALL, "Portuguese");
    float f, c;
    printf("Digite a temperatura em Fahrenheit: ");
    scanf("%f", &f);
    c = (f - 32) / 1.8;
    printf("A temperatura em Celsius é: %.1f", c);
    return(0);
}
```


Códigos Especiais

Esses códigos são utilizados na função *printf*.

Código	Significado
\n	Faz com que o cursor pule uma linha
\t	Produz uma tabulação horizontal
\a	Produz um sinal sonoro
\”	Exibe na tela uma ”
\\	Exibe na tela uma \
%%	Exibe na tela um %

O que são?

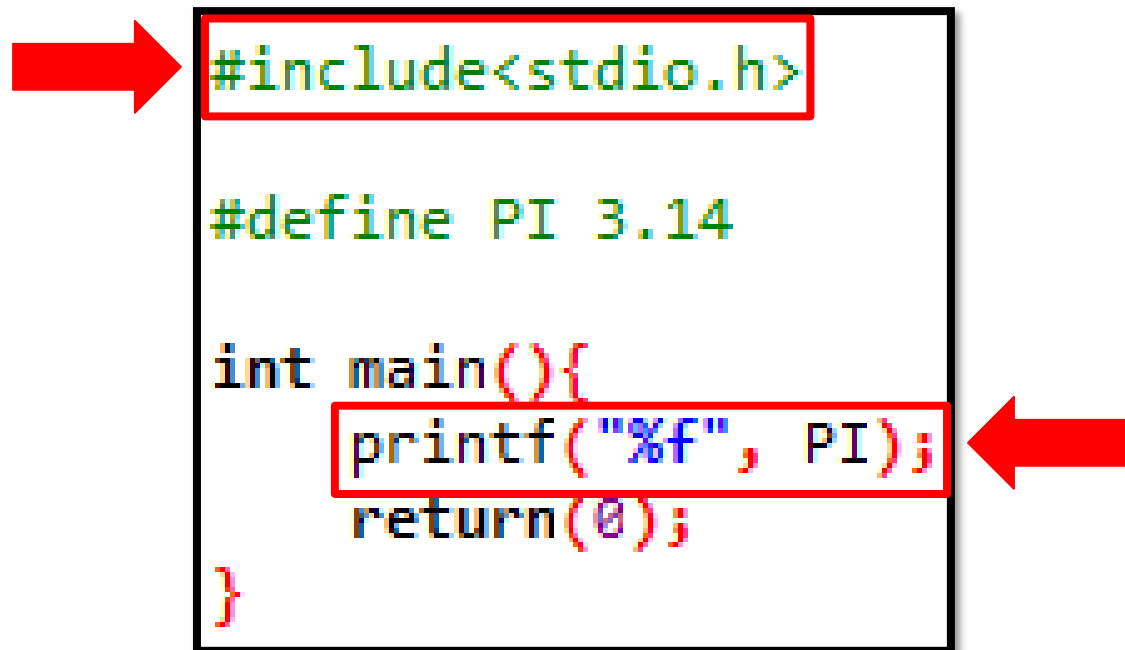
São comandos que não são compilados, sendo dirigidos ao pré-processador, que é executado pelo compilador antes da execução do processo de compilação propriamente dito.

Diretivas

- As diretivas são iniciadas pelo caractere #.
- Exemplos:
 - #include
 - #define

Diretivas

- A diretiva `#include` permite incluir uma biblioteca.
- Exemplo:



```
#include<stdio.h>

#define PI 3.14

int main(){
    printf("%f", PI);
    return(0);
}
```

Diretivas

- Exemplos:

Diretiva	Significado
<code>#include<stdio.h></code>	Funções de entrada e saída
<code>#include<stdlib.h></code>	Funções padrão
<code>#include<math.h></code>	Funções matemáticas
<code>#include<string.h></code>	Funções de texto
<code>#include<locale.h></code>	Função para definir o idioma/região

Diretivas

- A diretiva #define permite definir uma constante.
- Exemplo:

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



```
#define PI 3.14
```

```
int main(){
```

```
    printf("%f", PI);
```

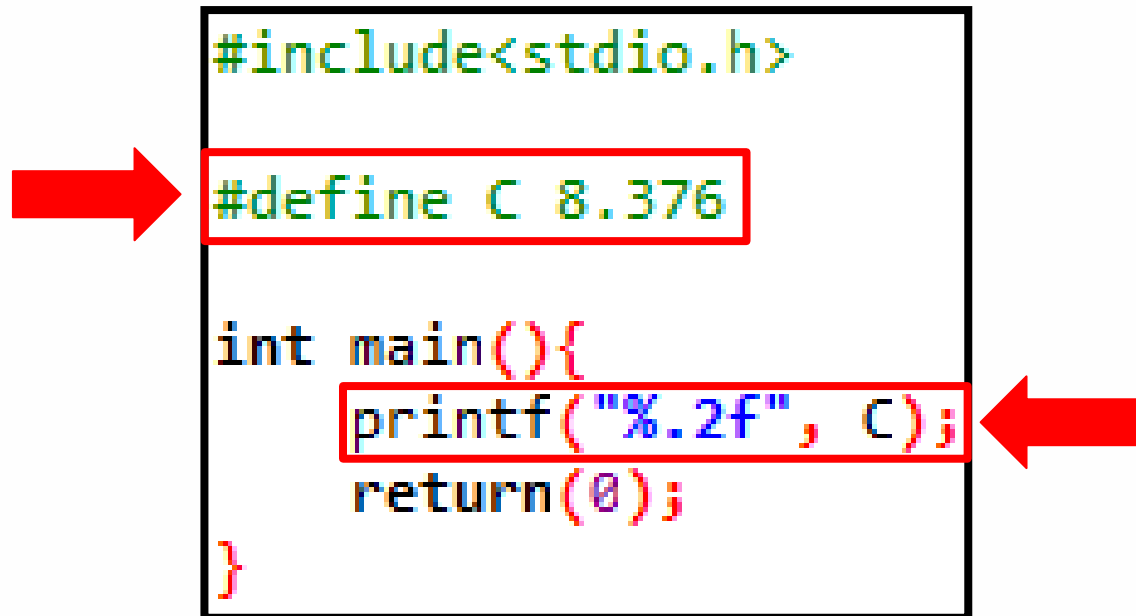


```
    return(0);
```

```
}
```

Diretivas

- A diretiva #define permite definir uma constante.
- Exemplo:



```
#include<stdio.h>

#define C 8.376

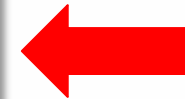
int main(){
    printf("%.2f", C);
    return(0);
}
```

Constantes

- A palavra reservada **const** também permite definir uma constante.
- Exemplo:

```
#include<stdio.h>

int main(){
    const int p = 30;
    printf("%i", p);
    return(0);
}
```



Variáveis

Variáveis

São espaços alocados na memória principal para armazenar algum dado.

Identificadores

Identificadores

São nomes usados para se fazer referência a variáveis e funções.

Identificadores

- Em C, um identificador deve ser constituído apenas de:
 - Letra;
 - Número;
 - *Underline*.
- Além disso, não se pode:
 - Começar com número;
 - Ter acentos;
 - Ter espaços;
 - Ser uma palavra reservada.

Identificadores

- Exemplos válidos:

- media
- _media
- nota2
- media_final

- Exemplos inválidos:

- média
- 2nota
- media final
- nome-completo

Identificadores

A linguagem C é *case sensitive*, ou seja, faz distinção entre letras maiúsculas e minúsculas.

- Exemplo:

media



Media



Tipos de Dados

- Exemplos de tipos de dados da linguagem C:

Tipo	Natureza	Bits na Memória
char	Caractere	8
int	Inteiro	32
float	Real	32
double	Real	64

char

Representa um caractere entre apóstrofos, podendo ser um número, letra ou caractere especial.

- **Exemplo:**

```
#include<stdio.h>

int main(){
    char a;
    a = 'm';
    printf("%c", a);
    return(0);
}
```

int

Toda e qualquer informação numérica que pertença ao conjunto dos números inteiros.

- Exemplo:

```
#include<stdio.h>

int main(){
    int y;
    y = 35679987;
    printf("%i", y);
    return(0);
}
```


float

Toda e qualquer informação numérica que pertença ao conjunto dos números reais.

- Exemplo:

```
#include<stdio.h>

int main(){
    float r;
    r = 5.678;
    printf("%f", r);
    return(0);
}
```

double

Toda e qualquer informação numérica que pertença ao conjunto dos números reais.

- Exemplo:

```
#include<stdio.h>

int main(){
    double w;
    w = 123423.6657567;
    printf("%lf", w);
    return(0);
}
```

Outros Tipos de Dados – Valores Lógicos

- C não possui um tipo de dados para valores lógicos:
 - Verdadeiro;
 - Falso.

Outros Tipos de Dados – Valores Literais

- C não possui um tipo de dados para valores literais (string/texto)
- Para utilizar valores literais, deve-se utilizar:
 - Um vetor do tipo char (assunto da terceira unidade)

Declaração de Variáveis

Para que o computador possa executar comandos que envolvem variáveis da maneira correta, ele deve conhecer os detalhes das variáveis que pretendemos utilizar.

- Esses detalhes são:
 - O identificador desta variável;
 - O tipo de valores que essa variável irá conter.

Declaração de Variáveis

<tipo da variável> <identificador da variável>;

- Exemplo:

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

int main(){
    setlocale(LC_ALL, "Portuguese");
    int soma;
    soma = 3 + 9;
    printf("O resultado da soma é: %i", soma);
    return(0);
}
```


Atribuições

Uma atribuição, representada pelo operador =, define a ação de atribuir um determinado valor a uma variável.

- Exemplo:

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

int main(){
    setlocale(LC_ALL, "Portuguese");
    int soma;
    soma = 3 + 9;
    printf("O resultado da soma é: %i", soma);
    return(0);
}
```



Atribuições

- O valor a ser atribuído pode ser:
 - Uma constante;
 - Uma variável;
 - Uma expressão.
- Exemplos:

```
float y = PI; // Onde PI é uma constante  
int z = x; // Onde x é uma variável  
int w = x + 4; // Onde x + 4 é uma expressão
```



Inicialização

Às vezes, é desejável que uma variável assuma um certo valor logo no início do bloco de instruções, antes de sua manipulação.

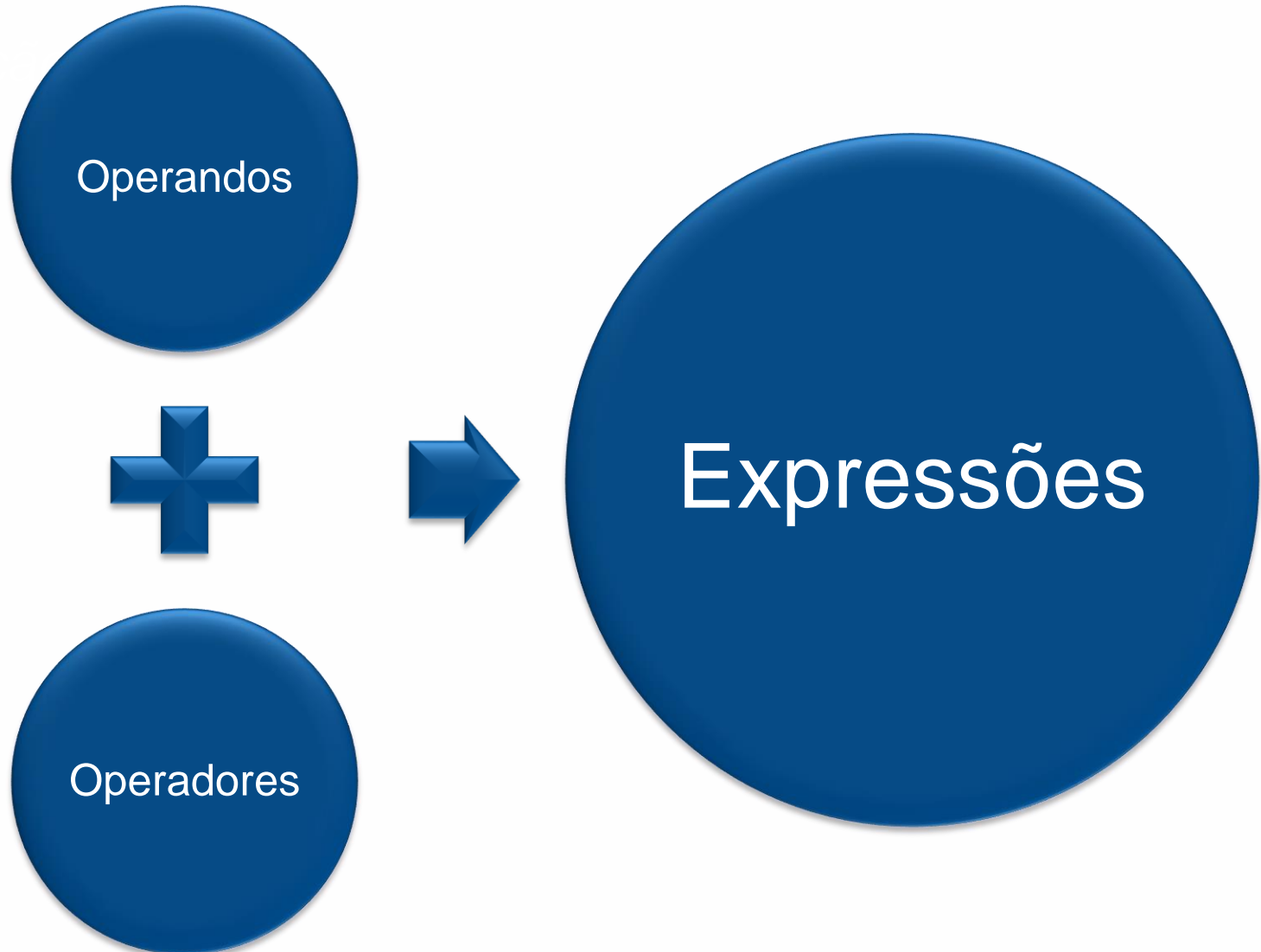
- Exemplo:

```
#include<stdio.h>

int main(){
    int p;
    p = 0;
    p = p + 3;
    printf("%i", p);
    return(0);
}
```



Expressões



Operandos

São os elementos de uma expressão que sofrem uma ação.

- **Exemplos:**
 - Variáveis;
 - Valores;
 - Outras expressões.

Operadores

São os elementos de uma expressão que realizam a ação.

- **Exemplos:**
 - Operadores aritméticos;
 - Operadores relacionais;
 - Operadores lógicos.

Expressões

```
#include<stdio.h>

int main(){
    int x;
    x = 3 + 12;
    printf("%i", x);
    return(0);
}
```

- Na expressão $x = 3 + 12$, temos:

x
3
12

} Operandos

+
=

} Operadores

Operadores Aritméticos

Operador Aritmético	Linguagem C
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Módulo (resto da divisão)	%

* A linguagem C não possui operador de exponenciação, mas possui uma função predefinida para tal: *pow(<base>, <expoente>)*. Por exemplo, para calcular x^y deve-se utilizar *pow(x, y)*. Para utilizar a função *pow*, deve-se incluir a biblioteca *math.h*.

Operadores Aritméticos

Expressão	Resultado
$3 + 12$	15
$3.1 - 1$	2.1
$2 * 1.5$	3
$5 / 2$	2
$5.0 / 2.0$	2.500000
$10 \% 5$	0

Operadores Aritméticos

- Exemplos:

```
#include<stdio.h>

int main(){
    printf("%i", 5 / 2);
    return(0);
}
```



2

```
#include<stdio.h>

int main(){
    printf("%f", 5.0 / 2.0);
    return(0);
}
```



2.500000

Operadores Aritméticos

Operador Aritmético	Linguagem C
Incremento	++
Decremento	--

Operadores Aritméticos

- Exemplo:

```
#include<stdio.h>

int main(){
    int h = 0;
    h++; ←
    printf("%i", h);
    return(0);
}
```

Operadores Aritméticos

Forma Tradicional	Forma Equivalente
<code>cont = cont + 1;</code>	<code>cont++;</code>
<code>cont = cont - 1;</code>	<code>cont--;</code>

Expressões Lógicas

Podem ser consideradas afirmações que serão testadas pelo computador.

- Em C, as expressões lógicas retornam:
 - 1 para verdadeiro;
 - 0 para falso.
- São utilizadas com os operadores relacionais e lógicos.

Operadores Relacionais

Operador Relacional	Linguagem C
Maior	>
Menor	<
Maior ou igual	>=
Menor ou igual	<=
Igual	==
Diferente	!=

Operadores Relacionais

Expressão	Resultado
$1 == 2$	0
$"A" == "a"$	0
$5 > 2$	1
$3 \leq 3$	1
$2 + 3 \neq 5$	0

Operadores Lógicos

Operador Lógico	Linguagem C
AND	&&
OR	
NOT	!

Operadores Lógicos

- **&&**
 - Resulta verdadeiro se todas as partes forem verdadeiras.
- **||**
 - Resulta verdadeiro se ao menos uma das partes for verdadeira.
- **!**
 - Nega a expressão, ou seja, inverte o valor.

Operadores Lógicos

- Exemplo do operador **&&** (AND):

$(n > 0) \ \&\& \ (n \% 2 == 0)$

Operadores Lógicos

- Exemplo do operador **||** (OR):

$$(i \geq 65) \parallel (t \geq 30)$$

Operadores Lógicos

- Exemplo do operador ! (NOT):

$$!(n \% 2 == 0)$$

Operadores Lógicos

- Tabela verdade do operador && (AND):

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

Operadores Lógicos

- Tabela verdade do operador && (AND):

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

Operadores Lógicos

- Tabela verdade do operador ! (NOT):

A	! A
1	0
0	1

Prioridade dos Operadores



Operador	Associatividade
! ++ --	Direita para esquerda
* / %	Esquerda para direita
+ -	Esquerda para direita
< <= > >=	Esquerda para direita
== !=	Esquerda para direita
&&	Esquerda para direita
	Esquerda para direita

Prioridade dos Operadores

- Exemplo:

$$3 * 7 + 16 / 8 - 4 == 18 \parallel 10 \% 4 >= 6 - 4$$

$$21 + 2 - 4 == 18 \parallel 2 >= 6 - 4$$

$$19 == 18 \parallel 2 >= 2$$

$$19 == 18 \parallel 1$$

$$0 \parallel 1$$

1

Comentários

Os comentários são declarações não compiladas que podem conter qualquer informação textual para referência e documentação de seu programa.

- São representados de duas formas distintas:
 - `/* */`
 - `//`

Comentários

- Exemplo:

```
/*  
    UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO  
    DISCIPLINA: ALGORITMOS  
    PROFESSORA: MABEL FONTES  
*/  
// Este programa exibe o dobro de um número inteiro  
  
#include<stdio.h>  
#include<locale.h>  
  
int main(){  
    setlocale(LC_ALL, "Portuguese");  
    int n;  
    printf("Digite um número inteiro: ");  
    scanf("%i", &n);  
    printf("O dobro é: %i", n * 2);  
    return(0);  
}
```

Vamos exercitar?

1. Você foi procurado por uma nutricionista para automatizar o cálculo do Índice de Massa Corpórea (IMC) de seus pacientes. Para isto, crie um programa em C que solicita e lê o peso e a altura de uma pessoa e apresenta seu IMC com duas casas decimais após a vírgula. O IMC é calculado por meio da seguinte fórmula:

$$IMC = \frac{peso}{altura^2}$$