Métodos Computacionais

Operadores, Expressões Aritméticas e Entrada/Saída de Dados

Tópicos da Aula

- Hoje aprenderemos a escrever um programa em C que pode realizar cálculos
  - Conceito de expressão
  - Tipos de Operadores em C
  - Operador de atribuição
  - Operadores aritméticos
  - Operadores especiais
- Depois aprenderemos como utilizar funções de entrada/saída para escrever um programa
  - Funções de entrada
  - Funções de saída

Expressões

- Uma **expressão** é uma combinação de um ou mais **operadores** e **operandos** que geralmente realiza um cálculo
- A **avaliação** ou cálculo da expressão se faz obedecendo regras de associação e precedência estabelecidas na linguagem

```
int total = 3 + 4/2 ;
```

Expressão é avaliada e o resultado é atribuído a *total* que armazena agora o valor 5

Divisão (/) tem precedência sobre soma (+)

- O valor calculado pode não ser necessariamente um número
  - Pode ser um caractere, cadeia de caracteres, etc

Operadores em C

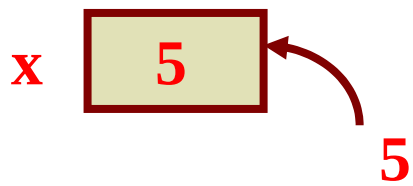
- Categorias de operadores em C
  - Atribuição
  - Aritméticos
  - Relacionais
  - Lógicos
  - Especiais

Operador de Atribuição

- Operador de Atribuição
  - Pode ser usado em qualquer expressão válida em C
  - Representado por =
  - Forma geral:

    <nome_da_variável> = <expressão> ;

    Ex.: x = 5 ;
  - Ação é executada da direita para a esquerda



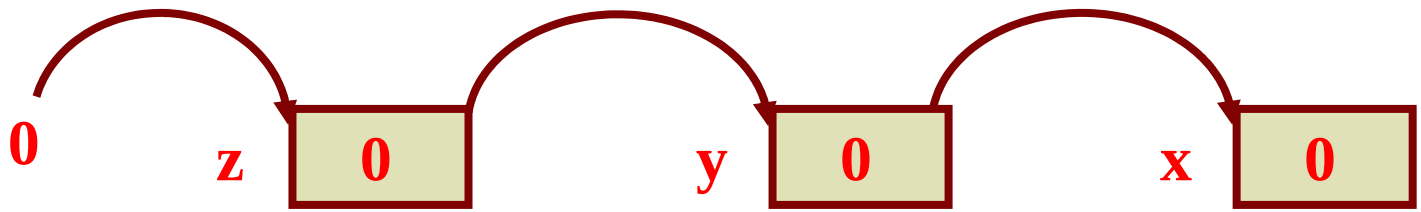
Operador de Atribuição

- Atribuição (Cont.)
  - É usado para atribuir valores às variáveis
  - Não é igual ao operador = de matemática

    5 = a **não faz sentido em C!**

  - Valor da expressão é armazenado fisicamente em <nome_da_variável>
  - Múltiplas atribuições

    Ex: x = y = z = 0



Operadores Aritméticos

- Operadores aritméticos unários
  - Um só operando

    operador operando

Sinal	Ação	Precedência
-	Troca de sinal	1ª
++	Incremento de 1	1ª
--	Decremento de 1	1ª

Ex.: -(a)

Operadores Aritméticos

- Operadores aritméticos binários
  - Dois operandos

operando operador operando

Sinal	Ação	Precedência
+	Adição	3ª
-	Subtração	3ª
*	Multiplicação	2ª
/	Divisão	2ª
%	Resto da Divisão (só para inteiros)	2ª

Operadores de Incremento e Decremento

- Operadores de *incremento* e *decremento* são operadores unários (usam um só operando)
- O operador de *incremento* (++) soma 1 ao seu operando
- O operador de *decremento* (--) subtrai 1 de seu operando
- A instrução

```
contador++;
```

é funcionalmente equivalente a

```
contador = contador + 1;
```

Operadores de Incremento e Decremento

- Estes operadores podem ser empregados de forma *pós-fixada* ou *pré-fixada*

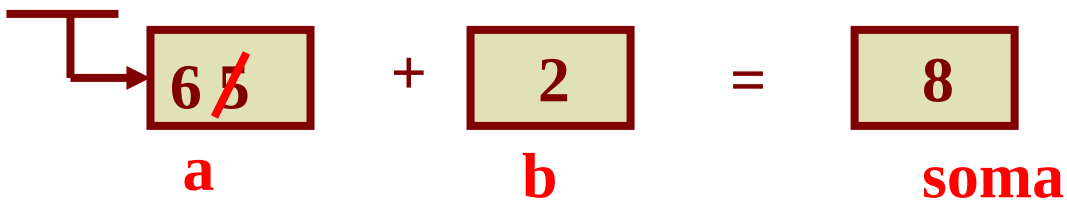
```
contador++;  ou  ++contador;
```

- Quando **isolados** têm comportamentos **equivalentes**
- Quando fazem parte de **expressões maiores**, eles podem ter comportamentos **diferentes**

Operadores de Incremento e Decremento

- Pré-fixado : ++a ou –a
  - Incrementa (decrementa) de 1 o valor de a. Se aparece em uma expressão, o valor é incrementado (decrementado) antes do cálculo da expressão

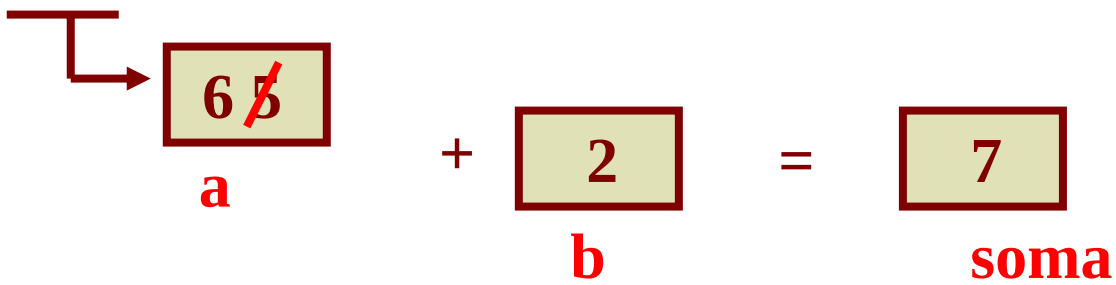
Pré: int soma = ++a + b



Operadores de Incremento e Decremento

- Pós-fixado: a++ ou a--
  - Incrementa (decrementa) de 1 o valor de a. Se aparece em uma expressão, o valor é incrementado (decrementado) após o cálculo da expressão

Pós: int soma = (a++) + b



Operadores de Incremento e Decremento

- Devem ser utilizadas com cuidado em expressões maiores!

```
int contador  = 3;
contador++;
++contador;
int valor  = contador++;
```

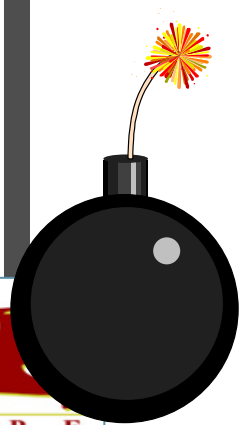
`contador` agora armazena 4

`contador` agora armazena 5

`valor` agora armazena 5 e depois `contador` é incrementado para 6

```
valor  = ++ contador;
```

`contador` é incrementado para 7 e agora `valor` armazena 7



Divisão e Resto da Divisão

- Se ambos operandos da expressão aritmética forem valores inteiros, o resultado será um inteiro (a parte decimal será descartada)
- Portanto

14/3	➡	4
8/12	➡	0
14%3	➡	2
8%12	➡	8

Operadores Aritméticos de Atribuição

- É comum fazermos algum tipo de operação com uma variável e depois armazenar o valor da operação na própria variável
  - Operadores aritméticos de atribuição facilitam codificação de expressões do tipo a = a op b
  - Forma geral
    - variavel op = expressão

```
x += 1;   ⟺   x = x + 1;
x -= 2;   ⟺   x = x - 2;
x *= k;   ⟺   x = x * k;
x /= 3;   ⟺   x = x / 3;
x %= 3;   ⟺   x = x % 3;
```

Expressões Aritméticas

- Uma **expressão aritmética** computa resultados numéricos e utiliza operadores aritméticos combinados com operandos numéricos
  - Variáveis, constantes, funções numéricas
  - Ordem de precedência
    - Operadores unários (- , -- , ++) e Funções
    - Multiplicação (*), Divisão (/) e Módulo (%)
    - Adição (+) e Subtração (-)
  - Comandos Equivalentes
    - a = a + 1; a += 1 ; a++ ; ++a ;

Operadores Aritméticos

Quais serão os valores das variáveis declaradas após a avaliação das expressões abaixo?

```
int a , r ;
double  b , c ;
a  = 3.5 ;
b = a / 2.0 ;
c  = 1/2 + b ;
r  = 10  %  a ;
```

Divisão Inteira! Avaliado como 0 (zero)

a=3 b=1.5 c=1.5 e r=1

Expressão Booleana

- O resultado da avaliação de uma **expressão booleana** é **ou verdadeiro ou falso**
  - Em C, **NÃO** existe o tipo de dado boolean
  - Verdadeiro é representado como 1
  - Falso é representado como 0

- Uma expressão booleana é composta de operandos booleanos (lógicos) e operadores **relacionais** e/ou **lógicos**

Operadores Relacionais

- São usados para fazer comparações

Operador	Ação
<	*menor que*
>	*maior que*
<=	*menor ou igual que*
>=	*maior ou igual que*
==	*igual a*
!=	*diferente de*

Resultado de Comparação

Falso ou Verdadeiro

4 < 5 é verdadeiro (valor 1)

3 >= 10 é falso (valor 0)

Operadores Lógicos (Booleanos)

- São usados para combinar comparações
  - Operam sobre valores booleanos (0 ou 1)

Operador	Ação
&&	*E*
||	*Ou*
!	*Negação*

Resultado da Avaliação

```
int   a , b ;
int   c = 23 ;
int   d = 27 ;
a = ( c < 20 ) || ( d > c ) ;   /*  1 */
b = (c < 20 ) && ( d< c ) ;   /*  0  */
```

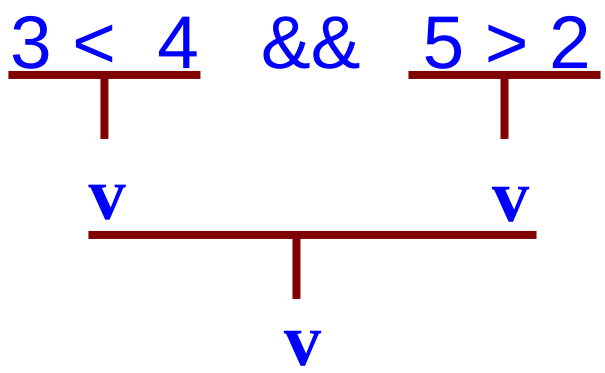
Tabela Verdade

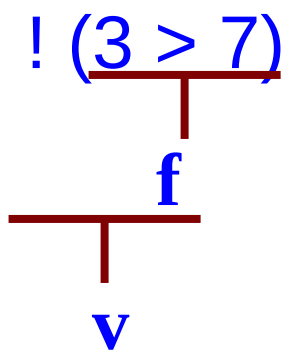
- Uma tabela verdade contém todas as combinações verdadeiro-falso de uma expressão booleana

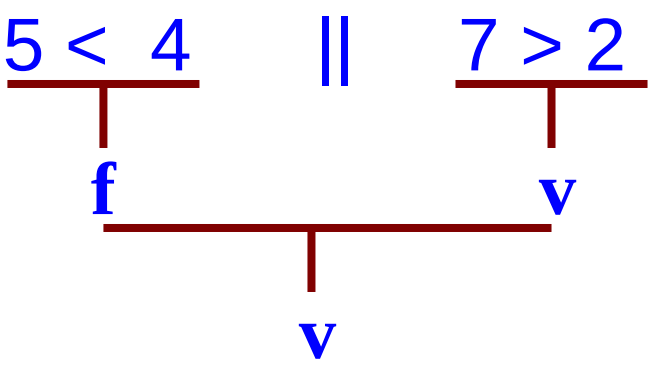
a	b	a && b	a || b	!a
verdadeiro	verdadeiro	verdadeiro	verdadeiro	falso
verdadeiro	falso	falso	verdadeiro	falso
falso	verdadeiro	falso	verdadeiro	verdadeiro
falso	falso	falso	falso	verdadeiro

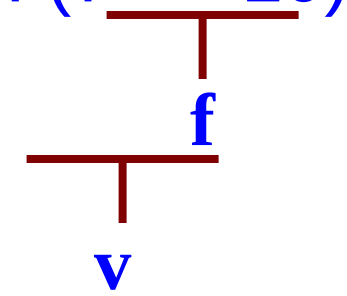
Expressões Booleanas

- Exemplos









Expressões Booleanas

```
{   ...
    int b, c;
    b = 1;
    c = !b;
    c = !(1 || b) && c;
    b = c || !(!b);
    ...
}
```

Parênteses são usados para evitar ambigüidades

Qual o valor de **b** neste ponto?

1 (verdadeiro)

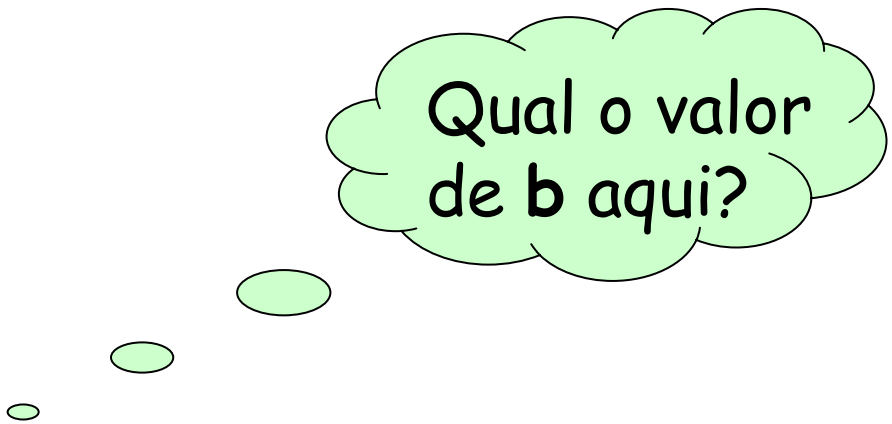
Expressões Booleanas

```
int b, c;

b = 1 || 0;

c = 1 && b;

b = b == c;
```



1 (verdadeiro)

Operadores Especiais

- O operador ***&*** é utilizado para se obter o endereço da memória que a variável representa
  - Forma geral
    - &variável
- Um endereço de memória é visto como um número inteiro sem sinal

Conversão de Tipos

- Existem conversões automáticas de valores em uma avaliação de uma expressão quando operandos possuem tipos diferentes
  - Operando de tipo de menor tamanho é convertido automaticamente para o tipo de maior tamanho
    - Conversão é feita em área temporária da memória antes da avaliação da expressão
    - Resultado é novamente convertido para o tipo da variável a esquerda da atribuição

1. O inteiro 3 é convertido para real

```
int a  =  3/2.0 + 0.5;
```

2. Expressão é avaliada como 2.0,

3. Valor é convertido para um inteiro e atribuído a variável

Valor de a é 2

Operadores Especiais (Cast)

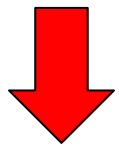
- Algumas vezes a conversão automática dá resultados não desejados
- Devemos então usar o operador de ***cast***
  - Forma geral
    - (tipo desejado) variável ou (tipo desejado) (expressão)
    - Armazenamento de um valor real em um tipo de dado inteiro gera erro ou perde-se precisão

```
int a = 3/2 + 0.5;
printf("a = %d",a);
```

a = 1

Usando cast

```
int a = ((float)3)/2 +0.5;
printf("a = %d",a);
```



a = 2

Entrada de Dados

- A função ***scanf***
  - Usada para a entrada formatada de dados
  - Para cadeia de caracteres, a leitura é feita até o primeiro espaço em branco, ou o return, ou o tab.
  - Está definida na biblioteca “stdio.h”

- Forma Geral: Tem duas partes:

  scanf (“expressão de controle”, lista de argumentos)
  - Expressão de controle
    - Códigos de formatação, precedidos por **%**

Entrada de Dados

- Códigos de Formatação

Código	Função
%c	ler um único caractere
%d	ler um número inteiro
%u	ler um inteiro sem sinal
%f %e %g	ler um número real (tipo *float*)
%lf %le %lg	ler um número real (tipo *double*)
%l	ler um inteiro longo
%s	ler uma cadeia de caracteres

Entrada de Dados

- Lista de Argumentos
  - Cada código de formatação deve corresponder a uma variável de entrada, que deve ser representada com o operador de endereço
- Operador de Endereço
  - Utilizado para permitir que o dado lido do dispositivo de entrada seja armazenado na variável correspondente. Ele retorna o endereço da variável.

```
int a ;
```

&a → endereço da variável a

```
scanf ( " % d " , &a ) ;
```

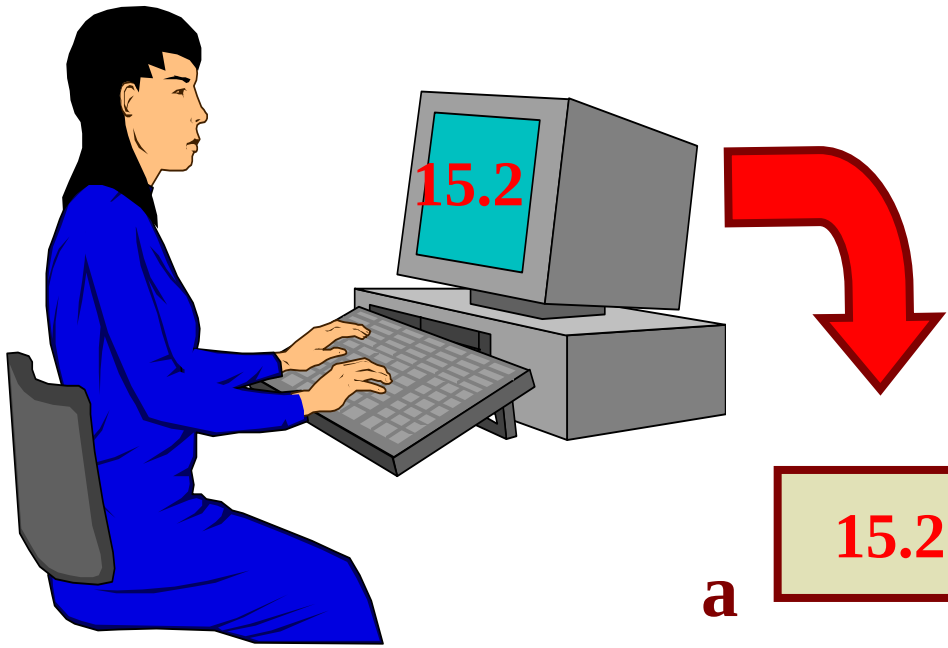
Entrada de Dados

- Exemplo:

```
char   a, b, c ;
int  n ;   double  x ;
scanf (“%c%c%c%d%lf ”,&a,&b,&c,&n,&x);
```

Entrada de Dados

```
#include   “stdio.h“
 int main ( )     /* ler um valor numérico */
  { float a;
     scanf(“%f”,&a);
      .....
  }
```



Saída de Dados

- A Função printf
  - Permite a impressão formatada de números e cadeias de caracteres
- Forma Geral

printf(“estruturas de controle”, lista de parâmetros)

  - Estruturas de Controle
    - Caracteres especiais
    - Códigos de formatação precedidos por %

Saída de Dados

Formato	Significado
%c	caracter
%d	inteiro
%u	inteiro sem sinal
%e	notação científica com e (7.12300e+00)
%E	notação científica com E (7.12300E+00)
%f	ponto flutuante decimal (7.12300)
%g	escolhe %e ou %f ,o menor dos 2 formatos
%G	escolhe %E ou %f ,o menor dos 2 formatos
%s	cadeia de caracteres
%%	imprime o caracter ‘%’

Exemplo de Saída de Dados

- Suponha o seguinte código em C

```
int i = 123 ;
float x = 0.123456789 ;
```

Formato	*Argumento*	*Saída*
%d	i	123
%05d	i	00123
%10.5f	x	bbbb0.12346
%-12.5e	x	1.23456e-01b

Exemplo E/S de Dados

- Escreva um programa em C para ler 2 valores para as variáveis A e B, efetuar a troca dos valores de forma que a variável A passe a possuir o valor da variável B e que a variável B passe a possuir o valor da variável A. Apresentar os valores trocados.

Exemplo E/S de Dados

```
#include “stdio.h”
int main(  ){
  int   a , b, aux;
  printf (“\nTroca de valores entre variáveis\n”);
  printf (“\nEntre com o valor de A: ” );
  scanf (“%d”, &a );
  printf (“\nEntre com o valor de B: ”) ;
  scanf (“%d”, &b );
  aux  =  a ;
  a  =  b ;
  b  =  aux ;
  printf (“A variável A agora vale: %d”, a ) ;
  printf (“ \n”) ;
  printf ( “A variável B agora vale: %d”, b ) ;
  return 0;
}
```