Estruturas de Repetição: while, for, do...while. As instruções *break* e *continue* utilizadas em ciclos

while

A instrução while, executa uma instrução ou bloco de instruções enquanto uma determinada condição for verdadeira.

Sintaxe:

```
while (condição)
   Instrução;
```

Notas: Coloca-se entre parêntesis a condição que se tem que verificar para que a instrução ou bloco seja executado.

Funcionamento:

- Avalia uma condição;
- Se verdadeira executa a instrução ou bloco associado ao while;
- Se falsa o ciclo termina e o programa continua imediatamente a seguir ao while.

Exemplo:

Escreve um programa que envie para o ecrã os dez primeiros números inteiros positivos.

```
#include <stdio.h>
main()
{
  int i;
  i=1;
  while ( i<=10)
  {
    printf("%2d \n", i);  /* %2d, reserva 2 carateres para representar o número. */
    ++i;                  /* Se o nº não ocupar o nº de carateres indicados são */
                          /* colocados espaços à esquerda, evitando que os valores*/
   }                      /* apareçam desalinhados */
}
```

Nota: qual o output do programa se não colocássemos chavetas?

EXERCÍCIOS:

1. Escreve um programa que imprima no ecrã o conjunto das 5 primeiras tabuadas (output semelhante a : 5*1= 5). Deve colocar uma linha em branco depois de cada tabuada.

for

A instrução for adapta-se à partida a situações em que se conhece o número de vezes que se repete o ciclo, isto é, o número de iterações do ciclo.

Sintaxe:

```
for ( inicializações ; condição ; pós-condição)
   Instrução;
```

Nota: O ciclo **for** identifica as suas componentes, separando-as por ponto e vírgula (;). Assim, se for necessário realizar mais do que uma inicialização ou mais do que uma pós-instrução, estas deverão ser separadas por vírgulas (,).

Funcionamento:

- É executado o código presente em inicializações (normalmente inicializações de variáveis presentes no ciclo). Esta componente é executada uma única vez!
- A condição é avaliada;
- Se for verdadeira, então é executada a instrução (ou bloco) do ciclo;
- Após a execução da instrução é executada a pós instrução. Nesta componente é realizado o incremento ou decremento da variável, etc, isto é, operações que permitam passar à próxima iteração do ciclo, voltando a condição a ser avaliada;
- Se for falsa (zero), então o **ciclo for** termina e o programa continua na instrução imediatamente a seguir;

Exemplo Escreve um programa que envie para o ecrã os dez primeiros números inteiros positivos.

```
#include <stdio.h>
main()
{
   int a;
   for( i=1; i<=10 ; i=i+1)
   {
      printf("%2d \n",i);
}
```

EXERCÍCIOS:

2. Escreve um programa que execute a soma e o produto dos n primeiros números naturais.

3. Escreve um programa que escreva no ecrã, toda a tabela ASCII (0..255), escrevendo em cada linha o código ASCII e o caráter correspondente.

4. Escreve um programa que solicite ao utilizador um número e escreva em simultâneo a sequência crescente e decrescente entre 1 e esse número. Exemplo: Introd. Um nº: 5

```
1    5
2    4
3    3
4    2
5    1
```

5. Escreve um programa que solicite ao utilizador um número e um caráter. Em seguida terá que preencher **n** linhas, cada uma com **n** carateres.

Exemplo de output possível:

```
Introd. Um nº: 3
Introd. Um caráter: *
***
***
***
```

do while

A instrução *do while*, difere dos ciclos anteriores porque o teste da condição é realizado no fim do corpo (instrução ou bloco de instruções) do ciclo e não antes, como acontecia com os ciclos **while** e **for**.

Desta forma o corpo do ciclo é executado pelo menos uma vez, enquanto que nos ciclos while e for o corpo do ciclo pode nunca ser executado (caso a condição seja falsa à partida).

Sintaxe:

```
do
    Instrução;
while (condição);
```

Funcionamento:

- A instrução (ou bloco de instruções) é executada;
- A condição é avaliada;
- Se verdadeira, volta a executar a instrução (ou bloco de instruções);
- Se falsa, o ciclo termina e o programa continua imediatamente a seguir ao while (condição);

Nota: O ciclo do…while adapta-se particularmente ao processamento de menus.

Exemplo: Escreve um programa que apresente um menu com as opções Clientes, Fornecedores, Encomendas e Sair. O programa deve apresentar a opção escolhida pelo utilizador até que o utilizador deseje sair.

```
#include <stdio.h>
main()
{
   char op;
   do
   {
      printf("\t MENU PRINCIPAL\n");
      printf("\n\n\t Clientes");
      printf("\n\n\t Fornecedores");
      printf("\n\n\t Encomendas");
      printf("\n\n\t Sair");
      printf("\n\n\n\t\t Opção: ");scanf(" %c", &op);
      fflush(stdin);          /*limpar o buffer do teclado */

      switch (op)
      {
            case 'c':
            case 'C': puts("Opção CLIENTES"); break ;
            case 'f':
            case 'F': puts("Opção cFORNECEDORES"); break ;
            case 'e':
            case 'E': puts("Opção ENCOMENDAS"); break ;
            case 's':
            case 'S': break ; /* NÃO FAZ NADA*/
            default : puts("Opção INVÁLIDA !!!");
      }
      getchar() ; /*parar o ecrã*/
    }
    while (op!='s' || op!='S');
}
```

Ciclos (resumo)	while	for	do … while
Sintaxe	while(cond) Instrução;	for (inic;cond;pos-cond) Instrução;	do Instrução; while (cond);
Executa a instrução	Zero ou mais vezes	Zero ou mais vezes	Uma ou mais vezes
Testa a condição	Antes da instrução	Antes da instrução	Depois da instrução
Utilização	Frequente	Frequente	Pouco frequente

Break (num ciclo)

A instrução break, quando utilizada dentro dum ciclo, termina o correspondente ciclo, continuando a execução do programa na instrução imediatamente a seguir a esse ciclo.

Exemplo:

```
#include <stdio.h>
main()
{
   int i;
   for(i=1; i<=100 ; i=i+1) /*QUAL O OUTPUT DESTE PROGRAMA?*/
            if (i==30)
                break;
           else
                printf("%2d \n",i*2);
    printf("FIM DO CICLO\n");
}
```

continue (num ciclo)

A instrução continue, quando utilizada dentro dum ciclo, permite que a execução da instrução ou bloco de instruções corrente seja terminado, passando à próxima iteração do ciclo.

Exemplo:

```
#include <stdio.h>
main()
{
       int i;
   for( i=1; i<=100 ; i++)   / *QUAL O OUTPUT DESTE PROGRAMA?*/
            if (i==60)
          break ;
            else
              if (i%2==1)     /* se for ímpar */
                  continue;
              else
                   printf("%2d \n",i ) ;
    printf("FIM DO CICLO\n");
}
```

Ciclos encadeados

Ciclos encadeados, são ciclos (*while*, *for*, *do…while*) que estejam presentes dentro de outros ciclos. Não existe qualquer limitação ao número de ciclos que pode ocorrer dentro de outros ciclos.

Nota: Uma boa endentação facilita a leitura e representação de ciclos encadeados.

Exemplo (de ciclos encadeados):

```
While (x<=5)
{
   do
   {
       printf("Introd. Um nº: "); scanf("%d", &n);
    }
   While (n<0);                              /* Fim do ciclo do … while */
}                                            /* Fim do ciclo while */
```

ciclos infinitos

Denominam-se por ciclos infinitos, os ciclos que nunca terminam, isto é, apresentam condições que são sempre verdadeiras.

Exemplos:

While (1) Instrução; /*sempre verdade*/	for(; ;) instrução;	do instrução; While (1);

Este tipo de ciclos é utilizado normalmente quando não se sabe à partida qual o nº de vezes que se vai iterar o ciclo. Para terminar um ciclo infinito utiliza-se a instrução *break* ou *return*

```
While (1)
{
                                  /* Apresentar o menu e ler opção*/
   if (op==…)
    ….
   If (op===SAIR)
     Break;                  /*terminar o ciclo infinito*/
}
```

EXERCÍCIOS:

6. Escreve um programa que coloque os seguintes números no ecrã.

1
1 2
1 2 3
…
1 2 3 4 5 6 7 8 9 10

7. Escreve um programa que solicite ao utilizador um número. Em seguida escreve todos os números inteiros a partir desse número exceto os múltiplos de 3. Quando encontrar o 1º múltiplo de 10 termina a execução.

Exemplo de Saída de dados:

Introduza um nº : **13**
13
14
16
17
19

8. Escreve um programa que coloque no ecrã meia árvore de Natal com asteriscos. O nº de ramos deverá ser introduzido pelo utilizador.

Exemplos com 3 e 4 ramos:

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

9. Qual o resultado do seguinte extrato de programa:

```
n=0;
do
{
  printf("…");
    n=n+1;
}
while (n!=0);
```

10. Qual a diferença entre estes dois extratos de código:

```
i=0;
while (i++)
   printf("%d\n", i);
```

```
i=0;
while (++i)
   printf("%d\n", i);
```

11. O que faz o seguinte código:

```
for (i=1; i<=200; i++) ;
      printf("%d  ",i);
```