

# MC-102 — Aula 07

## Comandos de Repetição I: `while` e `do-while`

Instituto de Computação – Unicamp

Primeiro Semestre de 2006

# Roteiro

- 1 Introdução
- 2 while (condicao) { comandos }
- 3 do { comandos } while (condicao);
- 4 Exemplos

# Introdução

- Até agora, vimos como escrever programas capazes de executar comandos de forma linear, e, se necessário, tomar decisões com relação a executar ou não um bloco de comandos.
- Entretanto, eventualmente é necessário executar um bloco de comandos várias vezes para obter o resultado.

## Exemplo

Calcule a divisão inteira de dois numeros usando apenas soma e subtração

# Solução

- Duas variáveis: temporario, contador
  - ① temporario=dividendo;
  - ② contador=0;
  - ③ Enquanto *temporario > divisor*
    - ① temporario = temporario - divisor
    - ② contador = contador + 1
  - ④ Exiba contador

Por que?

Contador equivale a divisão inteira de dividendo por divisor

# Introdução

- Será que dá pra fazer com o que já temos?
- Ex.: Programa que imprime todos os números de 1 a 4

```
printf("1");  
printf("2");  
printf("3");  
printf("4");
```

# Introdução

- Ex.: Programa que imprime todos os números de 1 a 100

```
printf("1");  
printf("2");  
printf("3");  
printf("4");  
/*repete 95 vezes a linha acima*/  
printf("100");
```

# Introdução

- Ex.: Programa que imprime todos os números de 1 a n (dado)

```
printf("1");  
if (n>=2)  
    printf("2");  
if (n>=3)  
    printf("3");  
/*repete 96 vezes o bloco acima*/  
if (n>=100)  
    printf("100");
```

# Introdução

- Ex.: Programa que imprime 2 elevado a todos os números de 1 a n (dado)

```
printf("2^1 = 2");  
if (n>=2)  
    printf("2^2 = 4");  
/*repete 97 vezes o bloco acima*/  
if (n>=100)  
    printf("2^100 = ???");
```

# Introdução

- Ex.: Programa que imprime 2 elevado a todos os números de 1 a n (dado)

```
int i=1, pot=2;  
printf("2^%d = %d",i,pot);  
i++; pot *=2;  
if (n>=2) {  
    printf("2^%d = %d",i,pot);  
    i++; pot *=2; }  
/*repete 97 vezes o bloco acima*/  
if (n>=100) {  
    printf("2^%d = %d",i,pot);  
    i++; pot *=2; }
```

# Introdução

- Ex.: Programa que imprime 2 elevado a todos os números de 1 a n (dado)

```
int i=1,pot=2;  
if (i<=n) {  
    printf("2^%d = %d",i,pot);  
    i++; pot *=2; }  
/*repete 98 vezes o bloco acima*/  
if (i<=n) {  
    printf("2^%d = %d",i,pot);  
    i++; pot *=2; }
```

# Introdução

- Reparem, no exemplo anterior, que o trecho abaixo é executado 100 vezes

```
if (i<=n)  
{  
    printf("2^%d = %d",i,pot);  
    i++;  
    pot *=2;  
}
```

# Introdução

- Para cada comparação, fazemos:
  - 1 imprimimos o expoente e sua potência.
  - 2 incrementamos o expoente
  - 3 multiplicamos a potência
- Quando  $i$  supera  $n$ , todas as demais comparações retornam falso, e não são executadas.

## Problema

$n$  é limitado ao tamanho do nosso código.

# Introdução

- Seria interessante fazer com que o código repetisse a comparação e executasse o comando dentro até que a condição fosse falsa

```
/* Enquanto for verdade que  $i \leq n$ , execute */  
{  
    printf("2^%d = %d", i, pot);  
    i++; pot *= 2;  
}
```

# while (condicao) { comandos }

- Estrutura:

```
while ( condicao ) comando;  
while ( condicao ) { comandos }
```

- Enquanto a condição for verdadeira ( $\neq 0$ ), ele executa o(s) comando(s);

## Imprimindo os 100 primeiros números inteiros

```
int i=1;  
while (i<=100)  
{  
    printf("%d ",i);  
    i++;  
}
```

# Imprimindo os $n$ primeiros números inteiros

```
int i=1,n;  
scanf("%d",&n);  
while (i<=n)  
{  
    printf("%d ",i);  
    i++;  
}
```

## Imprimindo as $n$ primeiras potências de 2

```
int i=1,n,pot=2;  
scanf("%d",&n);  
while (i<=n)  
{  
    printf("2^%d = %d ",i,pot);  
    i++;  
    pot*=2;  
}
```

# while (condicao) { comandos }

- 1. O que acontece se a condição for falsa na primeira vez?  
`while (a!=a) a=a+1;`
- 2. O que acontece se a condição for sempre verdadeira?  
`while (a==a) a=a+1;`

# while (condicao) { comandos }

- 1. O que acontece se a condição for falsa na primeira vez?

```
while (a!=a) a=a+1;
```

R: Ele nunca entra na repetição (*loop*).

- 2. O que acontece se a condição for sempre verdadeira?

```
while (a==a) a=a+1;
```

R: Ele entra na repetição e nunca sai (*loop* infinito).

# while (condicao) { comandos }

- Estudando a estrutura “normal” do while mais a fundo.

```
while (i<=n) ← condição de repetição  
{  
    printf(“%d ”,i);  
    i++; ← Comando de passo  
}
```

- O oposto (negação) da condição de repetição é conhecida como condição de parada:  
     $!(i \leq n) \Rightarrow i > n$  é a condição de parada.

# while (condicao) { comandos }

- *loop* de fim determinado

```
scanf("%d",&preco);  
while (i<=n) {  
    total = total + preco;  
    i++;  
    scanf("%d",&preco);  
}
```

# while (condicao) { comandos }

- *loop* de fim indeterminado

```
scanf("%d",&preco);  
while (preco>0) {  
    total = total + preco;  
    scanf("%d",&preco);  
}
```

```
do { comandos } while (condicao);}
```

- Estrutura:

```
do comando; while ( condicao );  
do { comandos } while ( condicao );
```

- Diferença do while: Sempre entra na primeira vez

## MDC(x,y)

Supondo, sem perda de generalidade, que  $x \geq y$ , o  $\text{MDC}(x, y)$  é definido da seguinte forma:

$$\text{MDC}(x, y) = \begin{cases} y & \text{caso } x \bmod y = 0 \\ \text{MDC}(y, x \bmod y) & \text{caso contrário} \end{cases}$$

### Exercício

Complete o programa em `mdc.c`

# MDC(x,y)

```
x = maior;  
y = menor;  
do  
{  
    r = x % y;  
    x = y;  
    y = r;  
} while (r!=0);
```

- Repare que  $r$  só é calculado dentro do *loop*
- Veja exemplo em `mdc-completo.c`

# Soma de n valores inteiros

```
soma = 0;
while (n > 0) {
    printf("número a ser somado: ");
    scanf("%d", &parcela);
    soma += parcela;
    n--;
}
printf("Soma: %d\n", soma);
```

- Veja exemplo em soma-n.c

# Soma até 0

```
soma = 0;
printf("número a ser somado (0 para sair): ");
scanf("%d", &parcela);

while (parcela != 0) {
    soma += parcela;
    printf("número a ser somado (0 para sair): ");
    scanf("%d", &parcela);
}

printf("Soma: %d\n", soma);
```

- Veja exemplo em soma-ate-0.c

## Soma até 0

```
soma = 0;  
  
do {  
    printf("número a ser somado (0 para sair): ");  
    scanf("%d", &parcela);  
    soma += parcela;  
} while (parcela != 0);  
  
printf("Soma: %d\n", soma);
```

- Veja exemplo em `soma-ate-0-do-while.c`

# Arte em ASCII

Como imprimir uma linha de '\*'s

```
*****
```

- Veja exemplo em `linha.c`

# Arte em ASCII

[illegible]

- Veja exemplo em desenho.c

# Arte em ASCII

```
*****  
*****  
***  
**  
*  
*  
**  
***  
****  
*****
```

- Veja exemplo em `desenho2.c`

# Arte em ASCII

```
*****  
*****  *****  
*****  *****  
***      ***  
**       **  
*        *  
*        *  
**       **  
***      ***  
*****  *****  
*****  *****  
*****
```

- Veja exemplo em `desenho3.c`

# Arte em ASCII

```
  *
 ***
*****
*****
*****
*****
*****
*****
***
 *
```

- Veja exemplo em `desenho4.c`