

MC-102 — Aula 03

Escrita, Leitura e Operações Aritméticas

Eduardo C. Xavier

Instituto de Computação – Unicamp

2017

Roteiro

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

Escrevendo na tela

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

```
printf("Olá Pessoal!");
```

Saída: Olá Pessoal!

- No meio da constante **string** pode-se incluir caracteres de formatação especiais. O símbolo especial `\n` é responsável por pular uma linha na saída.

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

Saída: Olá Pessoal!
Olá 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.

```
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 **int**.

Formatos inteiros

`%d` — Escreve um inteiro na tela.

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

Saída: 10

```
int a=12;  
printf ("O valor e %d", a);
```

Saída: O 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.

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

Saída: -294967296

Enquanto que

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

Saída: 4000000000.

Formatos ponto flutuante

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

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

Saída: 10.000000

Formatos ponto flutuante

`%.Nf` — Escreve um ponto flutuante na tela, com N casas decimais.

```
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**.

```
double a = 10.0;  
printf ("%2lf", a);
```

Saída: 10.00

Formato caracter

%c — Escreve um caracter.

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

Saída: A

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

Formato **string**

%s — Escreve uma **string**

```
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);
}
```

Note que no **scanf**, cada variável para onde será lido um valor, deve ser precedida do caracter **&**.

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
%c	Lê um único caracter
%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 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 caractere:");
    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);
}
```

A função **scanf**

É possível ler várias variáveis em um mesmo comando **scanf**, basta especificar todos os formatos das variáveis a serem lidas e depois as variáveis separadas por virgula:

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

```
int main(){  
    int m, n, o;  
    printf("Digite três números: ");  
    scanf("%d %d %d",&m, &n, &o);  
    printf("O valores digitados foram %d %d %d\n", m, n, o);  
}
```


Expressões

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

$$a + b$$

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

Expressões Aritméticas

- Os operadores aritméticos são: $+$, $-$, $*$, $/$, $\%$
- $expressão + expressão$: Calcula a soma de duas expressões.
Ex: $10 + 15$;
- $expressão - expressão$: Calcula a subtração de duas expressões.
Ex: $5 - 7$;
- $expressão * expressão$: Calcula o produto de duas expressões.
Ex: $3 * 4$;

Expressões

- $expressão / expressão$: Calcula a divisão de duas expressões.
Ex: $4 / 2$;
- $expressão \% expressão$: Calcula o resto da divisão (inteira) de duas expressões.
Ex: $5 \% 2$;
- $- expressão$: Inverte o sinal da expressão.
Ex: -5 ;

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", 29%3);
```

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

```
}
```

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

- $(expressão)$ também é uma expressão, que calcula o resultado da expressão dentro dos parênteses, para só então calcular o resultado das outras expressões.
 - ▶ $5 + 10 \% 3$ é igual a 6
 - ▶ $(5 + 10) \% 3$ é igual a 0
- Você pode usar quantos parênteses desejar dentro de uma expressão.
- 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--;`

Exercício

- Crie um programa que:
 - ▶ Lê um caracter, pula uma linha e imprime o caracter lido.
 - ▶ Lê um inteiro, pula uma linha e imprime o inteiro lido.
 - ▶ Lê um número ponto flutuante, pula uma linha e imprime o número lido.

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.

Informações Extras: 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;  
  
    b = ++a;  
  
    printf("b: %d \n", b);  
    printf("a: %d \n", a);  
}
```

Já o programa abaixo imprime "b: 5".

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

```
int main(){  
    int a=5, b;  
  
    b = a++;  
  
    printf("b: %d \n", b);  
    printf("a: %d \n", a);  
}
```

Mas em ambos o valor de **a** no final é 6.

Informações Extras: Atribuições simplificadas

Uma expressão da forma

$$a = a + b;$$

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

$$a += b;$$

Informações Extras: Atribuições simplificadas

Comando	Exemplo	Corresponde 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;

Informações Extras: Conversão de tipos

- É possível converter alguns tipos entre si: de forma implícita ou explícita.
- Na implícita atribui-se diretamente um dado de um tipo para uma variável de outro tipo.

```
#include <stdio.h>

int main(){
    int a = 9;
    double b;

    b = a;
    printf("a: %d e b: %lf\n", a, b);

    b = 5.56;
    a = b;
    printf("a: %d e b: %lf\n", a, b);

    b = 4000000000.56;
    a = b;
    printf("a: %d e b: %lf\n", a, b);

    printf("Tamanho em bytes de um double: %ld\n", sizeof(double));
    printf("Tamanho em bytes de um int: %ld\n", sizeof(int));
}
```

- ▶ Notem que a capacidade (tamanho) do destino deve ser maior que a origem senão há perda de informação.

Informações Extras: Conversão de tipos

- Conversão Explícita:

- ▶ Explicitamente informa o tipo que o valor da variável ou expressão é convertida.

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

```
int main(){  
    double b;
```

```
    b = ((double) 5 / (double) 2) ;  
    printf("%lf\n", b);
```

```
    b = 5 / 2 ;  
    printf("%lf\n", b);  
}
```