Aula: Introdução a C

Introdução a Programação

Túlio Toffolo & Puca Huachi
http://www.toffolo.com.br

Departamento de Computação
Universidade Federal de Ouro Preto

Aula: Introdução a C

1. Variáveis
2. Operadores aritméticos
3. Comandos de entrada e saída
4. Exemplos e exercícios

Aula: Introdução a C

Variáveis

Não é possível fazer programas de computador úteis sem utilizar alguma porção de memória.

Variáveis:

- são locações na memória do computador onde pode-se armazenar um valor;
- são utilizadas para armazenar e manipular dados.

Toda variável tem:

- **tipo**,
- **nome**,
- **endereço de memória**,
- **valor**.

Variáveis

- **Tipos fundamentais**
  - `int` – armazena um número inteiro, Jorge e Lucas!!!
  - `double` – especifica os números reais; 3.4, -0.985, etc.
  - `char` – armazena um único caractere minúsculo ou maiúsculo, um dígito, ou um caractere especial ($ * @).
- Em C, os tipos fundamentais são **palavras reservadas** escritas em **letras minúsculas**.

Variáveis

- As variáveis podem ocupar tamanhos diferentes na memória, dependo do tipo, exemplo:

Tipo	***Bytes***	**Intervalo**
`char`	1	0 a 255
`short`	2	-32.768 a 32.767
`int`	4	$-2.147.483.648$ a $2.147.483.647$
`long`	4	$-2.147.483.648$ a $2.147.483.647$
`float`	4	$1,2 \times 10^{-38}$ a $3,4 \times 10^{+38}$
`double`	8	$2,2 \times 10^{-308}$ a $1,8 \times 10^{+308}$

Variáveis

- **Identificador**
  - é o nome da variável, e **não pode ser uma palavra-chave**.
  - é formado por uma combinação de letras, dígitos e “ _ ” sublinhado (*underline*), **começando sempre com uma letra** ou “ _ ”.
  - **case sensitive**: letras maiúsculas e minúsculas **são diferentes**.
  - para assegurar a portabilidade use no máximo **31 caracteres**.
  - escolha **nomes significativos** para facilitar a documentação e o entendimento do código.

Declaração de Variáveis

- Criar (declarar) uma variável em C é muito fácil!
- **Declaração de variáveis**:
  - em um programa C, uma variável envolve um tipo e um identificador:
    `tipo identificador;`
- Exemplo: `int number;`
  - O tipo `int` especifica que o valor armazenado é do tipo inteiro (valor inteiro).
  - O identificador `number` é o nome da variável.
- Pode-se declarar várias variáveis em uma mesma linha:
  - `int number1, number2, number3, number4;`

Variáveis

Alguns erros...

```c
int var1, 2var, _var3;
```

```c
int int;
```

```c
int x, y, z;
int double;
```

Porque os códigos acima geram erros?

Declaração de Variáveis

- **Onde declarar?**
  - Variáveis podem ser declaradas em qualquer lugar de um programa C/C++, mas **devem aparecer antes** de serem usadas no programa.

Exemplo 1	**Exemplo 2**
`int x;`	`int x;`
`x = 80;`	`int y;`
`printf("%d", x);`	`x = 80;`
`int y;`	`y = 60;`
`y = 60;`	`printf("%d", x);`
`printf("%d", y);`	`printf("%d", y);`

Memória

Endereço	Valor
00010000	??
00010001	??
00010002	??
00010003	??
00010004	??
00010005	??
00010006	??
00010007	??
00010008	??
00010009	??
0001000A	??
0001000B	??
0001000C	??
0001000D	??

- A memória é formada por várias células.
- Cada célula contém um endereço e um valor (veja exemplo ao lado).
- O tamanho do endereço e do valor dependem da arquitetura (32/64 bits).

Memória

Endereço	Valor
00010000	??
00010001	??
00010002	??
00010003	??
00010004	??
00010005	??
00010006	??
00010007	??
00010008	??
00010009	??
0001000A	??
0001000B	??
0001000C	??
0001000D	??

i

Exemplo:

- O caractere `char i` ocupa 1 byte na memória

```
1  int main()
2  {
3      char i;
4      return 0;
5  }
```

Memória

Endereço	Valor
00010000	??
00010001	
00010002	
00010003	
00010004	??
00010005	??
00010006	??
00010007	??
00010008	??
00010009	??
0001000A	??
0001000B	??
0001000C	??
0001000D	??

i

Exemplo:

- O inteiro `int i` ocupa 4 bytes na memória (considerando uma arquitetura de 32 bits)

```
1  int main()
2  {
3      int i;
4      return 0;
5  }
```

Memória

Endereço	Valor
00010000	??
00010001	
00010002	
00010003	
00010004	??
00010005	??
00010006	??
00010007	??
00010008	??
00010009	??
0001000A	??
0001000B	??
0001000C	??
0001000D	??

i

Exemplo:

- O ponto flutuante `float i` ocupa 4 bytes na memória (considerando uma arquitetura de 32 bits)

```
1  int main()
2  {
3      float i;
4      return 0;
5  }
```

Memória

Endereço	Valor
00010000	??
00010001	
00010002	
00010003	
00010004	
00010005	
00010006	
00010007	
00010008	??
00010009	??
0001000A	??
0001000B	??
0001000C	??
0001000D	??

i

Exemplo:

- `double i` ocupa 8 bytes na memória (considerando uma arquitetura de 32 bits)

```
1  int main()
2  {
3      double i;
4      return 0;
5  }
```

Endereços

- Ao declararmos uma variável `x`, ela será associada a:
  - Um nome (exemplo: `x`)
  - Um endereço de memória ou referência (exemplo: 0xbfd267c4)
  - Um valor (exemplo: 9)

```
1  int x = 9;
```

- Para acessar o endereço de uma variável, utilizamos o operador **&**

Aula: Introdução a C

Operador de atribuição

- `sum = number1 + number2;`
  - O símbolo '=' é um operador de atribuição.
  - Avalia-se a expressão matemática do lado direito do '=' e atribui-se o resultado à variável do lado esquerdo.
  - = e + são operadores binários; requerem dois operandos.

- **Dica**: coloque espaços em branco em ambos os lados de um operador binário para facilitar a leitura do programa.

Operadores aritméticos

Operação	Operador aritmético	Exemplo	Exemplo em C/C++
Adição	`+`	$f + 7$	`f + 7`
Subtração	`-`	$p - c$	`p - c`
Multiplicação	`*`	bm ou $b \times m$	`b * m`
Divisão	`/`	x/y ou $x \div y$ ou $\frac{x}{y}$	`x / y`
Módulo	`%`	$r \bmod s$	`r % s`

Observações:

- Operador módulo `%`: resulta no resto da divisão inteira (somente usado com operandos inteiros)
- Exemplo: `7 % 4` é igual a 3

Operadores aritméticos

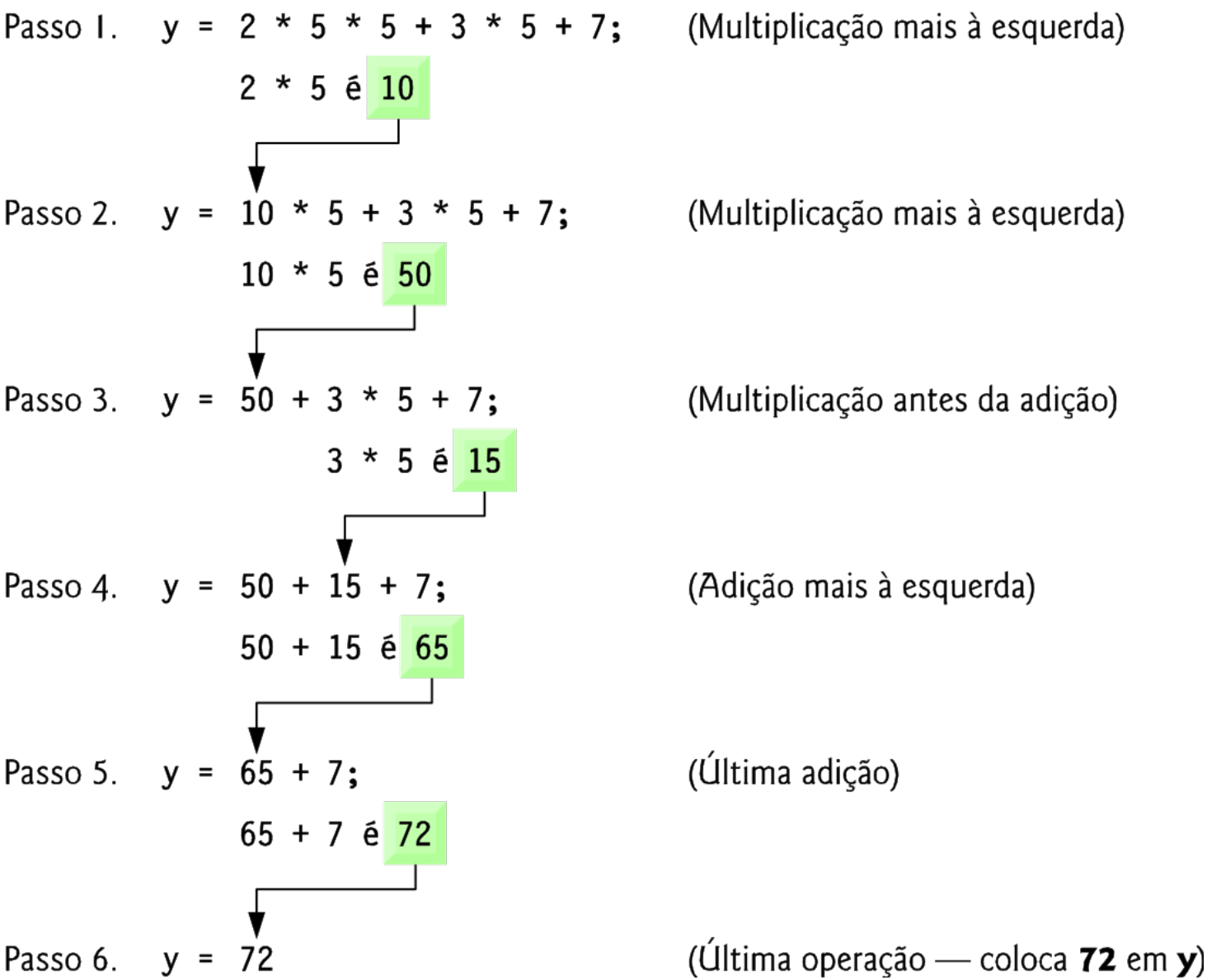
Regras da Precedência de Operadores

- São as mesma da álgebra:
  1. Operadores entre parênteses são avaliados primeiro; note que o parênteses quebra a precedência de um operador.
  2. A seguir, aplicam-se as operações de **multiplicação**, **divisão** e **módulo**. Se uma expressão contém vários desses operadores, as operações são aplicadas da esquerda para a direita.
  3. Por último aplicam-se a **adição** e a **subtração**. Se há vários + e −, a aplicação ocorre da esquerda para a direita.

Operadores aritméticos

Regras da Precedência de Operadores

Operação	Operador	Ordem de avaliação
Parênteses	( )	Avaliados primeiro (pares mais internos avaliados antes)
Multiplicação Divisão Módulo	* / %	Avaliados em segundo lugar. Se houver vários, avaliação da esquerda para direita.
Adição Subtração	+ -	Avaliados por último. Se houver vários, avaliação da esquerda para direita.


Passo 1. y = 2 * 5 * 5 + 3 * 5 + 7; (Multiplicação mais à esquerda)
2 * 5 é 10
Passo 2. y = 10 * 5 + 3 * 5 + 7; (Multiplicação mais à esquerda)
10 * 5 é 50
Passo 3. y = 50 + 3 * 5 + 7; (Multiplicação antes da adição)
3 * 5 é 15
Passo 4. y = 50 + 15 + 7; (Adição mais à esquerda)
50 + 15 é 65
Passo 5. y = 65 + 7; (Última adição)
65 + 7 é 72
Passo 6. y = 72 (Última operação — coloca 72 em y)

Aula: Introdução a C

A função `printf`

A função `printf` é parte da biblioteca `<stdio.h>`:

- Utilizada para imprimir na tela.
- Exemplo de uso:

```
1  printf("Olá mundo!!!\n");
```

A função `printf`

Mas.. Como imprimir um número inteiro?

Erros comuns

```c
printf(10);
```

```c
int valor = 10;
printf(valor);
```

Os códigos acima produzirão um **erro**, pois `printf` deve receber um texto/formato (entre aspas), não um inteiro (seja valor ou variável).

A função `printf`

Uso de `printf`:

- `printf(formato, valor/variável);`

Exemplo:

```
1  printf("%d", 10);
```

(note que `"%d"` é usado para números inteiros)

A função `printf`

Alguns possíveis formatos para o comando `printf`:

- `"%d"`: `int` (número inteiro)
- `"%ld"`: `long long` (número inteiro)
- `"%f"`: `float` (ponto flutuante)
- `"%lf"`: `double` (ponto flutuante)
- `"%c"`: `char` (caractere)
- `"%s"`: `string` (cadeia de caracteres)

A função `printf`

Assim, para imprimir um número inteiro usamos o formato `"%d"` como texto e indicamos o inteiro como próximo argumento.

- Exemplos:

```c
printf("%d", 100); // imprime o número inteiro 100
```

```c
int number = 10;
printf("%d", number); // imprime o valor da variável number
```

```c
int n1 = 10;
int n2 = 20;
int soma = n1 + n2;
printf("%d\n", soma); // imprime o valor de soma e a quebra de linha
```

A função `printf`

Note que é possível mesclar formato com texto, como por exemplo em `"O resultado é %d"`

```
1  int n1 = 10;
2  int n2 = 20;
3  int soma = n1 + n2;
4  printf("A soma de %d e %d é igual a %d.\n", n1, n2, soma);
```

Naturalmente, o código acima produzirá a saída:

```
1  A soma de 10 e 20 é igual a 30.
```

A função `printf`

Outro exemplo:

```
1  double n1 = 10.8;
2  double n2 = 19.3;
3  double soma = n1 + n2;
4  printf("%lf + %lf = %lf\n", n1, n2, soma);
```

O código acima produzirá a saída:

```
1  10.8 + 19.3 = 30.1
```

Note que o caractere ponto (.) é usado para separar os decimais,

Formatando a saída

A função `printf` permite formatar a saída de dados. O usuário pode especificar, entre outros:

- número de casas decimais;
- número de caracteres ocupados pela impressão.

Exemplos:

- `"%3d"`: um `int` usando no mínimo 3 espaços
- `"%-3d"`: um `int` usando no mínimo 3 espaços (alinhado à esquerda)
- `"%5s"`: uma `string` usando no mínimo 5 espaços
- `"%.3f"`: um `float` usando 3 casas decimais
- `"%3f"`: um `float` usando no mínimo 3 espaços
- `"%5.3f"`: um `float` usando 3 casas decimais e no mínimo 5 espaços

Formatando a saída

Exemplo:

```
1  printf("%-3s %8s\n", "Var", "Val");
2  printf("%-3s %8.1f\n", "x", 10.222);
3  printf("%-3s %8.1f\n", "y", 20.33);
4  printf("%-3s %8.1f\n", "z", 30);
```

Imprimirá na saída:

```
1  Var       Val
2  x        10.2
3  y        20.3
4  z        30.0
```

Formatando a saída

Caracteres especiais:

- `\n`: quebra de linha, ou seja, passa para a linha debaixo;
- `\t`: tabulação horizontal, equivalente a um **tab**;
- `\"`: aspas duplas;
- `\'`: aspas simples ou apóstrofo;
- `\\`: barra invertida
- `\a`: ???**beep** ;)

Formatando a saída

Exemplo de impressão de tabela:

```
1  printf("Var \t Val\n");
2  printf("x \t 10\n");
3  printf("y \t 20\n");
4  printf("z \t 30\n");
```

Resultado:

```
1  Var      Val
2  x        10
3  y        20
4  z        30
```

A função `scanf`

A função `scanf` também é parte da biblioteca `<stdio.h>`:

- Utilizada para ler da entrada padrão (*terminal*).
- O `scanf` tem algumas (grandes) diferenças em relação ao `printf`:
  - A função `printf` imprime texto e o **valor** de variáveis.
  - A função `scanf` **altera o conteúdo** das variáveis.
  - Alterar conteúdo equivale a **modificar o que está na memória**.
  - Por esta razão, sempre passamos um **endereço de memória** para a função `scanf`.

A função `scanf`

Uso de `scanf`:

- `scanf(formato, endereços de memória);`

Obter o endereço de memória de uma variável é fácil:

- utilizamos o operador `&`.

Exemplo:

```
1  int x;
2  scanf("%d", &x); // &x retorna o endereço de memória de x
```

("%d" é usado para números inteiros)

A função `scanf`

A função `scanf` usa os mesmos “formatos” que `printf`.

Exemplos:

- `"%d"`: `int` (número inteiro)
- `"%ld"`: `long long` (número inteiro)
- `"%f"`: `float` (ponto flutuante)
- `"%lf"`: `double` (ponto flutuante)
- `"%c"`: `char` (caractere)
- `"%s"`: `string` (cadeia de caracteres)

A função `scanf`

Porquê os códigos abaixo geram **erros**?

Erros comuns

```
1 int x;
2 scanf(x);
```

```
1 double valor = 10.0;
2 scanf(valor);
```

1. `scanf` deve receber um texto/formato (entre aspas), não um `int` ou `double` (seja valor ou variável).
2. `scanf` deve receber um endereço de memória, e não um valor.

A função `scanf`

E os códigos a seguir? Também geram **erros**?

Erros comuns

```
1  int x;
2  scanf("%d", x);
```

```
1  double valor = 10.0;
2  scanf("%lf", valor);
```

Sim: `scanf` deve receber **endereços de memória**, não valores.

A função `scanf`

Assim, para ler da entrada padrão usamos um “formato” e indicamos o endereço de memória como próximo argumento.

- Exemplos:

```c
int x;
scanf("%d", &x); // lê um inteiro da entrada padrão
```

```c
char c;
scanf("%c", &c); // lê um caractere da entrada padrão
```

```c
int n1, n2, soma;
scanf("%d %d", &n1, &n2); // lê dois inteiros da entrada padrão
soma = n1 + n2;
printf("A soma de %d e %d eh igual a %d", n1, n2, soma);
```

Aula: Introdução a C

Exemplos

Exemplo 1

Elabore um programa em C que lê o valor das variáveis `x`, `y` e `z` do tipo `int`. Em seguida, calcule o resultado da expressão a seguir:

$$r = x^3 + y^2 + xyz$$

Imprima o resultado no formato do exemplo abaixo:

```
x = 10, y = 10, z = 10
r = 2100
```

Exercício 1

Elabore um programa que imprime o resto da divisão $100000/3$.

Exercício 2

Elabore um programa em C que lê o valor das variáveis `a`, `b`, `c`, `d` e `e` do tipo `float` e, em seguida, calcula o resultado de:

$$x = a^3 \times \left(\frac{b+c}{d} + e \right)$$

O resultado deve ser impresso no formato do exemplo abaixo:

```
1 a = 10, b = 10, c = 10, d = 10, e = 10
2 x = 12000
```

Exercício 3 (Opcional)

Elabore um programa que lê o valor de `x1`, `x2` e `x3` para calcular (e imprimir) o resultado da expressão $y = (x_1 + 3)^4 + (x_2 \times x_3)^3$

Perguntas?