Operadores e Expressões

Profa. Elloá B. Guedes

www.elloaguedes.com

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

Valores Lógicos

- No ANSI C 89/90 não existe nenhum tipo específico de dados para armazenar valores lógicos

- Convenção
  - O valor lógico FALSO é representado por ZERO
  - Tudo aquilo que seja diferente de zero representa o valor lógico VERDADEIRO

- Cuidado!
  - O valor 1 é apenas um dos valores possíveis para representar a verdade

- Relembre o tipo _Bool da aula passada
  - Inserido no padrão ANSI C 99

Operadores Aritméticos

- C oferece 6 operadores aritméticos binários (operam sobre dois operandos) e um operador aritmético unário (opera sobre um operando).

Binários	
=	Atribuição
+	Soma
-	Subtração
*****	Multiplicação
/	Divisão
%	Módulo (resto da divisão)
Unário	
-	Menos unário

O uso de parênteses altera a ordem de prioridade das operações.

Ex:

$$(a + b)* 80 \neq a + b * 80$$

Operadores Aritméticos

- C tem vários operadores que permitem comprimir comandos.

variável **op**= expressão; variável = (variável) **op** (expressão);

Operador	Exemplos		
++	i++;	equivale a	i = i +1;
--	i--;	equivale a	i = i – 1;
+=	i += 2;	equivale a	i = i + 2;
-=	d -= 3;	equivale a	d = d – 3;
***=**	x *= y+1;	equivale a	x = x*(y+1);
/=	t /= 2.5;	equivale a	t = t/2.5;
%=	p %= 5;	equivale a	p = p%5;

Operadores Relacionais

Operador	Nome	Exemplo	Significado do Exemplo
==	**Igualdade**	`a == b`	a é **igual** a b?
>	**Maior que**	`a > b`	a é **maior** que b?
>=	**Maior ou Igual que**	`a >= b`	a é **maior ou igual a** b?
<	**Menor que**	`a < b`	a é **menor** que b?
<=	**Menor ou Igual que**	`a <= b`	a é **menor ou igual a** b?
!=	**Diferente de**	`a != b`	a é **diferente** de b?

Operadores Lógicos

Operador	Significado	Exemplo
&&	AND (E lógico)	`x>=1 && x<=19`
||	OR (OU lógico)	`x==1 || x ==2`
!	NOT (Negação lógica)	`! Continuar`

Operadores de Ponteiros

- Um ponteiro é um endereço na memória de uma variável
- Uma variável de ponteiro é uma variável especialmente declarada para guardar um ponteiro para seu tipo especificado
- Operador &: É um operador unário que devolve o endereço na memória de seu operando
- Operador *: Devolve o valor da variável localizada no endereço que o segue

Exemplo

```
#include <stdio.h>

void main(void)
{
  int target, source;
  int *m;

  source = 10;
  m = &source;
  target = *m;

  printf("%d", target);
}
```

Operador sizeof

- O operador sizeof retorna o tamanho, em bytes, da variável ou especificador de tipo, em parênteses

```
float f;

printf("%f", sizeof f);
printf("%d", sizeof (int));
```

Precedência

- Indica qual operação é realizada primeiro
  - Quanto menor a precedência, mais prioridade na realização da operação

- Exemplo:

Precedência	Operador
1	- unário
2	* / %
3	+ -

Associatividade

- Indica em qual ordem a operação deve ser aplicada
  - À Esquerda (→): a ~ b ~ c significa que (a ~ b) ~ c
  - À Direita (←): a ~ b ~ c significa que a ~ (b ~ c)

- Soma, subtração, multiplicação são associativas à esquerda

- Exponenciação e atribuição são associativas à direita

- Existem certos operadores em determinadas linguagens de programação que não são associativos nem à direita nem à esquerda

Tabela de Precedências

Operador	Associatividade
`( )` `[ ]` `->` `.`	→
`!` `~` `++` `--` `-` *(unário)* `+` *(unário)* `(casting)` `*`*(apontado)* `&` `sizeof`	←
`*` `/` `%`	→
`+` `-`	→
`<<` `>>`	→
`<` `<=` `>` `>=`	→
`==` `!=`	→
`&`	→
`^`	→
`|`	→
`&&`	→
`||`	→
`?` `:` *(Operador Ternário)*	←
`=` `+=` `-=` `*=` `/=` `%=` `<<=` `>>=` `&=` `^=` `|=`	←
`,` *(Vírgula)*	→

Nota: As precedências aumentam no sentido da seta.

Constantes

- Variáveis do tipo **const** recebem um valor inicial e não podem ser modificadas ao longo do programa

- Exemplo:
  - float PI= 3.1415;

- Utilizando o #define antes do main
  - #define PI 3.1415

Constantes

```c
#include <stdio.h>

#define ZERO 0

int main(){

    const float PI = 3.14;

    printf("O valor de zero eh %d e o de pi eh %g",ZERO,PI);

    return 0;
}
```

Expressões

- Uma expressão em C é qualquer combinação valida de operadores, constantes e variáveis

- Regras gerais da álgebra, com poucos aspectos particulares da linguagem

- x = f1() + f2();
  - O padrão ANSI C não estipula qual a ordem de execução
  - Depende do compilador!

Conversão de Tipos em Expressões

- Quando constantes e variáveis de tipos diferentes são misturadas em uma expressão, elas são convertidas a um do mesmo tipo

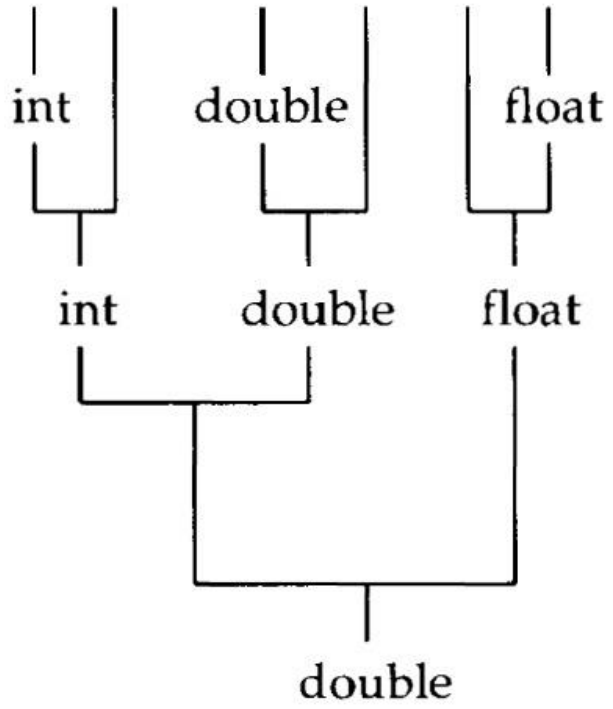
- *Promoção de tipo*
  - O compilador C converte todos os operandos de uma expressão no tipo do maior operando
  - “Maior operando” segue um conjunto de regras!
  - Há, entretanto, algumas exceções que devem ser verificadas
    - Schildt – Cap. 2 – p. 57

Conversão de Tipos em Expressões

SE um operando é **long double**
ENTÃO o segundo é convertido para **long double**
SENÃO, SE um operando é **double**
ENTÃO o segundo é convertido para **double**
SENÃO, SE um operando é **float**
ENTÃO o segundo é convertido para **float**
SENÃO, SE um operando é **unsigned long**
ENTÃO o segundo é convertido para **unsigned long**
SENÃO, SE um operando é **long**
ENTÃO o segundo é convertido para **long**
SENÃO, SE um operando é **unsigned int**
ENTÃO o segundo é convertido para **unsigned int**

Conversão de Tipos em Expressões

```
char ch;
int i;
float f;
double d;
result = (ch/i)  +  (f*d)  -  (f+i);
```



Cast

- Cast é uma operação que força um expressão a ser de um determinado tipo

- Forma genérica
  - (tipo) expressão

```
#include <stdio.h>

void main(void) /* imprime i e i/2 com frações */
{
  int i;

  for (i=1; i<=100; ++i)
    printf("%d / 2 é: %f\n", i, (float) i /2);
}
```