PROGRAMAÇÃO COMPUTACIONAL PARA ENGENHARIA

COMANDOS

Maurício Moreira Neto[1]

[1] **Universidade Federal do Ceará**
Departamento de Computação

30 de janeiro de 2020

DEPARTAMENTO DE COMPUTAÇÃO

Sumário

DEPARTAMENTO DE COMPUTAÇÃO

Objetivos

- Aprender os comandos de Entrada e Saída de dados
- Aprender quais são os diversos operadores (lógicos, aritméticos, atribuição, ...)
- Aprender como utilizar os operadores na linguagem C
- Aprender a realizar conversão de tipos

DEPARTAMENTO DE COMPUTAÇÃO

Comandos de Saída

- `printf()`
  - *print formatted*
  - Comando que realiza a impressão dos dados do programa na tela

```
printf(“Ola Mundo”);
```

```
Ola Mundo
```

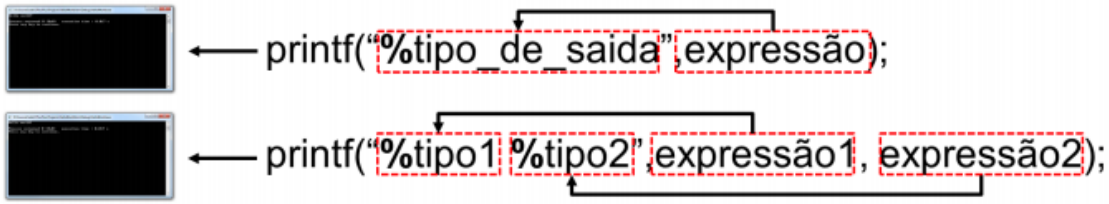
- O texto a ser escrito deve ser sempre definido entre “**aspas duplas**”

```
#include <stdio.h>
int main(){
printf(“Este texto deve aparecer na tela!”);
return 0;
}
```

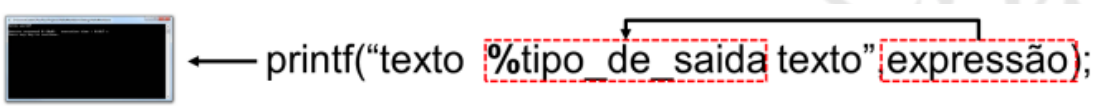
DEPARTAMENTO DE COMPUTAÇÃO

Comandos de Saída

- `printf()`
  - Quando queremos escrever dados formatados na tela usamos a forma geral da função, a qual possui os tipos de saída
  - Eles especificam o formato de saída de dados que serão escritos pela função `printf()`



- Podemos misturar o texto a ser mostrado com os especificadores de formato



DEPARTAMENTO DE COMPUTAÇÃO

Comandos de Saída

- `printf()`
  - Especificadores de formato

Alguns tipos de saídas	
%c	Escrita de um caractere (char)
%d ou %i	Escrita de números inteiros (int ou char)
%u	Escrita de números inteiros sem sinal (unsigned)
%f	Escrita de números reais (float ou double)
%s	Escrita de vários caracteres
%p	Escrita de um endereço de memória
%e ou %E	Escrita em notação científica

DEPARTAMENTO DE COMPUTAÇÃO

Comandos de Saída

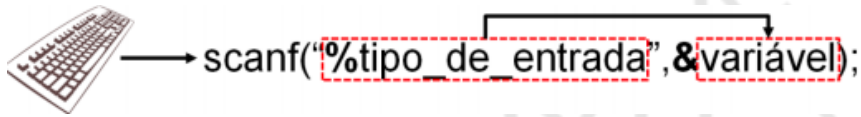
- Exemplo do uso de `printf()`:

```c
int main(){
    printf("Esse texto sera escrito na tela");

    int x = 10;
    float y = 20;
    printf("%d",x);

    printf("%d %f",x,y);

    printf("Valor de x eh %d e o de y eh %f",x,y);

    return 0;
}
```

DEPARTAMENTO DE COMPUTAÇÃO

Comandos de Entrada

- `scanf()`
  - Comando que realiza a leitura dos dados da entrada padrão (no caso o teclado)
  - `scanf("tipo de entrada", lista de variáveis);`



  - O tipo de entrada deve ser sempre definido entre "**aspas duplas**"
  - Na linguagem C, é necessário colocar o símbolo & antes do nome de cada variável a ser lida pelo comando `scanf()`
    - O símbolo & indica qual é o endereço da variável que vai receber os dados lidos

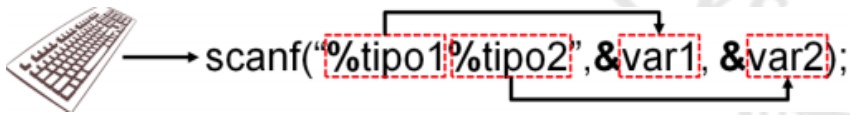
DEPARTAMENTO DE COMPUTAÇÃO

Comandos de Entrada

- `scanf()`
  - Especificadores de formato do tipo de entrada

Alguns tipos de saída	
%c	Leitura de um caractere (char)
%d ou %i	Leitura de números inteiros (int ou char)
%f	Leitura de números reais (float ou double)
%s	Leitura de vários caracteres

  - Podemos ler mais de um valor em um único comando
    - Quando digitar vários valores, separar com espaço, TAB ou ENTER



DEPARTAMENTO DE COMPUTAÇÃO

Comandos de Entrada

- Exemplo de uso do `scanf()`

```c
int main(){
    int x,z;
    float y;
    //Leitura de um valor inteiro
    scanf("%d",&x);
    //Leitura de um valor real
    scanf("%f",&y);
    //Leitura de um valor inteiro e outro real
    scanf("%d%f",&x,&y);
    //Leitura de dois valores inteiros
    scanf("%d%d",&x,&z);
    //Leitura de dois valores inteiros com espaço
    scanf("%d %d",&x,&z);

    return 0;
}
```

DEPARTAMENTO DE COMPUTAÇÃO

Comandos de Entrada

- `getchar()`
  - Comando que realiza a leitura de um único caractere

```c
int main(){
    char c;
    c = getchar();
    printf("Caractere: %c\n", c);
    printf("Codigo ASCII: %d\n", c);

    return 0;
}
```

DEPARTAMENTO DE COMPUTAÇÃO

Sequências de Escape

- São constantes predefinidas
- Elas permitem o envio de caracteres de controle não gráficos para dispositivos de saída

Códigos	Comandos
\a	Som de alerta (bip)
\b	Retrocesso (backspace)
\n	Nova linha (new line)
\r	Retorno de carro (carriage return)
\v	Tabulação vertical
\t	Tabulação horizontal
\'	Apóstrofe
\"	Aspas
\\	Barra invertida (backslash)
\f	Alimentação de folha (form feed)
\?	Símbolo de interrogação
\0	Caractere nulo (cancela a escrita do restante)

DEPARTAMENTO DE COMPUTAÇÃO

Sequências de Escape

Exemplo

```
#include <stdio.h>
#include <stdlib.h>
int main(){
printf(“Hello World\n”);
printf(“Hello\nWorld\n”);
printf(“Hello\ \World\n”);
printf(“\“Hello World\“ \n”);
return 0;
}
```

- Saída:

```
clang: warning: treating 'c' input as 'c++' when in C++ mode, this behavior is deprecated [-Wdeprecated]
Hello World
Hello
World
Hello \ World
" Hello World"
[Finished in 4.5s]
```

DEPARTAMENTO DE COMPUTAÇÃO

Operadores

- Os operadores são usados para desenvolver diferentes tipos de operações. Com eles podemos:
  - Realizar operações **matemáticas** com suas variáveis
  - Realizar operações de **comparação** entre suas variáveis
  - Realizar operações **lógicas** entre suas variáveis
  - Realizar operações em **nível de bits** com suas variáveis

DEPARTAMENTO DE COMPUTAÇÃO

Operadores Aritméticos

- São operadores que realizam operações em números (valores, variáveis, constantes e chamadas de funções) e/ou expressões e tem como resultado valores numéricos
  - Note que os operadores aritméticos são sempre usados em conjunto com o operador de atribuição

Operadores	Significados	Exemplo
+	Adição	z = x + y
-	Subtração	z = x - y
*	Multiplicação	z = x * y
/	Quociente	z = x / y
%	Resto da divisão	z = x % y

DEPARTAMENTO DE COMPUTAÇÃO

Operadores Aritméticos

- Podemos devolver o resultado para uma outra variável ou para um outro comando ou função que espere receber um valor do mesmo tipo do resultado da operação, no caso, a função `printf()`

```
int main(){
int x = 10, y = 15, z;
z = x * y;
printf("z = %d\n",z);
z = y / 10;
printf("z = %d\n", z);
printf("x + y = %d\n", x+y);
return 0;
}
```

DEPARTAMENTO DE COMPUTAÇÃO

Operadores Aritméticos

- **IMPORTANTE**
  - As operações de multiplicação, divisão e resto são executadas antes das operações de adição e subtração. Para forçar uma operação a ser executada antes das demais, ela deve ser colocadas entre parênteses
    - $z = x * y = 10;$
    - $z = x * (y + 10);$
  - O operador de subtração também pode ser utilizado para inverter o sinal de um número
    - $x = -y;$
  - Neste caso, a variável x receberá o valor de y multiplicado por -1, ou seja
    - $x = (-1) * y;$

DEPARTAMENTO DE COMPUTAÇÃO

Operadores Aritméticos

- **IMPORTANTE**
  - Em uma operação utilizando o operador de quociente /, se o numerador e o denominador forem números inteiros, por padrão o compilador retornará apenas a parte inteira da divisão

```
int main(){
float x;
x = 5/4; // x = 1.000000
printf("x = %f\n", x);

x = 5/4.0; // x = 1.250000
printf("x = %f\n", x);
return 0;
}
```

DEPARTAMENTO DE COMPUTAÇÃO

Operadores Relacionais

- São operadores que verificam a magnitude (maior ou menor) e/ou igualdade entre dois valores e/ou expressões
  - Os operadores relacionais são operadores de comparação de valores
  - Retornam **verdadeiro (1)** ou **falso (0)**

Operador	Significado	Exemplo
>	Maior que	X > 5
>=	Maior ou igual a	X >= Y
<	Menor	X < 5
<=	Menor ou igual a	X <= Z
==	Igual a	X == 0
!=	Diferente de	X ! = Y

DEPARTAMENTO DE COMPUTAÇÃO

Operadores Relacionais

- **IMPORTANTE**
  - O símbolo de atribuição = é diferente, muito diferente, do operador relacional de igualdade ==

```
int Nota;
Nota == 60; // Nota é igual a 60?
Nota = 50; // Nota recebe 50
/*
Erro comum em C
Teste se a nota é 60
Sempre entra na condição
*/
if (Nota = 60){
printf("Voce passou raspando!!");
}
// Versao correta!
if (Nota == 50){
printf("Voce passou raspando!!");
}
```

DEPARTAMENTO DE COMPUTAÇÃO

IMPORTANTE

- Símbolo de atribuição = é diferente do operador relacional de igualdade ==
- Por que sempre entra na condição?

```
if (Nota = 60) {
printf(“Você passou de semestre!”);
}
```

- Ao fazer **Nota = 60** (“Nota recebe 60”) estamos atribuindo um valor inteiro à variável Nota
- O valor atribuído **60 é diferente de Zero**. Como em C os booleanos são números inteiros, então vendo Nota como booleano, essa assume **true**, uma vez que é diferente de zero

DEPARTAMENTO DE COMPUTAÇÃO

Operadores Lógicos

- Certas situações não podem ser modeladas utilizando apenas operadores aritméticos e/ou relacionais
  - Um exemplo bastante simples disse é saber se determinada variável x esta dentro de uma faixa de valores
  - Por exemplo:
    - $0 < x < 10$
  - Indica que o valor de x deve ser maior do que 0 (zero) e também menor do que 10

DEPARTAMENTO DE COMPUTAÇÃO

Operadores Lógicos

- Os operadores lógicos permitem representar situações logicas unindo duas ou mais expressões relacionais simples em uma composta
  - Retorna **verdadeiro (1)** ou **falso (0)**
- **Exemplo**
  - A expressão $0 < x < 10$
  - Equivale a $(x > 0)\&\&(x < 10)$

Operador	Significado	Exemplo
&&	Operador **E**	(x > 0) && (x < 10)
||	Operador **OU**	(a == 'F') || (b != 32)
!	Operador **NEGAÇÃO**	!(x == 10)

DEPARTAMENTO DE COMPUTAÇÃO

Operadores Lógicos

- Tabela Verdade
  - Os termos **a** e **b** representam o resultado de duas expressões relacionais

a	b	!a	!b	a && b	a || b
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

DEPARTAMENTO DE COMPUTAÇÃO

Operadores Lógicos

```c
int main(){
int r, x = 5, y = 3;
r = (x > 2) && (y < x); // verdadeiro (1)
printf("Resultado = %d\n", r);
r = (x%2 == 0) && (y > 0); // falso (0)
printf("Resultado = %d\n", r);
r = (x > 0) || (y < 0); // verdadeiro (1)
printf("Resultado = %d\n", r);
r = (x%2 == 0) || (y < 0); // falso (0)
printf("Resultado = %d\n", r);
r = !(x > 2); // falso (0)
printf("Resultado = %d\n", r);
r = !(x > 7) && (x > y); // verdadeiro (1)
printf("Resultado = %d\n", r);
return 0;
}
```

DEPARTAMENTO DE COMPUTAÇÃO

Operadores de Pré e Pós Incremento/Decremento

- Esses operadores podem ser utilizados sempre que for necessário somar uma unidade (incremento) ou subtrair uma unidade (decremento) a determinado valor

Operador	Significado	Exemplo	Resultado
++	incremento	++x ou x++	x = x + 1
--	decremento	--x ou x--	x = x - 1

DEPARTAMENTO DE COMPUTAÇÃO

Operadores de Pré e Pós Incremento/Decremento

Operador	Significado	Resultado
++x	Pré-incremento	Soma + 1 a variável x antes de utilizar seu valor
x++	Pós-incremento	Soma + 1 a variável x depois de utilizar seu valor
--x	Pré-incremento	Subtrai -1 da variável x antes de utilizar seu valor
x--	Pos-incremento	Subtrai -1 da variável x depois de utilizar seu valor

DEPARTAMENTO DE COMPUTAÇÃO

Operadores de Pré e Pós Incremento/Decremento

- Usar o operador antes ou depois da variável somente tem importância se o operador for usado sozinho
  - Porém, se esse operador for utilizado dentro de uma expressão aritmética, a diferença entre os dois operadores será evidente!

```c
int main(){
int x, y;
x = 10;
y = x++;
printf("%d\n", x); // 11
printf("%d\n", y); // 10
y = ++x;
printf("%d\n", x); // 12
printf("%d\n", y); // 12

return 0;
}
```

DEPARTAMENTO DE COMPUTAÇÃO

Operadores de atribuição

- Operador de Atribuição: =
  - nome_da_variável = expressão, valor ou constante;

```
int main() {
int x = 5; // x recebe 5
int y;
y = x + 3; // y recebe x mais 5
return 0;
}
```

- O operador de atribuição "=" armazena o valor ou resultado de uma expressão contida a sua **direita** na variável especificada a sua **esquerda**
- A linguagem C suporta múltiplas atribuições
  - $x = y = z = 0;$

DEPARTAMENTO DE COMPUTAÇÃO

Operadores de atribuição

- Muitos operadores são sempre usados em conjunto com o operador de atribuição
  - Para tornar essa tarefa mais simples, a linguagem C permite simplificar algumas expressões

Operador	Significado	Exemplo		
+=	Soma e atribui	x += y	igual a	x = x + y
-=	Subtrai e atribui	x -= y	igual a	x = x – y
*=	Multiplica e atribui	x *= y	igual a	x = x * y
/=	Divide e atribui o quociente	x /= y	igual a	x = x / y
%=	Divide e atribui o resto	x %= y	igual a	x = x % y

DEPARTAMENTO DE COMPUTAÇÃO

Operadores de atribuição

Sem Operadores

```c
int main(){
int x = 10, y = 20;
x = x + y - 10;
printf("x = %d\n", x);
x = x- 5;
printf("x = %d\n", x);
x = x * 10;
printf("x = %d\n", x);
x = x / 15;
printf("x = %d\n", x);

return 0;
}
```

Com Operadores

```c
int main(){
int x = 10, y = 20;
x += y - 10;
printf("x = %d\n", x);
x -= 5;
printf("x = %d\n", x);
x *= 10;
printf("x = %d\n", x);
x /= 15;
printf("x = %d\n", x);

return 0;
}
```

DEPARTAMENTO DE COMPUTAÇÃO

Exercícios

- Diga o resultado das variáveis x, y e z depois da seguinte sequencia de operações:

1

```
int x, y, z;
x = y = 10;
z = ++x;
x -= x;
y++;
x = x + y - (z--);
```

2

```
int x, y;
int a = 14, b = 3;
float z;
x = a / b;
y = a % b;
z = y / x;
```

3

```
int x = 7;
(x > 5) || (x > 10)
(!(x == 6) && (x >= 6)
```

DEPARTAMENTO DE COMPUTAÇÃO

Conversão de Tipos

- Atribuição entre tipos diferentes
  - O compilador converte automaticamente o valor do lado direto para o tipo do lado esquerdo do operador de atribuição “=”
  - Pode haver perda de informação

```
int x = 65;
char c;
float f = 25.1;
// c recebe 8 bits menos significativos de x
// converte para a tabela ASCII
c = x;
printf("c = %c\n", c); // 'A'
// x recebe parte apenas a parte inteira de f
x = f;
printf("x = %d\n", x); //25
// f recebe valor 8 bits convertido para real
f = c;
printf("f = %f\n", f); // 65.00000
// f recebe o valor de x
f = x;
printf("f = %f\n", f); // 25.00000
```

DEPARTAMENTO DE COMPUTAÇÃO

Modeladores (Casts)

- Um modelador é aplicado a uma expressão
- Força o resultado da expressão a ser de um tipo especificado
  - (tipo) expressão
- **Exemplo:**

```
float x, y, f = 65.5;
x = f / 10.0;
y = (int) (f / 10.0);
printf("x = %f\n", x); // 6.550000
printf("y = %f\n", y); // 6.000000
```

DEPARTAMENTO DE COMPUTAÇÃO

Precedência dos Operadores

MAIOR PRECEDÊNCIA	
++ --	Pré-incremento/decremento
()	Parênteses (chamada de função)
[ ]	Elemento de array
.	Elemento de struct
->	Conteúdo de elemento de ponteiro para struct
++ --	Pós-incremento/decremento
+ -	Adição e subtração unária
! ~	Não lógico e complemento bit a bit
(tipo)	Conversão de tipos (*type cast*)
*	Acesso ao conteúdo de ponteiro
&	Endereço de memória do elemento
sizeof	Tamanho do elemento
* / %	Multiplicação, divisão e módulo (resto)
+ -	Adição e subtração
<< >>	Deslocamento de bits à esquerda e à direita
< <=	"Menor do que" e "menor ou igual a"
> >=	"Maior do que" e "maior ou igual a"
== !=	"Igual a" e "diferente de"
&	E bit a bit

DEPARTAMENTO DE COMPUTAÇÃO

Precedência dos Operadores

||	OU lógico
?:	Operador ternário
=	Atribuição
+= -=	Atribuição por adição ou subtração
*= /= %=	Atribuição por multiplicação, divisão ou módulo (resto)
<<= >>=	Atribuição por deslocamento de bits
&= ^= | =	Atribuição por operações lógicas
,	Operador vírgula
MENOR PRECEDÊNCIA	

DEPARTAMENTO DE COMPUTAÇÃO

Referências

- André Luiz Villar Forbellone, Henri Frederico Eberspächer, **Lógica de programação** (terceira edição), Pearson, 2005, ISBN 9788576050247.
- Ulysses de Oliveira, **Programando em C** - Volume I - Fundamentos, editora Ciência Moderna, 2008, ISBN 9788573936599
- **Slides baseados no material do site "Linguagem C Descomplicado"**
  - `https://programacaodescomplicada.wordpress.com/complementar/`

DEPARTAMENTO DE COMPUTAÇÃO