



Programação de Computadores I

Funções de Repetição da Linguagem C

PROFESSORA CINTIA CAETANO

Comando WHILE

- ▶ O comando while executa um bloco de comandos enquanto a condição testada for verdadeira (diferente de zero).
- ▶ Sua sintaxe geral é:

```
while (condição de teste)
{
    bloco_comandos;
}
```

Exemplo 1

```
# include <stdio.h>
void main(void)
{
    int i = 1, soma = 0;
    while (i <= 10)
    {
        soma += i;
        i++;
    }
    printf("A soma dos números inteiros de 1 a 10 é %d", soma);
}
```

Comando DO – WHILE

- ▶ O comando do – while é similar ao comando while.
- ▶ Ele primeiro executa o bloco de comandos para depois verificar a condição de teste, ou seja, neste comando, as instruções contidas no “loop” são executadas pelo menos uma vez.

- ▶ Sua sintaxe geral é:

```
do  
{  
    bloco_comandos;  
} while (condição de teste);
```

Exemplo 2

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

```
void main(void) {  
    int i = 1, soma = 0;  
    do  
    {  
        soma += i;  
        i++;  
    } while (i <= 10)  
    printf("A soma dos números inteiros de 1 a 10 é %d", soma);  
}
```

Comando FOR

- ▶ O comando for é utilizado quando se deseja executar instruções repetidas que envolvam o incremento/decremento das variáveis de controle do “loop”(ex: leitura e/ou manipulação de vetores e matrizes).

- ▶ Sua sintaxe geral é:

```
for (inicialização ; condição ; incremento)
{
    bloco_comandos;
}
```

Comando FOR

► Onde:

- **Inicialização** é geralmente um comando de atribuição utilizado para determinar a variável de controle do “loop” e seu valor inicial;
- **Condição** é uma expressão relacional que determina o critério de parada do comando for;
- **Incremento** é uma expressão aritmética que define como a variável de controle se altera a cada repetição do comando for.

Exemplo 3

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

```
void main(void)
```

```
{
```

```
    int i, soma=0;
```

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

```
        soma += i;
```

```
    printf("A soma dos números inteiros de 1 a 10 é %d", soma);
```

```
}
```

Variações do Comando FOR

- ▶ Qualquer uma das três instruções que formam o comando for pode ser omitida, embora os ponto e vírgulas devam permanecer.
- ▶ Se o teste for omitido, o “loop” se tornará infinito, podendo ser interrompido caso seja usado um comando de desvio (break, return ou goto) no bloco de comandos do for.

Exemplo 4

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

```
void main(void)
```

```
{
```

```
    char x;
```

```
    printf("\nDigite quaisquer caracteres ou tecle A para sair:\n");
```

```
    for (;;)
    {
```

```
        x=getchar();/* Lê um caracter */
```

```
        if (x == 'A')
```

```
            break; /* Sai do loop */
```

```
    }
```

```
    printf("\nVocê saiu do Loop");
```

```
}
```

Comandos de Desvio

- ▶ Às vezes, é conveniente se controlar a saída de uma estrutura de repetição (for, while ou do-while), de outro modo além dos testes condicionais de início ou fim do mesmo.
- ▶ Para isto pode-se usar alguns comandos que permitem desviar a execução normal de um programa.
- ▶ Tais comandos são apresentados a seguir.

Função BREAK

- ▶ O comando break encerra o processamento de estruturas de repetição ou a execução de um comando switch.
- ▶ Seu uso é opcional.

Função BREAK: Exemplo

```
#include <stdio.h>

void main(void)
{
    int i, num;
    for (i = 0; i < 100; i++)
    {
        printf("Digite um numero qualquer e um numero negativo para Sair: ");
        scanf("%d", &num);
        if (num < 0)
            break;
        printf("%d\n", num);
    }
}
```

Função CONTINUE

- ▶ O comando continue pára a seqüência de instruções que está sendo executada (iteração atual), passando para a próxima iteração, ou seja, voltando para a 1ª linha do bloco de comandos do “loop”.

Função CONTINUE: Exemplo

```
#include <stdio.h>

void main(void)
{
    int x;
    printf ("Digite um número inteiro positivo ou <100> para sair: \n");
    do
    {
        scanf ("%d", &x);
        if (x < 0)
        {
            printf ("O número digitado não será impresso por ser negativo. \n");
            continue;
        }
        printf("O número digitado foi %d. \n", x);
    } while (x != 100);
    printf ("O número 100 foi digitado.");
}
```

Função RETURN

- ▶ O comando return é o mecanismo de saída de uma função, retornando um valor para a função que a chamou.
- ▶ Sua sintaxe é:
`return (expressão_de_retorno);`
- ▶ Onde, expressão_de_retorno corresponde ao valor que será utilizado no lugar da chamada da função.
- ▶ Se o comando return estiver dentro de uma estrutura de repetição, ele não só sairá da estrutura como também da função onde foi executado.
- ▶ O exemplo abaixo mostra o return dentro da função que eleva um número ao quadrado.

Função RETURN: Exemplo

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

```
int quadrado(int y);
```

```
void main(void)
```

```
{  
    int num;  
    printf ("Digite um número: \n");  
    scanf ("%d", &num);  
    num = quadrado(num);  
    printf ("O quadrado do número é: %d", num);  
}
```

```
int quadrado (int y)
```

```
{  
    return (y *= y);  
}
```

Função GOTO

- ▶ O comando goto desvia a seqüência de execução lógica de um programa, ou seja, ele passa o controle de programa (realiza um salto incondicional) para a linha de comando identificada por um rótulo (“label”).
- ▶ O rótulo pode ser qualquer nome que atenda às regras de criação de identificadores e deve ser declarado na posição para onde se deseja saltar, seguido de dois pontos(:).
- ▶ Tanto o comando goto quanto o seu rótulo devem estar declarados dentro da mesma função.

Função GOTO

- ▶ A sintaxe geral do comando é:

```
goto nome_rotulo
```

```
...
```

```
...
```

```
...
```

```
nome_rotulo:
```

```
    bloco_comandos;
```

Função GOTO: Exemplo

```
#include <stdio.h>

void main(void)
{
    int x = 0;
    printf ("Entre com caracteres ou tecla 3 para sair: \n");
    for (;;)
    {
        x = getchar(); /* Lê um caracter */
        if (x == '3')
            goto saida; /* Sai do Loop */
        else
            putchar(x);
    }
    saida:
        printf("Você saiu do Loop");
}
```

Função Exit

- ▶ A função `exit()` é encontrada na biblioteca padrão do C.
- ▶ Sua utilização é limitada, pois provoca o encerramento imediato de um programa, retornando ao sistema operacional.

Função Exit: Exemplo

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

```
void main(void)
```

```
{
```

```
    char resposta;
```

```
    printf ("Digite um texto qualquer ou <S> para sair: \n");
```

```
    for (;;)
    {
```

```
        resposta = getchar();
```

```
        if (resposta == 'S' || resposta == 's')
```

```
            exit();
```

```
        else
```

```
            putchar(resposta);
```

```
    }
```

```
}
```