

Aula 3:

Variáveis, Entrada de Dados e Operadores

Disciplina: Fundamentos de Programação

Prof. Luiz Olmes

olmes@unifei.edu.br



Nas aulas anteriores...

▶ **O QUE JÁ ESTUDAMOS?**

- ▶ Algoritmos.
- ▶ Linguagem C.
 - ▶ Função de saída `printf()`.

▶ **OBJETIVOS:**

- ▶ Variáveis e tipos.
- ▶ Função de entrada `scanf()`.
- ▶ Operadores.

Segundo programa em C

- ▶ Somando dois números:

segundo.c

```
1.  #include <stdio.h>
2.
3.  int main() // Função Principal
4.  {
5.      // Variáveis
6.      int a = 2;
7.      int b = 3;
8.      int soma = a + b;
9.
10.     printf("Resultado da soma de A e B = %d\n", soma);
11.
12.     return 0;
13. } // Fim
```

Segundo programa em C: resultado da execução

- ▶ Somando dois números:

segundo.c

```
1.  #include <stdio.h>
2.
3.  int main() // Função Principal
4.  {
5.      // Variáveis
6.      int a = 2;
7.      int b = 3;
8.      int soma = a + b;
9.
10.     printf("Resultado da soma de A e B = %d\n", soma);
11.
12.     return 0;
13. } // Fim
```



Resultado da soma de A e B = 5

Segundo programa em C: análise

```
1.  #include <stdio.h>
2.
3.  int main() // Função Principal
4.  {
5.      // Variáveis
6.      int a = 2;
7.      int b = 3;
8.      int soma = a + b;
9.
10.     printf("Resultado da soma de A e B = %d\n", soma);
11.
12.     return 0;
13. } // Fim
```

Declaração de variáveis:
Terminam com ponto e vírgula!

Variáveis

- ▶ `int a = 2;`
- ▶ `int b = 3;`
- ▶ `int soma = a + b;`

- ▶ Declaração de três variáveis chamadas de `a`, `b` e `soma`.
- ▶ As variáveis armazenam números inteiros. Por isso, o tipo `int` é colocado antes de seus nomes.

- ▶ O operador `=` (igual) permite realizar atribuições:
 - ▶ Armazena o valor 2 na variável `a`.
 - ▶ Armazena o valor 3 na variável `b`.
 - ▶ Armazena a soma das variáveis `a` e `b` na variável `soma`.
 - ▶ Dica: leia esse operador como: “**recebe o valor (de)**”:
 - ▶ `a` recebe o valor 2, `soma` recebe o valor de `a + b`.

Variáveis

- ▶ Variáveis em C são usadas para nomear posições de memória.
- ▶ Toda variável tem, obrigatoriamente, um tipo:
 - ▶ Dois tipos foram apresentados no desenvolvimento de pseudocódigos:
 - ▶ Um tipo para armazenar **números inteiros**: **int** em C.
 - ▶ Um tipo para armazenar **números reais**: **float** ou **double** em C.

Variáveis


- ▶ Variáveis em C são usadas para nomear posições de memória.
- ▶ Toda variável tem, obrigatoriamente, um tipo:
 - ▶ Dois tipos foram apresentados no desenvolvimento de pseudocódigos:
 - ▶ Um tipo para armazenar **números inteiros**: **int** em C.
 - ▶ Um tipo para armazenar **números reais**: **float** ou **double** em C.
- ▶ Nomes de variáveis são uma sequência de **letras** e **dígitos**.
- ▶ Sempre iniciam por uma letra (**'A'** – **'Z'**, **'a'** – **'z'**) ou pelos símbolos de underline (**_**) ou cifrão (**\$**).
- ▶ **Nunca iniciam com um dígito (**'0'** – **'9'**).**
- ▶ **Nunca contêm espaços.**
- ▶ **Não podem ser palavras reservadas.**

Nomes de variáveis: quais são válidos?

- ▶ A
- ▶ Matricula
- ▶ X
- ▶ 2X
- ▶ X2
- ▶ Nome do aluno
- ▶ Nome_do_aluno
- ▶ _teste
- ▶ A32B
- ▶ \$_classe
- ▶ Apartamento(201)
- ▶ Sala_31
- ▶ Sala_3.1
- ▶ UNIFEI


Nomes de variáveis: quais são válidos?


▶ A 


▶ Matricula 

▶ X 


▶ 2X 

▶ X2 

▶ Nome do aluno 


▶ Nome_do_aluno 


▶ _teste 

▶ A32B 

▶ \$_classe 

▶ Apartamento(201) 

▶ Sala_31 

▶ Sala_3.1 

▶ UNIFEI 

Variáveis

- ▶ Nomes de variáveis podem conter quantos caracteres forem desejados.
- ▶ A linguagem C **faz distinção** entre maiúsculas e minúsculas:
 - ▶ **soma**, **Soma** e **SOMA** são variáveis distintas.
- ▶ É comum usar apenas letras minúsculas para nomes de variáveis.
- ▶ Uma variável **não** pode ter o mesmo nome de uma **palavra chave** (palavra reservada) da linguagem.

Palavras reservadas da linguagem C

Palavras reservadas			
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Tipos Primitivos

- ▶ Tipos primitivos armazenam um único **valor atômico**.
- ▶ C é uma linguagem fortemente tipada:
 - ▶ Toda variável deve ter seu tipo explicitamente declarado.
- ▶ C possui 5 tipos primitivos:
 - ▶ 1 para manipular números inteiros.
 - ▶ 2 para manipular números de ponto flutuante (números reais).
 - ▶ 1 para manipular caracteres.
 - ▶ 1 tipo sem valor.
- ▶ O tamanho e a faixa de valores de cada tipo variam de acordo com:
 - ▶ A arquitetura do processador.
 - ▶ Implementação do compilador C.

Tipos Primitivos

Tipo	Tamanho	Aplicação	Intervalo
int	4 bytes	Armazena números inteiros: (conjunto \mathbb{Z})	-2147483647 a $+2147483647$
float	4 bytes	Armazena números reais: (com parte fracionária: conjunto \mathbb{R})	<i>Intervalo Negativo:</i> $-3.4 \text{ E}+38$ a $-1.4 \text{ E}-45$ <i>Intervalo Positivo:</i> $1.4 \text{ E}-45$ a $3.4 \text{ E}+38$ (6 algarismos significativos)
double	8 bytes	Armazena números reais de precisão dupla: (conjunto \mathbb{R}) Possui o dobro da precisão do tipo float.	<i>Intervalo Negativo:</i> $-1.79 \text{ E}+308$ a $-4.94 \text{ E}-324$ <i>Intervalo Positivo:</i> $4.94 \text{ E}-324$ a $1.79 \text{ E}+308$ (10 algarismos significativos)
char	1 byte	Armazena um único caractere	-128 a $+127$
void	1 byte	Não armazena valores	---

Modificador `long`

- Para obter um tamanho de variável maior que os tamanhos padrões, utiliza-se o modificador `long` nos tipos `int` e `double`, somente:

Tipo	Tamanho	Aplicação	Intervalo
<code>long int</code>	4 bytes	Armazena números inteiros: (conjunto \mathbb{Z})	-2147483647 a $+2147483647$
<code>long long int</code>	8 bytes	Armazena números inteiros longos: (conjunto \mathbb{Z})	-9223372036854775807 a $+9223372036854775807$
<code>long double</code>	10 bytes	Armazena números reais: (conjunto \mathbb{R})	<i>Dependente da arquitetura</i>

Modificador `long`

- Para obter um tamanho de variável maior que os tamanhos padrões, utiliza-se o modificador `long` nos tipos `int` e `double`, somente:

Tipo	Tamanho	Aplicação	Intervalo
<code>long int</code>	4 bytes	Armazena números inteiros: (conjunto \mathbb{Z})	-2147483647 a $+2147483647$
<code>long long int</code>	8 bytes	Armazena números inteiros longos: (conjunto \mathbb{Z})	-9223372036854775807 a $+9223372036854775807$
<code>long double</code>	10 bytes	Armazena números reais: (conjunto \mathbb{R})	<i>Dependente da arquitetura</i>

- Não existe `long float`. Não é possível colocar mais que 2 `long` no tipo `int`.
- Atualmente, na maioria das arquiteturas os tipos `int` e `long int` são idênticos. Porém, no passado, o tipo `int` possui 2 bytes de tamanho: -32768 a $+32767$

Segundo programa em C: análise

```
1.  #include <stdio.h>
2.
3.  int main() // Função Principal
4.  {
5.      // Variáveis
6.      int a = 2;
7.      int b = 3;
8.      int soma = a + b;
9.
10.     printf("Resultado da soma de A e B = %d\n", soma);
11.
12.     return 0;
13. } // Fim
```

Instrução printf: imprime o resultado na janela de comando.

Imprimindo valores de variáveis

- ▶ `printf("Resultado da soma de A e B = %d\n", soma);`
- ▶ Para que a função `printf` seja capaz de imprimir o valor de uma variável, deve-se indicar o **especificador de saída** (também chamado de **máscara**) no **local** em que a impressão deve ser realizada.
- ▶ Especificadores de saída sempre iniciam com o símbolo `%`.

Tipo	Especificador	Tipo	Especificador
int	%d	long int	%ld
float	%f	long long int	%lld
double	%lf	long double	%Lf

Imprimindo valores de variáveis

▶ `printf("Resultado da soma de A e B = %d\n", soma);`

Indica que o valor de uma variável do tipo `int` será impresso aqui.

Tipo	Especificador	Tipo	Especificador
<code>int</code>	<code>%d</code>	<code>long int</code>	<code>%ld</code>
<code>float</code>	<code>%f</code>	<code>long long int</code>	<code>%lld</code>
<code>double</code>	<code>%lf</code>	<code>long double</code>	<code>%Lf</code>

Imprimindo valores de variáveis

▶ `printf("Resultado da soma de A e B = %d\n", soma);`

Indica que o valor de uma variável do tipo `int` será impresso aqui.

Após a vírgula, indica-se *qual* variável deve ser impressa no lugar do `%d`: no caso, `soma`

Tipo	Especificador	Tipo	Especificador
<code>int</code>	<code>%d</code>	<code>long int</code>	<code>%ld</code>
<code>float</code>	<code>%f</code>	<code>long long int</code>	<code>%lld</code>
<code>double</code>	<code>%lf</code>	<code>long double</code>	<code>%Lf</code>

Imprimindo valores de variáveis

- ▶ É comum que a função `printf` imprima mais de uma variável.
- ▶ Para cada variável, deve haver um especificador:

Imprimindo valores de variáveis

- ▶ É comum que a função `printf` imprima mais de uma variável.
- ▶ Para cada variável, deve haver um especificador:

```
1. #include <stdio.h>
2.
3. int main() // Função principal
4. {
5.     // Variáveis
6.     int x = 5;
7.     float p = 2.718;
8.
9.     printf("Valores: X = %d e P = %f \n", x, p);
10.
11.     return 0;
12. } // Fim
```

Imprimindo valores de variáveis

- ▶ É comum que a função `printf` imprima mais de uma variável.
- ▶ Para cada variável, deve haver um especificador:

```
1. #include <stdio.h>
2.
3. int main() // Função principal
4. {
5.     // Variáveis
6.     int x = 5;
7.     float p = 2.718;
8.
9.     printf("Valores: X = %d e P = %f \n", x, p);
10.
11.     return 0;
12. } // Fim
```

*O valor de p será impresso
no lugar do %f*

*O valor de x será impresso
no lugar do %d*

Imprimindo valores de variáveis

- ▶ É comum que a função `printf` imprima mais de uma variável.
- ▶ Para cada variável, deve haver um especificador:

```
1. #include <stdio.h>
2.
3. int main() // Função principal
4. {
5.     // Variáveis
6.     int x = 5;
7.     float p = 2.718;
8.
9.     printf("Valores: X = %d e P = %f \n", x, p);
10.
11.     return 0;
12. } // Fim
```



As variáveis são separadas por vírgula.

Comando de entrada

- ▶ A função `scanf` é uma das funções de entrada da linguagem C. Ela permite receber dados através do teclado.
- ▶ `scanf` está definida no arquivo de cabeçalho `stdio.h`.
- ▶ Para que seja possível utilizar a função `scanf`, deve-se incluir este arquivo de cabeçalho antes da função `main`.
- ▶ Ou seja, o código deve conter o comando `#include <stdio.h>`.
- ▶ Do estudo de pseudocódigos, enquanto que o comando `printf` traduz-se por uma instrução **MOSTRAR**, o comando `scanf` é o equivalente à uma instrução **LER**.

Terceiro programa em C

- ▶ Lendo dados numéricos do teclado e somando:

terceiro.c

```
1.  #include <stdio.h>
2.
3.  int main() // Função Principal
4.  {
5.      // Variáveis
6.      int a, b, resp;
7.
8.      printf("Digite um valor para A: ");
9.      scanf("%d", &a);
10.
11.     printf("Digite um valor para B: ");
12.     scanf("%d", &b);
13.
```

```
14.     resp = a + b;
15.
16.     printf("Soma de A e B = %d\n", resp);
17.
18.     return 0;
19. } // Fim
```



Terceiro programa em C

- ▶ Lendo dados numéricos do teclado e somando:

terceiro.c

```
1.  #include <stdio.h>
2.
3.  int main() // Função Principal
4.  {
5.      // Variáveis
6.      int a, b, resp;
7.
8.      printf("Digite um valor para A: ");
9.      scanf("%d", &a);
10.
11.     printf("Digite um valor para B: ");
12.     scanf("%d", &b);
13.
```

Declaração de 3 variáveis inteiras

Mensagem para o usuário

Lê dados do teclado: %d para int

```
14.     resp = a + b;
15.
16.     printf("Soma de A e B = %d\n", resp);
17.
18.     return 0;
19. } // Fim
```

Realiza a adição

Imprime a resposta na tela

Comando de entrada

- ▶ `scanf("%d", &a);`
- ▶ Composta por duas partes: `scanf("Expressão", Argumentos)`
 - ▶ **Expressão**: contém os especificadores (%) dos tipos de variáveis lidas:

Tipo	Especificador	Tipo	Especificador
int	%d	long int	%ld
float	%f	long long int	%lld
double	%lf	long double	%Lf

- ▶ **Argumentos**: contêm os nomes das variáveis a serem lidas, **SEMPRE PRECEDIDOS PELO SÍMBOLO &**.

Exemplos

- ▶ **Exemplo 1:** criar um programa que leia um inteiro e um real e mostre o seu produto.
- ▶ **Exemplo 2:** criar um programa que leia um inteiro e o eleve ao cubo.
- ▶ **Exemplo 3:** criar um programa que leia uma temperatura em graus Fahrenheit e a converta para graus Celsius. A fórmula é: $^{\circ}\text{C} = (^{\circ}\text{F} - 32) / 1,8$
- ▶ **Exemplo 4:** sabendo que uma loja está fazendo uma promoção e dando desconto de 5% em todos os itens vendidos, criar um programa que leia o preço de um produto e mostre o seu novo valor, com desconto.

Operador de atribuição

- ▶ A **atribuição** é uma das operações mais utilizadas em qualquer linguagem de programação estruturada.
- ▶ Ela é responsável por armazenar um valor em uma variável.
- ▶ Expressa em linguagem C pelo símbolo = (igual).

Operador de atribuição

- ▶ Sintaxe:

`nome_da_variável = expressão;`

- ▶ *Expressão*: qualquer combinação aritmética de valores, variáveis, constantes ou funções cuja resposta tenha o mesmo tipo da variável definida por `nome_da_variável`.

Operador de atribuição

- ▶ Sintaxe:

`nome_da_variável = expressão;`

- ▶ *Expressão*: qualquer combinação aritmética de valores, variáveis, constantes ou funções cuja resposta tenha o mesmo tipo da variável definida por `nome_da_variável`.
- ▶ O operador de atribuição calcula o resultado da expressão que está à direita do operador `=` e armazena esse valor na variável à esquerda do operador, nunca o contrário.

Operadores aritméticos

- Os operadores aritméticos operam sobre números (**valores**, **variáveis**, **constantes**, **funções**) e têm como resultado valores numéricos.

Operação	Símbolo utilizado	Exemplo
Adição	+	<code>resp = a + b;</code>
Subtração	-	<code>resp = a - b;</code>
Multiplicação	*	<code>resp = a * b;</code>
Divisão	/	<code>resp = a / b;</code>
Resto de divisão	%	<code>resp = a % b;</code>

- Adição e subtração têm a mesma prioridade, que é menor do que a prioridade de multiplicação, divisão e resto de divisão.

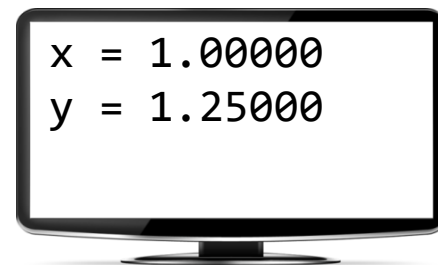
Operadores aritméticos

- ▶ Em uma operação de divisão, se o numerador e o denominador forem números inteiros, o resultado conterá somente a parte inteira da divisão.

Operadores aritméticos

- ▶ Em uma operação de divisão, se o numerador e o denominador forem números inteiros, o resultado conterá somente a parte inteira da divisão.

```
1. #include <stdio.h>
2. int main()
3. {
4.     float x, y;
5.
6.     x = 5 / 4;
7.     printf("x = %f \n", x);
8.
9.     y = 5 / 4.0;
10.    printf("y = %f \n", y);
11.    return 0;
12. }
```

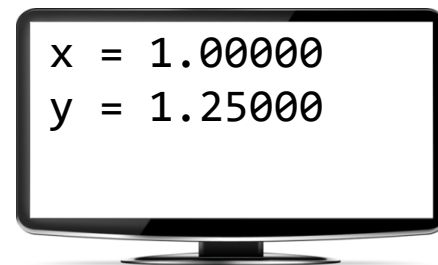


Operadores aritméticos

- ▶ Em uma operação de divisão, se o numerador e o denominador forem números inteiros, o resultado conterá somente a parte inteira da divisão.

```
1. #include <stdio.h>
2. int main()
3. {
4.     float x, y;
5.
6.     x = 5 / 4;
7.     printf("x = %f \n", x);
8.
9.     y = 5 / 4.0;
10.    printf("y = %f \n", y);
11.    return 0;
12. }
```

int / int = int
float / int = float
int / float = float
idem para double

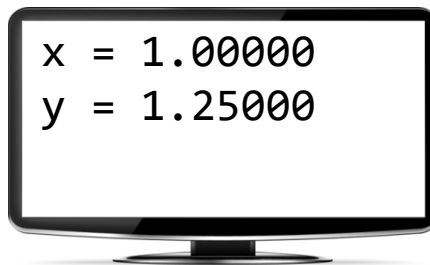


Operadores aritméticos

- ▶ Em uma operação de divisão, se o numerador e o denominador forem números inteiros, o resultado conterá somente a parte inteira da divisão.

```
1. #include <stdio.h>
2. int main()
3. {
4.     float x, y;
5.
6.     x = 5 / 4;
7.     printf("x = %f \n", x);
8.
9.     y = 5 / 4.0;
10.    printf("y = %f \n", y);
11.    return 0;
12. }
```

Separador de casas decimais é o **ponto**, não a vírgula!



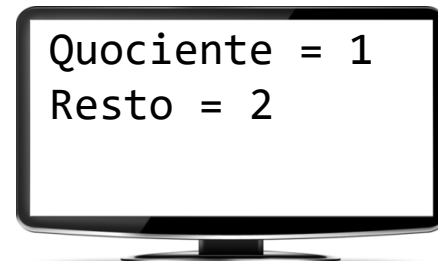
Operadores aritméticos

- ▶ O operador % (resto de divisão) só é usado com **números inteiros**.

Operadores aritméticos

- ▶ O operador `%` (resto de divisão) só é usado com **números inteiros**.

```
1. #include <stdio.h>
2. int main()
3. {
4.     int x = 5, y = 3;
5.     int quoc, resto;
6.
7.     quoc = x / y;
8.     resto = x % y;
9.
10.    printf("Quociente = %d\nResto = %d\n", quoc, resto);
11.
12.    return 0;
13. }
```



Funções matemáticas

- ▶ A linguagem C traz diversas funções que realizam cálculos matemáticos, como potenciação, radiciação, logaritmos, seno, etc...
- ▶ O arquivo de cabeçalho onde estas funções estão definidas é chamado `math.h`.
- ▶ Assim, o código deve conter o comando `#include <math.h>` antes da função `main`.
- ▶ Para realizar operações aritméticas, não é necessário realizar esta inclusão.

Funções matemáticas

- ▶ Potenciação: `pow(b, e)`
 - ▶ Eleva o argumento `b` (base) ao expoente `e`.
 - ▶ Tanto o resultado quanto os parâmetros `b` e `e` são tratados como tendo o tipo `double`.
 - ▶ O resultado possui o tipo `double`.
- ▶ Raiz quadrada: `sqrt(n)`
 - ▶ Extrai a raiz quadrada de `n`. O parâmetro `n` é tratado como tendo o tipo `double`.
 - ▶ O resultado possui o tipo `double`.
- ▶ Seno, cosseno e tangente: `sin(a)`, `cos(a)`, `tan(a)`
 - ▶ Calculam, respectivamente, o seno, cosseno e a tangente do ângulo `a`.
 - ▶ O ângulo deve ser informado em `radianos`.
 - ▶ O parâmetro `a` é tratado como tendo o tipo `double`.
 - ▶ O resultado possui o tipo `double`.

Funções matemáticas

- ▶ Exponencial e logaritmos: `exp(n)`, `log(n)`, `log10(n)`
 - ▶ Calculam, respectivamente, a exponencial (base e), o logaritmo natural (base e), e o logaritmo (base 10) de n . O parâmetro n é tratado como tendo o tipo `double`.
 - ▶ O resultado possui o tipo `double`.
- ▶ Valor absoluto (módulo): `fabs(n)`
 - ▶ Retorna o valor absoluto de n . O parâmetro n é tratado como tendo o tipo `double`.
 - ▶ O resultado possui o tipo `double`.
- ▶ Valor das constantes π (pi) e e (base dos logaritmos naturais):
 - ▶ Pode-se usar as constantes presentes na linguagem: `M_PI` e `M_E`
 - ▶ Ou obter o valor de π através de: `acos(-1)`, isto é, o arco cujo cosseno é -1 .
- ▶ Outras funções: consulte <https://cplusplus.com/reference/cmath/>

Funções matemáticas: compilação do código

- ▶ Sempre quando o cabeçalho `math.h` for adicionado no código, é necessário adicionar um parâmetro a mais na linha de comando, para realizar a compilação do programa.
- ▶ Trata-se do parâmetro `-lm` (menos L e M minúsculos).
- ▶ Dessa forma, a compilação e a execução são realizadas da seguinte forma:
- ▶ `gcc -o saida arquivo_fonte.c -lm` (compila)
- ▶ `./saida` (executa)

Funções matemáticas

- ▶ **Exemplo 5:** ler as medidas dos catetos de um triângulo retângulo e calcular o valor da hipotenusa.

Dúvidas?



Aula 3:

Variáveis, Entrada de Dados e Operadores

Disciplina: Fundamentos de Programação

Prof. Luiz Olmes

olmes@unifei.edu.br

