FATEC - SP

Faculdade de Tecnologia de São Paulo

Prof° Luiz Paulo Zanetti

E-mail: luizpaulozanetti@hotmail.com

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Disciplina
Linguagem de Programação

FOR = PARA

for

O loop for em C é muito mais forte e mais flexível que o da maioria das outras linguagens. Sua forma geral é

for (inicialização; condição; incremento)

Observe que as três partes do loop for são separadas por ponto e vírgula.

Nenhuma destas partes precisa existir.

FOR = PARA

ATRIBUIÇÃO | COMPARAÇÃO | INCREMENTO OU DECREMENTO

```
for ( a=0 ; a<=10 ; a++ )
```

ATRIBUIÇÃO	COMPARAÇÃO	INCREMENTO OU DECREMENTO
a=b	a!=b	a--
a=a+2	a>=b	a=a+2
a=a-b	a==b	a=a-2

FOR = PARA

Inicialização

Na forma mais simples, inicialização é um comando de atribuição que o compilador usa para estabelecer a variável de controle de loop.

A inicialização pode conter qualquer comando válido em C.

FOR = PARA

Condição

A condição é uma expressão de relação que testa se a condição final desejada pelo loop for ocorreu.

Aqui também pode ser colocado qualquer comando válido em C.

FOR = PARA

Incremento

O incremento define a maneira como a variável de controle do loop será alterada cada vez que o computador repetir o loop.

Também aqui, podemos colocar qualquer comando válido em C.

FOR = PARA

Exemplo - 01

```
/* imprimir os números de 1 a 100 */
main()
{
   int x;
   for (x = 1; x <= 100; x ++)
   {
   printf("%d ", x);
   }
}
```

FOR = PARA

Exemplo - 02

```
/* imprimir os números de 100 a 1 */
main()
{
   int x;
   for (x = 100; x > 0; x --)
   {
   printf("%d ", x);
   }
}
```

FOR = PARA

Exemplo - 03

```
/* imprimir os números de 0 a 100, 5 em 5 */
main()
{
   int x;
   for (x = 0; x <= 100; x = x + 5)
   {
   printf("%d ", x);
   }
}
```

FOR = PARA

Exemplo - 04

```
/* executa um bloco de código 100 vezes */  main()
{
   int x;
   for (x = 0; x < 100; x ++)
   {
   printf("O valor de x é: %d ", x);
   printf("e o quadrado de x é: %d\n", x* x);
   }
 }
```

FOR = PARA

Variações do loop for

Podem ser executados mais de um comando nas partes de inicialização e de incremento. Veja que:

```
main()
{
    int x, y;
    for (x = 0, y = 0; x + y < 100; ++x, y++)
    {
    printf("%d ", x + y);
    }
}
```

Mostrará números de 0 a 98, 2 a 2.

NOTA ! ! !
++ x incrementa x e retorna o número incrementado, enquanto x ++ retorna x e, em seguida, incrementa-o

FOR = PARA

Um uso diferente para for

```
main()
{
    int t;
    for (prompt(); t=readnum(); prompt()) sqrnum(t);
}
prompt()
{
printf("digite um inteiro:" );
}

readnum()
{   int t;  scanf("%d", &t);  return t;
}

sqrnum(int num)
{   printf("%d\n", num * num);
}
```

FOR = PARA

Loop infinito

Podemos fazer um comando for executar para sempre simplesmente não especificando sua parte condicional. Veja:

```
for (;;)
{
printf("este loop rodará para sempre\n");
}
```

FOR = PARA

Saindo de um loop

Podemos usar o comando break para encerrar um for a qualquer momento. Veja um exemplo:

```
main()
{
int a;
for (a = 1; a < 100; a++)  if (a == 10) break;
}
```

O loop só será executado 10 vezes.

FOR = PARA

Loops for sem nenhum corpo

Podem ser utilizados loops sem corpo para gerar retardo de tempo.

Veja um exemplo:

```
for (a = 0; a < 1000; a ++)
```

Sleep

A função sleep (pausa) pré-definida da Linguagem C que permite uma parada temporária (em segundos) na execução de um programa.

```
main()
{
int i;
    for(i = 10; i >= 0; i--)
    {
    printf("%i\n",i); // imprimi um contador ( 0 a 10 )
    sleep(1); // Espera 1 segundo
    }
}
```

NOTA ! ! !
Existem variações de sintaxe e bibliotecas para aplicações de compiladores distintos

Delay

A função delay (pausa) pré-definida do C que permite uma parada temporária (em milissegundos) na execução de um programa.

```
main()
{
int i;
    for(i = 10; i >= 0; i--)
    {
    printf("%i\n",i); // imprimi um contador ( 0 a 10 )
    delay(1000); // Espera 1 segundo
    }
}
```

NOTA ! ! !
Existem variações de sintaxe e bibliotecas para aplicações de compiladores distintos

FUNÇÃO

Definição

Conjunto de comandos agrupados em um bloco que recebe um nome e através deste pode ser ativado.

Porque usar funções ?

Para permitir o reaproveitamento de código já construído(por você ou por outros programadores);
Para evitar que um trecho de código que seja repetido várias vezes dentro de um mesmo programa;
Para permitir a alteração de um trecho de código de uma forma mais rápida. Com o uso de uma função é preciso alterar apenas dentro da função que se deseja;

FUNÇÃO

Porque usar funções ?

Para que os blocos do programa não fiquem grandes demais e, por consequência, mais difíceis de entender;

Para facilitar a leitura do programa-fonte de uma forma mais fácil;

Para separar o programa em partes(blocos) que possam ser logicamente compreendidos de forma isolada.

FUNÇÃO

Primeiro Exemplo

Criar um programa que some dois números.

Uma alternativa é criar uma função. Com o uso de funções, este processo de repetição fica simplificado. Observe o exemplo a seguir.

FUNÇÃO

```
#include<stdio.h>
#include<conio.h>
void soma();
void main()
{
int x;
clrscr();
printf("soma de dois n£meros\n");
soma();
getch();
}
void soma()
{
float a,b,c;
printf("digite o valor do primeiro numero e
tecle enter\n");
scanf("%f",&a);
printf("Digite o valor do segundo numero e
tecle enter\n");
scanf("%f",&b);
c=(a+b);
printf("resultado da soma %.2f\n",c);
}
```

FUNÇÃO

Segundo Exemplo

Em primeiro lugar, imaginemos que você necessite várias vezes em seu programa imprimir a mensagem "Pressione a tecla ENTER" e esperar que o usuário tecle ENTER, caso o usuário tecle algo diferente o programa deve imitir um BEEP.

Você pode fazer um laço de WHILE sempre que isto fosse necessário.

Uma alternativa é criar uma função. Com o uso de funções, este processo de repetição fica simplificado. Observe o exemplo a seguir.

FUNÇÃO

```
#include <conio.h>
#include <dos.h>
#include <stdio.h>

 void EsperaEnter()  // Definição da função "EsperaEnter"
      {
         int tecla;
         printf("Pressione ENTER\n");
         do
         {
            tecla = getch();
            if (tecla !=13) // Se nao for ENTER
            {
               sound(700); // Ativa a emissão de um BEEP
               delay(10);  // Mantém a emissão do som por 10 ms
               nosound();  // Para de emitir o som
            }
         } while(tecla != 13); // 13 e' o codigo ASCII do ENTER
      }

      void main()
      {
         EsperaEnter();      // Chamada da função definida antes
         // ..........
         EsperaEnter();      // Chamada da função definida antes
         // ..........
         EsperaEnter();      // Chamada da função definida antes
      }
```

FUNÇÃO

```
#include <conio.h>
#include <dos.h>
#include <stdio.h>

 void EsperaEnter()  // Definição da função "EsperaEnter"
      {
         int tecla;
         printf("Pressione ENTER\n");
         do
         {
            tecla = getch();
            if (tecla !=13) // Se nao for ENTER
            {
               sound(700); // Ativa a emissão de um BEEP
               delay(10);  // Mantém a emissão do som por 10 ms
               nosound();  // Para de emitir o som
            }
         } while(tecla != 13); // 13 e' o codigo ASCII do ENTER
      }

      void main()
      {
         EsperaEnter();      // Chamada da função definida antes
         // ..........
         EsperaEnter();      // Chamada da função definida antes
         // ..........
         EsperaEnter();      // Chamada da função definida antes
      }
```

Variável Global

Devem ser declaradas fora de todas as funções, incluindo a função main().

A variável declarada dessa maneira é chamada variável global.

Pode ser usada em qualquer parte do programa.

Programa exemplo:

```
/* soma os números */

int soma;        /* Variável global */
main()
{
int cont;        /* Variável local */

    for (cont = 0; cont < 10; cont ++)
    {
    printf("%d ", cont+soma);
    delay(1000);
    }
}
```

Variável Local

Devem ser declaradas dentro de uma função.

Estas variáveis são chamadas variáveis locais.

Podem ser usadas somente pelos comandos que estiverem na mesma função.

Programa exemplo:

```
/* soma os números */

int soma;        /* Variável global */
main()
{
int cont;        /* Variável local */

    for (cont = 0; cont < 10; cont ++)
    {
    printf("%d ", cont+soma);
    delay(1000);
    }
}
```

Programa exemplo:

```
/* soma os números */

int soma;        /* Variável global */
main()
{
int cont;        /* Variável local */

    for (cont = 0; cont < 10; cont ++)
    {
    printf("%d ", cont+soma);
    delay(1000);
    }
}
```