

# MC-102 — Aula 03

## Escrita, Leitura e Operações Aritméticas

Instituto de Computação – Unicamp

2015

# Roteiro

- 1 Saída de dados: `printf`
- 2 Entrada de dados: `scanf`
- 3 Expressões e Operadores Aritméticos
- 4 Operadores `++` e `--`
- 5 Exercícios
- 6 Outras Informações

## Escrevendo na tela

- Para imprimir um texto, utilizamos o comando `printf`. O texto pode ser uma constante do tipo `string`.

### Exemplo

```
printf("Ola Pessoal!");
```

Saída: Ola Pessoal!

- No meio da constante `string` pode haver comandos especiais. O símbolo especial `\n` é responsável por pular uma linha na saída.

### Exemplo

```
printf("Ola Pessoal! \n Ola Pessoal");
```

Saída: Ola Pessoal!

Ola Pessoal

## Escrevendo o conteúdo de uma variável na tela

- Podemos imprimir, além de texto puro, o conteúdo de uma variável utilizando o comando `printf`. Para isso utilizamos símbolos especiais no texto, para representar que aquele trecho deve ser substituído por uma variável ou constante, e no final, passamos uma lista de variáveis ou constantes, separadas por vírgula.

### Exemplo

```
int a=10;
printf("A variável %s contém o valor %d", "a", a);
Saída: A variável a contém o valor 10
```

- Nesse caso, `%s` deve ser substituído por uma variável ou constante do tipo `string`, enquanto `%d` deve ser substituído por uma variável ou constante do tipo inteiro.

# Formatos inteiros

`%d` — Escreve um inteiro na tela.

## Exemplo

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

Saída: 10

## Exemplo

```
int a=12;
```

```
printf ("0 valor e %d", a);
```

Saída: 0 valor e 12

# Formatos inteiros

- A letra **d** pode ser substituída pelas letras **u** e **ld**, quando desejamos escrever variáveis do tipo `unsigned` ou `long`, respectivamente.

## Exemplo

```
printf ("%d", 4000000000);
```

Saída: -294967296.

Enquanto que

```
printf ("%ld", 4000000000);
```

Saída: 4000000000.

# Formatos ponto flutuante

`%f` — Escreve um ponto flutuante na tela.

## Exemplo

```
printf ("%f", 10.0);
```

Saída: 10.000000

# Formatos ponto flutuante

`%e` — Escreve um ponto flutuante na tela, em notação científica

## Exemplo

```
printf ("%e", 10.02545);
```

Saída: 1.002545e+01

# Formatos ponto flutuante

`%.<decimais>f` — Escreve um ponto flutuante na tela, com  
<decimais> casas decimais.

## Exemplo

```
printf("%.2f", 10.1111);
```

Saída: 10.11

# Formatos ponto flutuante

- O formato `%f` pode ser substituído por `%lf`, para escrever um `double` ao invés de um `float`.

## Exemplo

```
printf("%.2lf", 10.0);
```

Saída: 10.00

# Formato caracter

`%c` — Escreve um caracter.

## Exemplo

```
printf ("%c", 'A');
```

Saída: A

Note que `printf ("%c", 65)` também imprime a letra A. Por que?

# Formato string

`%s` — Escreve uma string

## Exemplo

```
printf ("%s", "Meu primeiro programa");
```

Saída: Meu primeiro programa

# A função scanf

- Realiza a leitura de dados a partir do teclado.
- Parâmetros:
  - ▶ Uma string, indicando os tipos das variáveis que serão lidas e o formato dessa leitura.
  - ▶ Uma lista de variáveis.
- Aguarda que o usuário digite um valor e atribui o valor digitado à variável.

## A função scanf

O programa abaixo é composto de quatro passos:

- 1 Cria uma variável `n`;
- 2 Escreve na tela Digite um número:
- 3 Lê o valor do número digitado
- 4 Imprime o valor do número digitado

```
#include <stdio.h>
int main(){
    int n;
    printf("Digite um número: ");
    scanf("%d",&n);
    printf("O valor digitado foi %d\n",n);
}
```

## Formatos de leitura de variável

Os formatos de leitura são muito semelhantes aos formatos de escrita utilizados pelo `printf`. A tabela a seguir mostra alguns formatos possíveis de leitura

Código	Função
<code>%c</code>	Lê um único caracter
<code>%s</code>	Lê uma série de caracteres
<code>%d</code>	Lê um número decimal
<code>%u</code>	Lê um decimal sem sinal
<code>%ld</code>	Lê um inteiro longo
<code>%f</code>	Lê um número em ponto flutuante
<code>%lf</code>	Lê um double

## A função scanf

O programa abaixo, lê um caracter, depois um número ponto flutuante e por fim um decimal. Por fim o programa imprime os dados lidos.

```
#include <stdio.h>

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

    printf("Entre com um caracter:");
    scanf("%c", &c);
    printf("Entre com um ponto flutuante:");
    scanf("%f", &b);
    printf("Entre com um número:");
    scanf("%d",&a);

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

# Expressões

- Já vimos que constantes e variáveis são expressões.
- Uma expressão também pode ser um conjunto de operações aritméticas, lógicas ou relacionais utilizados para fazer “cálculos” sobre os valores das variáveis.

## Exemplo

$a + b$

Calcula a soma de **a** e **b**.

# Expressões Aritméticas

- Os operadores aritméticos são:  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $\%$
- $\langle \textit{expressao} \rangle + \langle \textit{expressao} \rangle$ : Calcula a soma de duas expressões.  
Ex:  $a + b$ ;
- $\langle \textit{expressao} \rangle - \langle \textit{expressao} \rangle$ : Calcula a subtração de duas expressões.  
Ex:  $a - b$ ;
- $\langle \textit{expressao} \rangle * \langle \textit{expressao} \rangle$ : Calcula o produto de duas expressões.  
Ex:  $a * b$ ;

# Expressões

- $\langle \textit{expressao} \rangle / \langle \textit{expressao} \rangle$ : Calcula a divisão de duas expressões.  
Ex:  $a / b$ ;
- $\langle \textit{expressao} \rangle \% \langle \textit{expressao} \rangle$ : Calcula o resto da divisão (inteira) de duas expressões.  
Ex:  $a \% b$ ;
- $- \langle \textit{expressao} \rangle$ : Inverte o sinal da expressão.  
Ex:  $-b$ ;

# Expressões

Mais sobre o operador resto da divisão: %

- Quando computamos "  $a$  dividido por  $b$ ", isto tem como resultado um valor  $p$  e um resto  $r < b$  que são únicos tais que

$$a = p * b + r$$

- Ou seja  $a$  pode ser dividido em  $p$  partes inteiras de tamanho  $b$ , e sobrar um resto  $r < b$ .

Exemplos:  $5\%2$  tem como resultado o valor 1.

$15\%3$  tem como resultado o valor 0.

$1\%5$  tem como resultado o valor 1.

$19\%4$  tem como resultado o valor 3.

# Expressões

No exemplo abaixo, quais valores serão impressos?

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

```
int main(){
```

```
    printf("%d \n", 27%3);
```

```
    printf("%d \n", 4%15);
```

```
}
```

# Expressões

## Mais sobre o operador /

- Quando utilizado sobre valores inteiros, o resultado da operação de divisão será inteiro. Isto significa que a parte fracionária da divisão será desconsiderada.
  - ▶  $5/2$  tem como resultado o valor 2.
- Quando pelo menos um dos operandos for ponto flutuante, então a divisão será fracionária. Ou seja, o resultado será a divisão exata dos valores.
  - ▶  $5.0/2$  tem como resultado o valor 2.5.

# Expressões

No exemplo abaixo, quais valores serão impressos?

```
#include <stdio.h>

int main(){
    int a=5, b=2;
    float c=5.0, d=2.0;

    printf("%d \n",a/b);
    printf("%f \n", a/d);
    printf("%f \n", c/d);
}
```

# Expressões

- As expressões aritméticas (e todas as expressões) operam sobre outras expressões.
- É possível compor expressões complexas como por exemplo:  
 $a = b + 2 + c + (9 + d * 8);$

Qual o valor da expressão **5 + 10 % 3**?

E da expressão **5 \* 10 % 3**?

# Precedência

- Precedência é a ordem na qual os operadores serão avaliados quando o programa for executado. Em C, os operadores são avaliados na seguinte ordem:
  - ▶ \* e /, na ordem em que aparecerem na expressão.
  - ▶ %
  - ▶ + e -, na ordem em que aparecerem na expressão.
- Exemplo:  $8+10*6$  é igual a 68.

## Alterando a precedência

- ( $< \textit{expressao} >$ ) também é uma expressão, que calcula o resultado da expressão dentro dos parênteses, para só então calcular o resultado das outras expressões.  
Ex:  $5 + 10 \% 3$  retorna 6, enquanto  $(5 + 10) \% 3$  retorna 0
- Você pode usar quantos parênteses desejar dentro de uma expressão, contanto que utilize o mesmo número de parênteses para abrir e fechar expressões.
- **OBS:** Use sempre parênteses em expressões para deixar claro em qual ordem a expressão é avaliada!

# Incremento(++ ) e Decremento(-- )

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

```
a = a + 1;
```

- Em C, o operador unário ++ é usado para incrementar de 1 o valor de uma variável.

```
a = a + 1; é o mesmo que a++;
```

- O operador unário -- é usado para decrementar de 1 o valor de uma variável.

```
a = a - 1; é o mesmo que a--;
```

# Incremento(++ ) e Decremento(-- )

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 **--**.

O programa abaixo imprime "b: 6".

```
#include <stdio.h>

int main(){
    int a=5, b, c;

    b = ++a;

    printf(" b: %d \n",b);
}
```

Já o programa abaixo imprime "b: 5".

```
#include <stdio.h>

int main(){
    int a=5, b, c;

    b = a++;

    printf(" b: %d \n",b);
}
```

# Exercício

- Crie um programa que lê dois números **double** e que computa e imprime a soma, a diferença, a multiplicação e divisão dos dois números.

## Outras Informações: Atribuições simplificadas

Uma expressão da forma

$$a = a + b$$

onde ocorre uma atribuição a uma das variáveis da expressão pode ser simplificada como

$$a += b$$

# Atribuições simplificadas

Comando	Exemplo	Corresponde a:
<code>+=</code>	<code>a += b</code>	<code>a = a + b;</code>
<code>-=</code>	<code>a -= b</code>	<code>a = a - b;</code>
<code>*=</code>	<code>a *= b;</code>	<code>a = a * b;</code>
<code>/=</code>	<code>a /= b;</code>	<code>a = a / b;</code>
<code>%=</code>	<code>a %= b;</code>	<code>a = a % b;</code>

# Outras Informações: Conversão de tipos

- É possível converter alguns tipos entre si.
- Existem duas formas de fazê-lo: implícita e explícita:
- Implícita
  - ▶ Capacidade (tamanho) do destino deve ser maior que a origem senão há perda de informação.  
Ex.: `int a; short b; a = b;`  
Ex: `float a; int b=10; a = b;`
- Explícita:
  - ▶ Explicitamente informa o tipo que o valor da variável ou expressão é convertida.  
Ex. `a = (int)( (float)b / (float)c );`
  - ▶ Não modifica o tipo “real” da variável, só o valor de uma expressão.  
Ex. `int a; (float)a=1.0;` ← Errado

## Um uso da conversão de tipos

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. A expressão  $10 / 3$  tem como valor 3.
- Se **um** dos dois argumentos for de ponto flutuante, acontece a divisão de ponto flutuante. A expressão  $1.5 / 3$  tem como valor 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:

### Exemplo

A expressão  $10 / (\text{float}) 3$  tem como valor 3.33333333

## Outras Informações: comentários

- O código fonte pode conter comentários direcionados unicamente ao programador. Estes comentários devem estar delimitados pelos símbolos `/*` e `*/`, e são ignorados pelo compilador.

### Exemplo

```
#include <stdio.h>

/* Este é o meu primeiro programa. */
//Isto também é um comentário
int main() {
    printf("Hello, world!\n");
}
```

- Comentários são úteis para descrever o algoritmo usado e para explicitar suposições não óbvias sobre a implementação.