PUC Minas

Operadores e Expressões

João Pedro Oliveira Batisteli

Fevereiro, 2025

Operadores

A maioria dos programas em C executa cálculos aritméticos (operações numéricas).

Operadores

A maioria dos programas em C executa cálculos aritméticos (operações numéricas).

Operação em C	Operador aritmético	Expressão algébrica	Expressão em C
Adição	+	$f + 7$	f + 7
Subtração	–	$p - c$	p – c
Multiplicação	*	bm	b * m
Divisão	/	x / y ou $\frac{x}{y}$ ou $x \div y$	x / y
Módulo ou resto da divisão entre 2 inteiros	%	$r \bmod s$	r % s

Operadores

OBS 1: A divisão inteira (divisão entre variáveis do tipo `int`) gera um resultado inteiro.

Operadores

OBS 1: A divisão inteira (divisão entre variáveis do tipo `int`) gera um resultado inteiro.

Ex: o resultado da expressão $7/4$ é 1, e o resultado da expressão $17/5$ é 3.

Operadores

OBS 1: A divisão inteira (divisão entre variáveis do tipo `int`) gera um resultado inteiro.

Ex: o resultado da expressão $7/4$ é 1, e o resultado da expressão $17/5$ é 3.

OBS 2: O operador de módulo só pode ser usado com operandos inteiros.

Operadores

OBS 1: A divisão inteira (divisão entre variáveis do tipo `int`) gera um resultado inteiro.

Ex: o resultado da expressão $7/4$ é 1, e o resultado da expressão $17/5$ é 3.

OBS 2: O operador de módulo só pode ser usado com operandos inteiros.

Ex: o resultado de $7\%4$ é 3, e o resultado de $17\%5$ é 2.

Operadores

O que ocorrerá se executarmos o seguinte código?

Operadores

O que ocorrerá se executarmos o seguinte código?

```
#include <stdio.h>
int main(){
  int a=5;
  int b=0;
  int div = a/b;
  return 0;
}
```

Operadores

O que ocorrerá se executarmos o seguinte código?

```
#include <stdio.h>
int main(){
  int a=5;
  int b=0;
  int div = a/b;
  return 0;
}
```

A tentativa de dividir por zero normalmente é indefinida em sistemas computacionais, e em geral resulta em um erro fatal.

Parênteses para agrupamento de subexpressões

- Os parênteses são usados em expressões em C da mesma maneira que em expressões algébricas.

Parênteses para agrupamento de subexpressões

- Os parênteses são usados em expressões em C da mesma maneira que em expressões algébricas.
- Por exemplo, para multiplicar `a` pelo resultado de `b + c`, escrevemos `a * (b + c)`.

Parênteses para agrupamento de subexpressões

- Os parênteses são usados em expressões em C da mesma maneira que em expressões algébricas.
- Por exemplo, para multiplicar `a` pelo resultado de `b + c`, escrevemos `a * (b + c)`.
- Os parênteses podem ser usados para forçar a ordem de cálculo a acontecer em uma sequência desejada.

Parênteses para agrupamento de subexpressões

- Os parênteses são usados em expressões em C da mesma maneira que em expressões algébricas.
- Por exemplo, para multiplicar `a` pelo resultado de `b + c`, escrevemos `a * (b + c)`.
- Os parênteses podem ser usados para forçar a ordem de cálculo a acontecer em uma sequência desejada.

Regras de precedência de operadores

- C aplica os operadores nas expressões aritméticas em uma sequência determinada pelas regras de precedência de operadores, que geralmente são iguais às regras da álgebra.

Regras de precedência de operadores

- C aplica os operadores nas expressões aritméticas em uma sequência determinada pelas regras de precedência de operadores, que geralmente são iguais às regras da álgebra.
- Dizemos que os parênteses estão no nível **mais alto de precedência**.

Regras de precedência de operadores

- C aplica os operadores nas expressões aritméticas em uma sequência determinada pelas regras de precedência de operadores, que geralmente são iguais às regras da álgebra.
- Dizemos que os parênteses estão no nível **mais alto de precedência**.
- Assim, os parênteses podem ser usados para forçar a ordem de cálculo a acontecer em uma sequência desejada qualquer.

Regras de precedência de operadores

Dada a seguinte expressão escrita em C:

Regras de precedência de operadores

Dada a seguinte expressão escrita em C:

$$((a + b) + c)$$

Qual operação é realizada primeiro?

Regras de precedência de operadores

Dada a seguinte expressão escrita em C:

$$((a + b) + c)$$

Qual operação é realizada primeiro?

Os operadores no par mais interno de parênteses são aplicados primeiro.

Regras de precedência de operadores

Operador(es)	Operação(ões)	Ordem de avaliação (precedência)
( )	Parênteses	Avaliados em primeiro lugar. Se os parênteses forem aninhados, a expressão no par mais interno é a primeira a ser avaliada. Se houver vários pares de parênteses 'no mesmo nível' (ou seja, não aninhados), eles serão avaliados da esquerda para a direita.
* / %	Multiplicação Divisão Módulo	Avaliados em segundo lugar. Se houver vários, serão avaliados da esquerda para a direita.
+ –	Adição Subtração	Avaliados por último. Se houver vários, serão avaliados da esquerda para a direita.

Exercícios

1) Avalie a expressão a seguir escrita em C e indique a ordem em que a linguagem avalia os operadores:

```
z = p * r % q + w /x - y;
```

Exercícios

1) Avalie a expressão a seguir escrita em C e indique a ordem em que a linguagem avalia os operadores:

```
z = p * r % q + w /x - y;
```

2) Faça um programa que receba uma temperatura em Celsius, calcule e mostre essa temperatura em Fahrenheit. Sabe-se que
$F = 180 * (C + 32)/100$.

Operadores Matemáticos de Atribuição

- Também conhecidos como operadores compostos de atribuição.

Operadores Matemáticos de Atribuição

- Também conhecidos como operadores compostos de atribuição.
- +=, -=, *=, /= e %=.

Operadores Matemáticos de Atribuição

- Também conhecidos como operadores compostos de atribuição.
- +=, -=, *=, /= e %=.
- `x += y` equivale a `x = x + y`.

Operadores Matemáticos de Atribuição

- Também conhecidos como operadores compostos de atribuição.
- `+=`, `-=`, `*=`, `/=` e `%=`.
- `x += y` equivale a `x = x + y`.
- `x -= 8` equivale a `x = x - 8`.

Operadores Matemáticos de Atribuição

- Também conhecidos como operadores compostos de atribuição.
- +=, -=, *=, /= e %=.
- `x += y` equivale a `x = x + y`.
- `x -= 8` equivale a `x = x - 8`.
- `x *= z` equivale a `x = x * z`.

Operadores Matemáticos de Atribuição

- Também conhecidos como operadores compostos de atribuição.
- +=, -=, *=, /= e %=.
- `x += y` equivale a `x = x + y`.
- `x -= 8` equivale a `x = x - 8`.
- `x *= z` equivale a `x = x * z`.
- `x /= y` equivale a `x = x / y`.

Operadores Matemáticos de Atribuição

Operador de atribuição	Exemplo de expressão	Explicação	Atribui
Considere: **int** c = 3, d = 5, e = 4, f = 6, g = 12;			
+=	c += 7	c = c + 7	10 a c
-=	d -= 4	d = d - 4	1 a d
*=	e *= 5	e = e * 5	20 a e
/=	f /= 3	f = f / 3	2 a f
%=	g %= 9	g = g % 9	3 a g

Operadores Matemáticos de Incremento

C também possui operador unário de incremento $++$ e o operador unário de decremento $--$.

Operadores Matemáticos de Incremento

C também possui operador unário de incremento $++$ e o operador unário de decremento $--$.

- $++$

Operadores Matemáticos de Incremento

C também possui operador unário de incremento $++$ e o operador unário de decremento $--$.

- $++$
- x$++$ equivale a $x = x + 1$

Operadores Matemáticos de Incremento

C também possui operador unário de incremento $++$ e o operador unário de decremento $--$.

- $++$
- x$++$ equivale a $x = x + 1$
- $--$

Operadores Matemáticos de Incremento

C também possui operador unário de incremento $++$ e o operador unário de decremento $--$.

- $++$
- x$++$ equivale a $x = x + 1$
- $--$
- y$--$ equivale a $y = y - 1$

Exercício

Qual a saída do seguinte programa?

Exercício

Qual a saída do seguinte programa?

```
#include <stdio.h>
int main(){
  int x=7;
  printf("O valor de X é %d \n",x--);
  printf("O valor de X é %d \n",x--);
  printf("O valor de X é %d \n",x--);
  printf("O valor de X é %d \n",x);
  return 0;
}
```

Operadores Matemáticos de Incremento

Operador	Exemplo de expressão	Explicação
++	++a	Incrementa a em 1, e então usa o novo valor de a na expressão em que a estiver.
++	a++	Usa o valor corrente de a na expressão em que a estiver, e então incrementa a em 1.
--	--b	Decrementa b em 1, e então usa o novo valor de b na expressão em que b estiver.
--	b--	Usa o valor corrente de b na expressão em que b estiver, e então decrementa b em 1.

Operadores relacionais e de igualdade

- As instruções executáveis em C tanto realizam ações (como cálculos ou entrada/saída de dados) quanto tomam decisões.

Operadores relacionais e de igualdade

- As instruções executáveis em C tanto realizam ações (como cálculos ou entrada/saída de dados) quanto tomam decisões.
- Por exemplo, em um programa, poderíamos tomar a decisão de determinar se a nota de um aluno é maior ou igual a 60, e, caso for, imprimir a mensagem "*Parabéns! Você foi aprovado*".
- Por exemplo, em um programa, poderíamos tomar a decisão de determinar se a nota de um aluno é maior ou igual a 60, e, caso for, imprimir a mensagem "*Parabéns! Você foi aprovado*".
- Se a condição do operador é verdadeiro, o resultado será *true*, caso contrário o resultado será *false*.

Operadores relacionais e de igualdade

Operador de igualdade ou relacional na álgebra	Operador de igualdade ou relacional em C	Exemplo de condição em C	Significado da condição em C
Operadores de igualdade			
$=$	==	x == y	x é igual a y
$\neq$	!=	x != y	x não é igual a y
Operadores relacionais			
$>$	>	x > y	x é maior que y
$<$	<	x < y	x é menor que y
$\geq$	>=	x >= y	x é maior ou igual a y
$\leq$	<=	x <= y	x é menor ou igual a y

Exemplo

```
#include <stdio.h>
int main(){
  int x=7;
  int y=8;
  printf("%b",x==y);
  return 0;
}
```

Exemplo

```
#include <stdio.h>
int main(){
  int x=7;
  int y=8;
  printf("%b",x==y);
  return 0;
}
```

Qual será a saída do programa?

Exemplo

```
#include <stdio.h>
int main(){
  int x=7;
  int y=8;
  printf("%b",x==y);
  return 0;
}
```

Qual será a saída do programa?

Qual será a saída do programa se o operador for !=, <, <=, >, >= ?

Operadores relacionais e de igualdade

- Obsimp: em C, podemos usar o valor 0 para denotar *false* e qualquer outro valor inteiro para denotar *true*.

Operadores lógicos

- Operadores lógicos podem ser usados para formar condições mais complexas ao combinar.

Operadores lógicos

- Operadores lógicos podem ser usados para formar condições mais complexas ao combinar.
- **Ex**: Condição para um aluno ser aprovado em uma disciplina:

$$\text{Nota} >= 60 \text{ e faltas} <= 25\%$$

Operadores lógicos

- Operadores lógicos podem ser usados para formar condições mais complexas ao combinar.
- **Ex**: Condição para um aluno ser aprovado em uma disciplina:

$$\text{Nota} >= 60 \text{ e faltas} <= 25\%$$

- Essa condição mais *complexa* contém duas condições *simples*.

Operadores lógicos

- Operadores lógicos podem ser usados para formar condições mais complexas ao combinar.
- **Ex**: Condição para um aluno ser aprovado em uma disciplina:

$$\text{Nota} >= 60 \text{ e faltas} <= 25\%$$

- Essa condição mais *complexa* contém duas condições *simples*.

Operadores lógicos

- Negação(not) - !

Operadores lógicos

- Negação(not) - !
- Ex: !true = false, !false = true

Operadores lógicos

- Negação(not) - !
- Ex: !true = false, !false = true
- *E* lógico (And) - &&

Operadores lógicos

- Negação(not) - !
- Ex: !true = false, !false = true
- *E* lógico (And) - &&
- *Ou* lógico (Or) - ||

Operadores lógicos

Op1	Op2	Op1 || Op2
V	V	V
V	F	V
F	V	V
F	F	F

Operadores lógicos

Op1	Op2	Op1 && Op2
V	V	V
V	F	F
F	V	F
F	F	F

Exemplo

```
x=2; y=3; z=4; achou=false;
```

Exemplo

```
x=2; y=3; z=4; achou=false;
(x < y+1) && (!achou) && (z == x+3)
```

Exemplo

```
x=2; y=3; z=4; achou=false;
(x < y+1) && (!achou) && (z == x+3)
(2 < 3 + 1) && (!false) && (4 == 2 + 3)
```

Exemplo

```
x=2; y=3; z=4; achou=false;
(x < y+1) && (!achou) && (z == x+3)
(2 < 3 + 1) && (!false) && (4 == 2 + 3)
(2 < 4) && true && (4 == 5)
```

Exemplo

```
x=2; y=3; z=4; achou=false;
(x < y+1) && (!achou) && (z == x+3)
(2 < 3 + 1) && (!false) && (4 == 2 + 3)
(2 < 4) && true && (4 == 5)
```

true && true && false

Exemplo

```
x=2; y=3; z=4; achou=false;
(x < y+1) && (!achou) && (z == x+3)
(2 < 3 + 1) && (!false) && (4 == 2 + 3)
(2 < 4) && true && (4 == 5)
```

true && true && false

true && false

Exemplo

```
x=2; y=3; z=4; achou=false;
(x < y+1) && (!achou) && (z == x+3)
(2 < 3 + 1) && (!false) && (4 == 2 + 3)
(2 < 4) && true && (4 == 5)
```

true && true && false

true && false

false

Funções Matemáticas

- A biblioteca *math.h* fornece um conjunto de funções matemáticas que permitem realizar cálculos maix complexos, como funções trigonométricas, exponenciais, logarítmicas, etc

Funções Matemáticas

- A biblioteca *math.h* fornece um conjunto de funções matemáticas que permitem realizar cálculos maix complexos, como funções trigonométricas, exponenciais, logarítmicas, etc
- O resultado de cada função é do tipo double.

Funções Matemáticas

- A biblioteca *math.h* fornece um conjunto de funções matemáticas que permitem realizar cálculos maix complexos, como funções trigonométricas, exponenciais, logarítmicas, etc
- O resultado de cada função é do tipo double.
- Funções trigonométricas: `sin(x)`, `cos(x)`, `tan(x)`, `asin(x)`, `acos(x)`, `atan(x)`.

Funções Matemáticas

- A biblioteca *math.h* fornece um conjunto de funções matemáticas que permitem realizar cálculos maix complexos, como funções trigonométricas, exponenciais, logarítmicas, etc
- O resultado de cada função é do tipo double.
- Funções trigonométricas: `sin(x)`, `cos(x)`, `tan(x)`, `asin(x)`, `acos(x)`, `atan(x)`.
- Funções exponenciais e logarítmicas: `exp(x)`, `log(x)`, `log10(x)`, `pow(x, y)`, `sqrt(x)`.

Exemplo

Incluir a biblioteca *math.h*.

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265
int main (){
  double x, ret, val;
  x = 60.0;
  val = PI / 180.0;
  ret = cos( x*val );
  printf(\Cosseno de %lf é %lf graus\n", x, ret);
  return(0);
}
```

Funções Matemáticas em C

Função	Descrição do comando
floor( )	arredonda para baixo
ceil( )	arredonda para cima
sqrt( )	Calcula raiz quadrada
pow(variável, expoente)	potenciação
sin( )	seno
cos( )	cosseno
tan( )	Tangente
log( )	logaritmo natural
log10( )	logaritmo base 10

Funções Matemáticas em C

- O sorteio de número aleatórios é muito utilizado em diversas aplicações.

Funções Matemáticas em C

- O sorteio de número aleatórios é muito utilizado em diversas aplicações.
- Pode-se utilizar a hora do sistema como base para geração dos números aleatórios com a função *srand* e *time*, além de definir um intervalo (mínimo e máximo) dos valores que serão sorteados por uma expressão utilizada com a função rand.

Exemplo srand

```
#include <stdio.h>
#include <math.h>
#include <time.h>
int main (){
  // Inicializa gerador de números
  srand((unsigned) time(NULL));
  // Sorteia número entre 40 e 149
  printf("%d\n", (rand() % (150-40))+40);
  return(0);
}
```

Exercício

1. Fazer um código que leia três números reais a, b e c e mostrar o valor de y, sendo y = a + b*c + a + 2 * (a - b) + $log_{10}(64)$.

Dúvidas?