Algoritmos e Programação de Computadores

Operadores

Prof. Lucas Boaventura
lucas.boaventura@unb.br

Introdução

- Vimos que as variáveis podem ser declaradas para armazenar valores que serão usados durante o programa
- Um dos principais usos das variáveis é a manipulação de dados
- Na aula de hoje, vamos estudar operações matemáticas, atribuições e lógica que podemos realizar nas variáveis

Atribuição

- Uma das operações mais básicas na programação é a atribuição
- Nele, uma cópia de um valor é adicionado para a variável
- nome_da_variavel = outra_variavel;

Atribuição

- Atribuições podem ocorrer entre variáveis

```c
#include <stdio.h>

int main()
{
    int x = 5;
    int z;

    z = x;
    printf("%d\n", z);
    return 0;
}
```

Saída do código: 5

Atribuição

- O valor da parte direita da operação é copiado para a variável na parte esquerda
- No entanto, é preciso observar algumas regras: deve ser possível escrever em quem está na parte esquerda do comando

Atribuição

- // é um comentário, onde o programador explica algo no código: ignorado pelo compilador!

error: lvalue required as left operand of assignment

error: assignment of read-only variable ‘z’

```c
#include <stdio.h>

int main()
{
    int x = 5;
    const int z = 10;

    //Errado!
    6 = x;
    z = x;

    printf("%d\n", z);
    return 0;
}
```

Atribuição

```c
#include <stdio.h>

int main()
{
    float f = 80.9;
    int x;
    char c;

    x = f;
    c = x;
    printf("Char c: %c\nInt x: %d\n", c, x);
    return 0;
}
```

Saída do código:

Char c: P
Int x: 80

Atribuição

- Cuidado! Os tipos de variáveis possuem diferentes capacidades, alguma informação pode ser perdida ao se copiar para tipos diferentes

Operadores Aritméticos

- Além da atribuição, diversas operações podem ser feitas para manipular as variáveis:
  - + operador de soma
  - - operador de subtração
  - * operador de multiplicação
  - / operador de divisão (quociente)
  - % operador de resto de divisão

Operadores Aritméticos

```c
#include <stdio.h>

int main()
{
    int m = 10;
    int n = 15;
    int o, p, q, r;

    o = m + n;  // Resultado: 25
    p = n - m;  // Resultado: 5
    q = n / 15; // Resultado: 1
    r = n % m;  // Resultado: 5
}
```

Várias variáveis podem ser declaradas na mesma linha!

Operadores Aritméticos

- O operador – pode ser utilizado como a multiplicação por -1
- De forma semelhante, o operador + como multiplicação de +1, mas não faz muito sentido...

Operadores Aritméticos

```c
#include <stdio.h>

int main()
{
    int x = 5;
    int w = -5;
    int y, z;

    y = -x;
    z = +w;

    printf("%d %d\n", y, z);
    return 0;
}
```

Saída do código:

-5 -5

Operadores Aritméticos

```c
#include <stdio.h>

int main()
{
    float x, y;

    x = 5 / 4;
    y = 5 / 4.0;

    printf("%f %f\n", x, y);
    return 0;
}
```

Cuidado ao atribuir números inteiros a float

Saída:

1.000000 1.250000

Operadores Relacionais

- Os operadores relacionais são usados para comparações:

Operador	Significado
>	Maior
>=	Maior ou igual
<	Menor
<=	Menor ou igual
==	Igual
!=	Diferente

Operadores Relacionais

- A linguagem C não possui um “tipo” lógico
  - bool b; //NAO e' um codigo em C
- No C, são utilizados valores inteiros:
  - Valor 0: falso
  - Valor diferente de 0: verdadeiro
- Já os operadores relacionais retornam:
  - Valor 0 para falso
  - Valor 1 para verdadeiro

Operadores Relacionais

```c
#include <stdio.h>

int main()
{
    int i;

    i = 5 > 0;
    printf("%d\n", i);

    i = 2 < 3;
    printf("%d\n", i);

    i = 5 >= 5;
    printf("%d\n", i);
    return 0;
}
```

Saída:

1

1

1

Operadores Relacionais

- O operador == pode ser usado para comparar
- Ele é frequentemente confundido com o =
  - == operador de COMPARAÇÃO
  - = operador de ATRIBUIÇÃO

Operadores Relacionais

```c
#include <stdio.h>

int main()
{
    int i;
    int x = 3;
    int y = 4;

    i = x == y;
    printf("%d %d %d\n", i, x, y);

    i = x = y;
    printf("%d %d %d\n", i, x, y);
    return 0;
}
```

i recebe 1 se x igual a y

i e x recebem o valor de y

Saída:

0 3 4

4 4 4

Operadores Lógicos

- Operador &&: AND “E” lógico
- Operador ||: OR “OU” lógico

A	B	A && B	A || B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Lembrete: 0 é falso (F), 1 é verdadeiro (V)

Operadores Lógicos

- Operações complexas de lógica serão vistas mais adiante do curso (Matemática Discreta, Lógica, Lógica Computacional, ...)
- Em APC usaremos operações simples de lógica

Operadores Lógicos

```c
#include <stdio.h>

int main()
{
    int a = 10;
    int res;

    //A maior que zero E a menor que 20
    res = (a > 0) && (a < 20);
    printf("Resultado 1: %d\n", res);

    //A maior que zero E a menor que 5
    res = (a > 0) && (a < 5);
    printf("Resultado 2: %d\n", res);

    //A maior que zero OU a menor que 5
    res = (a > 0) || (a < 5);
    printf("Resultado 3: %d\n", res);
    return 0;
}
```

Saída:

Resultado 1: 1
Resultado 2: 0
Resultado 3: 1

Operadores Lógicos

- Operador !: NOT, negação

A	!A
0	1
1	0

Lembrete: 0 é falso (F), 1 é verdadeiro (V)

Operadores Lógicos

```c
#include <stdio.h>

int main()
{
    int x = 10, y = 15, z = 20;
    int r;

    r = (x < 20) && (y > 10); //1: verdadeiro
    r = (z < 30) && (x < 0); //0: falso
    r = (z < 30) || (x < 0); //1: verdadeiro
    r = !((x > 5) && (z < 20)); //1: verdadeiro
    return 0;
}
```

Operadores Bit a bit

- Alguns operadores lógicos podem ser usados para operar os bits da variável, não a variável toda

Operador	Uso
&	e bit a bit
|	ou bit a bit
^	ou exclusivo bit a bit
~	complemento
<<	Deslocamento a esquerda
>>	Deslocamento a direita

Operadores Bit a bit

- & e |

x	y	x & y	x | y
0101 (5)	0011 (3)	0001 (1)	0111 (7)
1101 (13)	0111 (7)	0101 (5)	1111 (15)

Operadores Bit a bit

```c
#include <stdio.h>

int main()
{
    int x = 5, y = 3;
    int and, or;

    and = x & y;
    or = x | y;
    printf("%d %d\n", and, or);
    return 0;
}
```

Saída:

1 7

Operadores Bit a bit

```c
#include <stdio.h>

int main()
{
    int x = 13, y = 7;
    int and, or;

    and = x & y;
    or = x | y;
    printf("%d %d\n", and, or);
    return 0;
}
```

Saída:
5 15

Operadores Bit a bit

- ^ e ~

x	y	x ^ y	~x
0101 (5)	0011 (3)	0110 (6)	1010 (10)
1101 (13)	0111 (7)	1010 (10)	0010 (2)

Operadores Bit a bit

```c
#include <stdio.h>

int main()
{
    int x = 5, y = 3;
    int xor, comp;

    xor = x ^ y;
    comp = ~x;
    printf("%d %d\n", xor, comp);
    return 0;
}
```

Saída:
6 -6

int tem 32 bits, não 4

Operadores Bit a bit

```c
#include <stdio.h>

int main()
{
    int x = 13, y = 7;
    int xor, comp;

    xor = x ^ y;
    comp = ~x;
    printf("%d %d\n", xor, comp);
    return 0;
}
```

Saída:
10 -14

int tem 32 bits, não 4

Operadores Bit a bit

- << e >>

x	x << 1	x << 4	x >> 1
0000 0001 (1)	0000 0010 (2)	0001 0000 (16)	0000 0000 (0)
0000 1101 (13)	0001 1010 (26)	1101 0000 (208)	0000 0110 (6)

Operadores Bit a bit

```c
#include <stdio.h>

int main()
{
    char x = 1;
    int r1 = x << 1;
    int r2 = x << 4;
    int r3 = x >> 1;
    printf("%d\n", x);
    printf("%d\n", r1);
    printf("%d\n", r2);
    printf("%d\n", r3);
    return 0;
}
```

Saída:

1
2
16
0

Operadores Simplificados

- Operadores de atribuição simplificada podem ser usados para reduzir o tamanho do código

x = x + 5;	+=	x += 5;
x = x * 10;	*=	x *= 10;
x = x – 3;	-=	x -= 3;

Operadores Simplificados

- Além disso, o operador ++ e – – são utilizados para as operações
- x = x + 1;
- e
- x = x – 1;
- Podem ser utilizados como pós-incremento ou como pré-incremento

Operadores Simplificados

Saída:
Antes: 10
Durante: 10
Depois: 11
Antes: 11
Durante: 12
Depois: 12

```c
#include <stdio.h>

int main()
{
    int x = 10;

    printf("Antes: %d\n", x);
    printf("Durante: %d\n", x++);
    printf("Depois: %d\n", x);

    printf("Antes: %d\n", x);
    printf("Durante: %d\n", ++x);
    printf("Depois: %d\n", x);
    return 0;
}
```

Pós-incremento

Pré-incremento

Precedência dos operadores

- Atribuições podem ocorrer entre expressões aritméticas
- nome_da_variavel = expressão;
- Nesse caso, precisamos analisar a ordem de precedência dos operadores
- A ordem pode ser utilizadas sempre organizada por parênteses

Precedência dos operadores

- x = 10 + 3 * 2;
- É interpretada como:
- x = 10 + (3 * 2);
- Resultado final: 16
- Para mudar a ordem, use parênteses:
- x = (10 + 3) * 2;
- Resultado final: 26

Precedência dos operadores

- Algumas ordens de precedência:
  - ++, -- (pré)
  - ++, -- (pós)
  - *, /, %
  - +, -
  - <<, >>
  - <=, >=, <, >
  - (cont...)

(cont)

==, !=

&

^

|

&&

||

=

+=, -=

Dúvidas?

- lucas.boaventura@unb.br