MC-102 — Aula 05
Expressões Relacionais, Lógicas e Comandos Condicionais

Instituto de Computação – Unicamp

13 de Março de 2012

Roteiro

Expressão

- Já vimos que constantes e variáveis são expressões.

Exemplo

```
a = 10;
a = b;
```

- Vimos também que operações aritméticas também são expressões.

Exemplo

```
a = 2 + 2;
a = 10 / (float) 3;
a = a + 1;
```

Expressões relacionais

Expressões relacionais são aquelas que realizam uma comparação entre duas expressões e retornam

1. Zero (0), se o resultado é falso
2. Um (1), ou qualquer outro número diferente de zero, se o resultado é verdadeiro.

Operadores Relacionais

Os operadores relacionais são:

- $==$: igualdade.
- $!=$: diferente.
- $>$: maior que.
- $<$: menor que.
- $>=$: maior ou igual que.
- $<=$: menor ou igual que.

Expressões relacionais

- $<$ *expressao* $>$ $==$ $<$ *expressao* $>$: Retorna verdadeiro quando as expressões forem iguais.
  Ex: `a == b`
- $<$ *expressao* $>$ $!=$ $<$ *expressao* $>$: Retorna verdadeiro quando as expressões forem diferentes.
  Ex: `a != b`

Expressões relacionais

- $<$ *expressao* $>$ `>` $<$ *expressao* $>$: Retorna verdadeiro quando a expressão da esquerda tiver valor maior que a expressão da direita.
  Ex: `a > b`
- $<$ *expressao* $>$ `<` $<$ *expressao* $>$: Retorna verdadeiro quando a expressão da esquerda tiver valor menor que a expressão da direita.
  Ex: `a < b`

Expressões relacionais

- $<$ *expressao* $>$ `>=` $<$ *expressao* $>$: Retorna verdadeiro quando a expressão da esquerda tiver valor maior ou igual que a expressão da direita.
  Ex: `a >= b`
- $<$ *expressao* $>$ `<=` $<$ *expressao* $>$: Retorna verdadeiro quando a expressão da esquerda tiver valor menor ou igual que a expressão da direita.
  Ex: `a <= b`

Expressões lógicas

Expressões lógicas são aquelas que realizam uma operação lógica (`ou`, `e`, `não`, etc...) e retornam verdadeiro ou falso (como as expressões relacionais).

Operadores Lógicos

- &&: operador E.
- ||: operador OU.
- !: operador NÃO.

Expressões lógicas

- $< expressao > \&\& < expressao >$: Retorna verdadeiro quando ambas as expressões são verdadeiras. Sua tabela verdade é:

Op_1	Op_2	Ret
V	V	V
V	F	F
F	V	F
F	F	F

Exemplo

```
a == 0 && b == 0
```

Expressões lógicas

- $<$ *expressao* $>$ || $<$ *expressao* $>$: Retorna verdadeiro quando pelo menos uma das expressões é verdadeiras. Sua tabela verdade é:

Op_1	Op_2	Ret
V	V	V
V	F	V
F	V	V
F	F	F

Exemplo

```
a == 0 || b == 0
```

Expressões lógicas

- ! $<$ *expressao* $>$: Retorna verdadeiro quando a expressão é falsa e vice-versa. Sua tabela verdade é:

Op_1	Ret
V	F
F	V

Exemplo

```
!(a == 0)
```

Simplificações úteis

- `!(a == b)` é equivalente a: `(a != b)`
- `!(a != b)` é equivalente a: `(a == b)`
- `!(a > b)` é equivalente a: `(a <= b)`
- `!(a < b)` é equivalente a: `(a >= b)`
- `!(a >= b)` é equivalente a: `(a < b)`
- `!(a <= b)` é equivalente a: `(a > b)`

Comandos condicionais

Um comando condicional é aquele que permite decidir se um determinado bloco de comandos deve ou não ser executado, a partir do resultado de uma expressão relacional ou lógica.

Bloco de comandos

- É um conjunto de instruções agrupadas.
- Limitada pelos caracteres { e }.

Exemplo

```
main(void)
{       ⟵ Início do bloco de comandos
 int a;
 a=1;
}       ⟵ Fim do bloco de comandos
```

Comandos condicionais

- O principal comando condicional da linguagem C é o `if`, cuja sintaxe é:

```
if (expressão lógica)
    comando;   ou
if (expressão lógica) {
    comandos
}
```

- Os comandos são executados somente se a expressão lógica for verdadeira.

Comandos condicionais

O programa abaixo determina se um valor é ímpar.

```
#include <stdio.h>

int  main () {
    int a;
    scanf("%d", &a);
    if ((a % 2) != 0) {
       printf ("O valor é ímpar.\n");
    }
  }
```

Comandos condicionais

Lembrando como C representa os valores Falso e Verdadeiro, o programa pode ser alterado para:

```
#include <stdio.h>

int  main () {
    int a;
    scanf("%d", &a);
    if (a % 2) {
       printf ("O valor é ímpar.\n");
    }
  }
```

Comandos condicionais

- Uma variação do comando `if` é o `if/else`, cuja sintaxe é:

```
if (expressão lógica) {
    comandos executados se a expressão é verdadeira
} else {
    comandos executados se a expressão é falsa
}
```

Comandos condicionais

Determinando o menor de dois números:

```
int main(void){
    int a,b;

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

    if(a < b){
        printf("O menor numero e: %d\n", a);
     }else{
        printf("O menor numero e:%d\n",b);
     }


}
```

Comandos condicionais

```
if (cond1)
  if (cond2)
    comando1;
else
  comando2;
```

Quando o comando2 é executado?

Comandos condicionais

```
if (cond1)
  if (cond2)
    comando1;
  else
    comando2;
```

Quando o comando2 é executado?

Comandos condicionais

```
if (cond1) {
  if (cond2)
    comando1;
} else
  comando2;
```

Quando o comando2 é executado?

if-else-if Encaixados

- Uma coisa muito comum em programação é o teste de várias alternativas.
- Podemos usar uma construção simples com ifs:

```
int main () {
  int ra;
  scanf("%d", &ra);
  if (ra == 10129)
     printf("Maria Cândida Moreira Telles\n");
  if (ra == 33860)
     printf("Larissa Garcia Alfonsi\n");
  if(....
  ....
}
```

if-else-if Encaixados

- Porém todos os testes condicionais serão executados!!
- Quando apenas uma de várias alternativas é verdadeira podemos usar a construção if-else-if:

```
int main () {
  int a;
  scanf("%d", &a);
  if (a == 10129)
     printf("Maria Cândida Moreira Telles\n");
  else if (a == 33860)
     printf("Larissa Garcia Alfonsi\n");
  else if (....
   ....
  else
     printf("Nenhum aluno com RA informado!");
}
```

if-else-if Encaixados

- Na construção if-else-if quando uma condição é verdadeira, o bloco de comandos correspondente será executado.
- Após a execução do bloco de comandos as outras alternativas não serão testadas.
- O último **else** (sem if) pode ser utilizado como uma opção padrão quando nenhuma das condições dos ifs é verdadeira.

O comando switch

- O objetivo do comando `switch` é simplificar uma expressão onde as condições ocorrem sobre uma variável inteira ou caracter:

Sintaxe

```
switch (variável inteira) {
    case valor:  comandos
    break;
    case valor:  comandos
    break;
}
```

O comando switch

```
switch(a) {
case 10129:
    printf("Maria Cândida Moreira Telles\n");
    break;
case 33860:
    printf("Larissa Garcia Alfonsi\n");
    break;
case 33967:
    printf("Leonardo Kozlowiski Kenupp\n");
    break;
}
```

O comando switch

- Os comandos começam a ser executados a partir do ponto onde o valor da variável corresponde ao valor antes dos dois pontos (:).
- Executa todos os comandos até que encontre um comando `break` ou que chegue ao final do bloco de comandos do `switch`

Valor padrão

- Você pode utilizar uma condição `default`. A execução dentro da alternativa `default` ocorre se nenhuma outra condição foi verdadeira (assim como o último else do if-else-if encaixados).

Sintaxe

```
switch (variável inteira) {
    valor:  comandos break;
    default:  comandos
}
```

Valor padrão

```
switch(a) {
case 10129:
    printf("Maria Cândida Moreira Telles\n");
    break;
case 33860:
    printf("Larissa Garcia Alfonsi\n");
    break;
default:
    printf("O aluno não está matriculado\n");
}
```

Exercícios

- Escreva um programa que lê um número inteiro do teclado e imprime "SIM"se o número for par e maior do que 10, ou for ímpar e menor do que 50. Caso contrário o programa deve imprimir "NAO".

Exercícios

- Escreva um programa lê três números e imprime o maior deles.

Exercícios

- Escreva um programa lê três números e imprime em ordem (ordem decrescente).