Universidade de Caxias do Sul (UCS)
Área de Ciências Exatas e Tecnologia
Disciplina: Programação de Computadores I

Linguagem de Programação C

Comando de Repetição For

Prof. Dr. Ricardo Vargas Dorneles

Comando For

- O comando for é um comando de repetição do tipo "enquanto", que contém a inicialização e o incremento da variável de controle

Comando For

for (inicialização; condição;incremento)

comando;

- onde **comando** é um comando simples ou um grupo de comandos, agrupados com chaves
- **Inicialização** é um conjunto de atribuições iniciais (pode ser mais de uma, separadas por vírgulas)
- **Condição** é a expressão lógica que, enquanto for verdadeira, faz com que o comando continue sendo executado.
- **Incremento** é um conjunto de comandos que será executado ao final de cada iteração

Equivalência entre o "while" e o "for"

```
i=0;
while (i<10){
    comando
    i=i+1;
}
```

```
for (i=0;i<10;i=i+1)
    comando;
```

Observações

- Os campos de inicialização, condição e incrementos são separados entre si por ponto-e-vírgula
- O campo **inicialização** do cabeçalho do for pode conter mais de uma inicialização
  - Nesse caso, as inicializações são separadas por vírgulas
  - O mesmo ocorre nos incrementos
  - Ex: *for (i=0,j=0; j<10; j=j+1,i=i+1)...*

Observações

- O cabeçalho do for não precisa obrigatoriamente conter inicializações ou incrementos
  - Nesse caso as listas de inicializações e incrementos ficam vazias, mas os ponto-e-vírgula entre os campos de inicializações, condição e incrementos são obrigatórios
  - Ex: *for (;i<10;) ...*

Exemplos

- Um comando de inicialização e um incremento:

```
for (i=0 ; i<10 ; i=i+1)
/* para i de 0, ENQUANTO i for menor que 10, faca */
    printf ( "%d \n " , i ) ;
```

- Dois comandos de inicialização e dois de incremento:

```
for (i=10 , j=1 ; i<10 ; i=i+1 , j=j-1 )
    printf ( "i = %d j = %d nn " , i , j ) ;
```

Exemplos

- Com mais de um comando na lista de comandos:

```
for (soma=0 , i=0 ; i<10 ; i++){
    soma=soma+i;
    printf("soma = %d \n",soma);
}
```

Laços infinitos

- A expressão de condição do comando for pode ser omitida (deixada em branco). Nesse caso, a condição é avaliada como verdadeira.
- O for a seguir implementa um "laço infinito":

```
for (;;)
    {
      //faz alguma coisa infinitamente ou até ser executado um break
    }
```

- Uma outra alternativa para um laço infinito é

  for (;1;) ou *while(1)*

- já que em C o valor 0 é interpretado como falso, e qualquer valor diferente de 0 é interpretado como verdadeiro.

Comando For

- Cada repetição dos comandos dentro do comando for também é e chamada **iteração** (dá o nome de "**programas iterativos**")

Erro Comum nº 1

- Colocar um ponto-e-vírgula na linha do for:

```
for (i=0;i<10;i=i+1);
    printf("%d\n",i);
```

- Assim como no ***if***, esse ponto-e-vírgula faz com que o comando ***for*** termine, de modo que o comando seguinte está fora do for e será executado só uma vez.

Erro Comum nº 2

- Não colocar chaves no grupo de comandos:

```
for (i=0,soma=0;i<10;i=i+1)
    soma=soma+i;
    printf("soma=%d\n",soma);
```

- Nesse exemplo, apenas o primeiro comando (*soma=soma+i*) está dentro do for e será repetido

Erro Comum nº 3

- Tratar a condição do for como uma condição de repita (até que...) ao invés de uma condição de enquanto. Ex.

```
for (i=0;i==10;i=i+1)
// pensando que a condição é ATÉ QUE i=10
    printf("%d\n",i);
```

Pós-incremento e Pré-incremento

- *i = i +1* pode ser escrito como *i++* ou *++i*
- *i = i -1* pode ser escrito como *i- -* ou *- -i*
- A diferença entre o *++i* (pré-incremento) e o *i++* (pós-incremento) é que, quando ocorre dentro de uma expressão, o valor utilizado na expressão é o valor antes do incremento ou depois do incremento.
- *x=i++;* // o incremento de i é feito após a atribuição
- *x=++i;* // o incremento de i é feito antes da atribuição

Formas Reduzidas de Atribuição

- Quando um dos operandos de uma operação é a variável que receberá o resultado (p.ex: `a = a + 3;`) pode-se omitir o operando da expressão, resultando o comando `a+=3;`
- Da mesma forma pode-se utilizar essa versão reduzida para os operadores de subtração, divisão e multiplicação:
  - `a*=3;` equivale a `a=a*3;`
  - `a-=3; a=a-3;`
  - `a/=3; a=a/3;`

Formas Reduzidas de Atribuição

- Exemplo de um laço para escrever os números de 1 a 10:

```
for ( i = 1 ; i <= 10 ; i+=1 )
printf ("%d\n",i);
```

Comando *Break*

- O comando ***break*** é utilizado quando se deseja interromper a execução das repetições, sem completá-las
- A execução do comando ***break*** faz com que a execução vá para o comando seguinte ao comando for, sem completar a iteração corrente.
- Exemplo:

```
for (i=0;i <10; i++)
{
scanf ("%d" ,&a);
if (a ==0) break ;   /* faz com que saia do for se for digitado 0 */
}
```

Comando *Continue*

- O comando ***continue*** é utilizado quando se deseja interromper a execução da iteração atual, mas deseja-se continuar executando as próximas iterações do for

Uso de valores numéricos como valores lógicos

- A linguagem C permite utilizar valores numéricos como valores lógicos (verdadeiro ou falso)
- Nesse sentido, o valor 0 representa o valor lógico falso, e um valor diferente de 0 representa o valor lógico verdadeiro
- Exemplo de um programa que utiliza o comando for. Neste laço é utilizado um valor lógico para o laço rodar indefinidamente até ser executado um comando break:

```
for (;1;)
{
/* comandos */
}
```