

Estrutura de Repetição

Do ... While



Estrutura de Repetição

Do ... While

O comando **DO WHILE** é utilizado quando necessitamos fazer um teste no final a cada iteração do laço.

Comando **do ... while**

Estrutura recomendada quando o número de repetições for desconhecido, sendo necessário um teste (condição) NO FINAL para interromper a repetição.

Sintaxe:

```
do {  
    comando A  
    comando B  
    ...  
} while (CONDIÇÃO);
```

Fluxo de execução:

- 1) Executa todos os comandos
- 2) Avalia a condição
- 3) Se resultar 0 (False), sai do laço
- 4) Se resultar 1 (True) retorna ao início e executa todos os comandos novamente.

Exemplo de teste no final

```
char palavra[15] = "UEM";  
printf("%s\n",palavra);  
int tamanho = strlen(palavra);  
printf("tamanho = %i\n",tamanho);  
int i = tamanho - 1;  
do{  
    printf("%c",palavra[i]);  
    i -= 1;  
    //outros comandos  
} while (i>=0);
```

Exemplo de teste no final

Suponha que você queira, por algum motivo estranho, somar os números naturais (1, 2, 3 etc.) até obter um total maior ou igual a 100.

Como seria o laço de repetição em pascal, python, java e C?

Laço em Pascal

repeat

$n := n + 1;$

$total := total + n;$

until ($total \geq 100$);

Laço em Python

```
while True:
```

```
    n += 1
```

```
    total += n
```

```
    if total >= 100: break
```

Laço em Java

```
do {  
    n = n + 1;  
    total = total + n;  
} while (total < 100);
```


Laço em C

```
do {  
    n += 1;  
    total += n;  
} while (total < 100);
```

Teste no final em C

Generalizando, qualquer loop com **teste no final** pode ser codificado em C usando-se o comando do ... while, assim:

```
do {  
    comando1  
    comando2  
    // etc.  
} while (condicao_final);
```

Exercícios

- 1) Faça um algoritmo que leia vários grupos de quatro valores (a,b,c,d) e mostre-os na ordem lida, em ordem crescente e decrescente.

Exercício 1

```
int a,b,c,d,aux,rep;
do {
    printf("\na= "); scanf("%i", &a);
    printf("\nb= "); scanf("%i", &b);
    printf("\nc= "); scanf("%i", &c);
    printf("\nd= "); scanf("%i", &d);
    printf("\nOrdem inicial: %i %i %i %i", a,b,c,d);
    do {
        if (a > b){
            aux = a; a=b; b=aux;
        }
        if (b > c){
            aux=b; b=c; c=aux;
        }
        if (c > d){
            aux=c; c=d; d=aux;
        }
    } while (a>b||b>c||c>d);
    printf("\nCrescente: %i %i %i %i", a,b,c,d);
    printf("\nDecrescente: %i %i %i %i", d,c,b,a);
    printf("\nDigite 1 para repetir ou 0 para parar: ");
    scanf("%i",&rep);
} while (rep!=0);
```

Comando **break**

Dentro de laços é possível usar um comando adicional para ajudar no controle dos mesmos. O comando é:

break: este comando interrompe a execução do laço.

Comando **break**

Exemplo:

```
int k;  
do{  
    printf("\nDigite um numero positivo ou 0 para sair: ");  
    scanf("%i",&k);  
    if (k==0) break;  
    printf("\nk = %i\n",k);  
} while (k<100);  
printf("\nSai do laco.\n");
```

Exercício 2

Faça um algoritmo que leia um número indeterminado de valores para M , todos inteiros e positivos, um de cada vez. Se M for par, verifique quantos divisores possui. Se M for ímpar, calcule a soma dos números inteiros de 1 até M (M não deve entrar nos cálculos). Mostre os resultados dos cálculos realizados. Finalize a entrada de dados com $M = 0$ ou negativo.

Exercício 2

```
int m,div,soma;
do {
    printf("\nM: "); scanf("%i", &m);
    if (m <= 0) break;
    if (m%2 == 0){
        div = 2;
        for (int k=1; k<m/2;k++)
            if (m%k == 0)    div = div + 1;
        printf("\nDivisores = %i", div);
    }
    else{
        soma = 0;
        for (int i=1; i<m; i++)
            soma = soma + i;
        printf("\nSoma = %i\n", soma);
    }
} while (m>0);
```


Comando **continue**

Dentro de laços é possível usar um comando adicional para ajudar no controle dos mesmos. O comando é:

continue: este comando vai para a próxima iteração do laço imediatamente.

Comando `continue`

Exemplo:

```
int k;  
do{  
    printf("Digite um numero positivo ou 0 para sair = ");  
    scanf("%i",&k);  
    if (k>0) printf("Numero valido\n\n");  
    else if (k<0) continue;  
}while(k!=0);
```

Exercício 3

O cardápio de uma lanchonete é o seguinte:

<i>Especificação</i>	<i>Código</i>	<i>Preço</i>
Cachorro quente	100	R\$ 4,50
Bauru simples	101	R\$ 5,30
Bauru com ovo	102	R\$ 6,00
Hambúrguer	103	R\$ 5,00
Cheeseburger	104	R\$ 5,00
Refrigerante	105	R\$ 3,50

Faça um algoritmo que leia o código dos itens pedidos e as quantidades desejadas. Calcule e mostre o valor a ser pago por item (preço * quantidade) e o total geral do pedido. Considere que o cliente deve informar quando o pedido deve ser encerrado.

Exercício 3

```
// Cardapio de uma lanchonete
int codigo, qtde, opcao;
float valor, total=0;
do{
    printf("\nInforme o codigo: "); scanf("%i",&codigo);
    if (codigo == 100)          valor = 4.50;
    else if (codigo == 101)      valor = 5.30;
    else if (codigo == 102)      valor = 6.00;
    else if (codigo == 103)      valor = 5.00;
    else if (codigo == 104)      valor = 5.00;
    else if (codigo == 105)      valor = 3.50;
    else{
        printf("\t** Informe um produto valido. **\n");  continue;
    }
    printf("\tInforme a quantidade: "); scanf("%i",&qtde);
    printf("\n\tProduto: %i \tQtde: %i ", codigo,qtde);
    printf("\tUnidade = %5.2f \tTotal = %5.2f", valor, valor * qtde);
    total += valor * qtde;
    printf("\n\nDigite 1 para encerrar o pedido ou 0 para continuar: ");
    scanf("%i",&opcao);
}while (opção != 1);
printf("\n\tTotal do pedido = %5.2f\n", total);
```

Exercício 4

Faça um algoritmo que apresente um **menu** de opções para o cálculo das seguintes operações entre dois números: adição, subtração, multiplicação e divisão. O programa deve possibilitar ao usuário a escolha da operação desejada, a exibição do resultado e a volta ao menu de opções. O programa só termina quando for escolhida a opção de saída.