Comandos de Repetição

Profa. Elloá B. Guedes
www.elloaguedes.com

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

Comando de Repetição while

- while = enquanto

```
<inicialização das variáveis de iteração>
while (<condição de permanência no laço>){
    // bloco de comandos a serem repetidos
   <atualização da variável de iteração>
}
```

Comando de Repetição while

```
// Program to introduce the while statement

#include <stdio.h>

int main (void)
{
      int  count = 1;

      while ( count <= 5 ) {
            printf ("%i\n", count);
            ++count;
      }

      return 0;
}
```

Comando de Repetição for

- Estrutura geral

```
for (<inicialização de variável de iteração>; <condição de permanência no laço>;
<incremento da variável de iteração>){
        // bloco de comandos a serem repetidos
 }
```

Comparação entre for e while

- Há uma semelhança na funcionalidade dos comandos for e while

- De fato, onde há um comando do tipo for, é possível reescrevê-lo utilizando um while e vice-versa

```
for ( init_expression;  loop_condition;  loop_expression )
     program statement
```

```
init_expression;
while ( loop_condition ) {
     program statement
     loop_expression;
}
```

Comando de Repetição do-while

- Nos comandos for e while a condição de entrada no laço é testada sempre antes
  - Isso pode fazer com que em determinada situações o laço nunca seja executado

- Alterantiva: comando do-while

```
do{
  // comandos
} while (<condição>)
```

- O laço do-while repete enquanto a condição é verdadeira

Exercício

- Escreva um programa para ler um número inteiro qualquer e determinar todos os seus divisores exatos.

Exercício

- Escreva um programa que inverta os caracteres de um número

- Casos de teste

  Entrada: 13579, Saída: 97531

  Entrada: 12, Saída 21

Exercício

- Escreva um programa que calcule a média de uma sequência de números reais positivos inseridos pelo usuário. A entrada de dados termina quando o número digitado for negativo.

Exercício

- Escreva um programa para ler um valor inteiro *k* e calcular a média aritmética de *k* valores reais lidos da unidade padrão de entrada. O programa deve imprimir o fatorial de cada número par positivo fornecido.

Exercício

- Escreva um programa que calcule o máximo divisor comum entre dois números.

- Entrada: u, v
- Passo 1: Se v == 0, então o máximo divisor comum é u
- Passo 2: Calcule temp =u %v

u = v;

v = temp;

Volte ao passo 1

Casos de Teste:

Entrada: 150 e 35, Saída: 5

Entrada: 1026 e 405, Saída: 27

Exercício

- Escreva um programa para ler dois números inteiros *m* e *n* e, a seguir, imprimir os números pares existentes no intervalo [*m*, *n*].

Exercícios

- Escreva um programa para calcular uma aproximação para *exp(x)*, onde *x* é um número qualquer lido da unidade padrão de entrada. A aproximação pode ser obtida de:

- exp(x) = 1 + x + x/2! + ... + x/i!

- O programa deve encerrar o processamento quando a variação no valor calculado for inferior a 0.0001.

Exercícios

- Escreva um programa para gerar os cinquenta primeiros termos da série:

- 1, 1, 2, 4, 3, 9, 4, 16, 5, 25, 6, 36, ...

Exercícios

- Escreva um programa para gerar os **quinze** primeiros termos da série de FIBONACCI:

- 1, 1, 2, 3, 5, 8, 13, ...

Exercícios

- Faça um algoritmo para ler diversos caracteres informados pelo dispositivo de entrada. Depois imprima:

  a) a quantidade total de letras 'A' e 'Z' informadas;

  b) a quantidade de caracteres informados;

  c) a quantidade de consoantes;

  d) a maior letra informada (de acordo com a ordem alfabética) (considere que o usuário irá digitar todas as letras em caixa alta);

  e) a quantidade de pontos de exclamação informados;

- A condição de término da leitura é o caractere '#'.

Resumo

	while	**for**	**do ... while**
Sintaxe	while (cond) instrução	for (carga inic; cond ; pos-inst) instrução	do instrução while (condição)
Executa a instrução	zero ou mais vezes	zero ou mais vezes	1 ou mais vezes
Testa a condição	antes da instrução	antes da instrução	depois da instrução
Utilização	freqüente	freqüente	pouco freqüente

Comandos de Desvio

- return
  - Usado para retornar de uma função
  - Programa é construído de maneira modular

- goto
  - Faz com que o fluxo da execução seja direcionado para um ponto específico
  - Não é considerada uma boa prática de programação!

```
x = 1;
loop1:
  x++;
  if(x<100) goto loop1;
```

Comandos de Desvio

- break
  - Possui dois usos: no comando switch-case
  - Terminação imediata de um laço

```
#include <stdio.h>

void main(void)
{
  int t;

  for(t=0; t<100; t++) {
    printf("%d ", t);
    if(t==10) break;
  }
}
```

Comandos de Desvio

- exit()
  - Função da biblioteca padrão de C
  - Provoca uma terminação imediata do programa inteiro, forçando um retorno ao sistema operacional
  - Mesma ideia de funcionamento que o return na função main
  - Códigos de erro podem ser passados, indicando o que causou a interrupção do programa

Comandos de Desvio

- continue
  - Força que ocorra a próxima iteração do laço, pulando qualquer código intermediário

```
void code(void)
{
  char done, ch;

  done = 0;
  while(!done) {
    ch = getchar();
    if(ch=='$') {
      done = 1;
      continue;
    }
    putchar(ch+1);   /* desloca o alfabeto uma posição */
  }
}
```

Laços Infinitos

- Denominam-se laços infinitos aqueles que quando são executados nunca terminam, isto é, apresentam condições que são sempre verdadeiras

```
While (1)
  instrução;
```

```
for ( ; ; )
  instrução;
```

```
do
  instrução;
while (1)
```

Diferenças entre ++x e x++

y = x++;	y = ++x;
Acontecem duas coisas, nessa ordem:	Acontecem duas coisas, nessa ordem:
1. O valor de x é atribuído a y	1. O valor de x é incrementado
2. O valor de x é incrementado	2. O valor de x é atribuído a y

Quando o operador de incremento ou decremento está antes da variável, esta é operada antes de ser usada.

Quando o operador de incremento ou decremento está depois da variável, esta é usada e só depois é incrementada ou decrementada.

Diferenças entre ++x e x++

`x=5;` `y=x++;`	`x=5;` `y=++x;`
Coloca o valor 5 na variável y. *Em seguida incrementa a variável x.*	*Incrementa o valor de x.* *Em seguida coloca o valor x na variável y.*
Valores finais: x ➔ 6 e y ➔ 5	Valores finais: x ➔ 6 e y ➔ 6

Como se pode observar, o valor final das variáveis não é o mesmo.

Dessa forma, verificam-se as seguintes equivalências:

y = x++;	é equivalente a	y = x; x++;

y = ++x;	é equivalente a	x++; y = x;

Diferenças entre ++x e x++

```c
#include <stdio.h>
#include <math.h>

int main(){

    int x = 2;

    printf("%f",pow(++x,2));
    printf("\n%f",pow(x++,2));
    printf("\n%d",x);


    return 0;
}
```