

Introdução à Programação

Comandos de Ciclo **WHILE, DO...WHILE, FOR**

1º ano - ESI e IGE (2011/2012)

Engenheiro Anilton Silva Fernandes
(afernandes@unipiaget.cv)

Analise o pedido:

- Faça um algoritmo que escreva por 200 vezes o seu nome no ecrã, em linhas distintas.
 - Pronto, posto a questão, vamos analisar a resolução

```
#include <stdio.h>
main()
{
    printf("Anilton\n");
    printf("Anilton\n");
    printf("Anilton\n");
    printf("Anilton\n");
    // ... ate 200
}
```

???

- Não parece muito boa a ideia de ter que escrever um **mesma** expressão por tantas vezes
- Alguém deve ter inventado alguma forma de fazer repetições sem ser necessário repetir expressões
 - É claro que alguém pensou, é por essa razão que temos as instruções de ciclo
 - Que basicamente servem para permitir repetições de expressões

Instruções de CICLO?

- Por variadíssimas vezes, o programados precisa fazer repetições várias, ou contar números sequenciais
 - Não seria boa prática ter que repetir pelas vezes que se pretende, uma mesma expressão, ou uma sequencia lógica.
 - Por exemplo: como é suposto fazer um programa que conta de 1 até 10?
 - Portanto, usamos instruções de CICLO para fazer repetições e para fazer sequências lógicas.

WHILE

- Uma das instruções de ciclo mais usado na programação em C, é o **WHILE**, que em português significa **ENQUANTO**, e permite basicamente repetir enquanto uma condição definida.
 - Exemplo do exercício 1:
 - ...

WHILE

```
#include <stdio.h>
main()
{
    int i=0;                // contador, serve para testar a condicao
                           // de paragem do ciclo

    while(i<200)           // enquanto i for menor que 200, ou
    {                       // enquanto i não chegar a 200
        printf("Anilton\n"); // expressão a repetir
        i++;                // instrução de incremento, para que i possa
                           // avançar. Mesmo que i = i + 1;
    }
}
```

WHILE

- Assim, é importante saber:
 - Para se ter um ciclo while, precisa-se de:
 - Um **CONTADOR**, cuja a função é incrementar-se, **APROXIMANDO-SE DA CONDIÇÃO DE PARAGEM**;
 - Uma **CONDIÇÃO de INICIALIZAÇÃO**, que é o valor inicial do contador. Permite especificar onde começa o ciclo.
 - Uma **CONDIÇÃO de PARAGEM**, que é a condição que diz quando é que o ciclo vai PARAR, e vem dentro do WHILE. Chega-se à condição de paragem com o incrementar ou decrementar do contador

WHILE

- **IMPORTANTÍSSIMO**
 - Quando o contador é definido com a **condição de inicialização**
 - E quando a **condição de paragem** for definida,
 - SE O CONTADOR FOR MAIOR QUE A CONDIÇÃO DE PARAGEM, este deve ser **DECREMENTADO (i--;)**
 - SE O CONTADOR FOR MENOR QUE A CONDIÇÃO DE PARAGEM, este deve ser **INCREMENTADO (i++;)**

WHILE

- **PORQUÊ?**

- Se o objectivo for aproximar-se da condição de paragem, então, não podemos estar a aumentar o que já é maior, porque vai afastar-se mais e mais.
 - Se $i=0$; e a condição de paragem for **`while(i != -1)`**, caso `i` for incrementado com `i++`;, ele será, 1, 2, 3, 4, 5, 6, 7..., e o nosso **ciclo nunca mais terá fim**

WHILE ...

- Fazer ciclos infinitos, pode ser devastador para o computador.
 - Tome sempre muito cuidado
 - Se um programa executado, começar um ciclo que se reconhece infinito, pressione, **CTRL+C**.
- Quando usar o ciclo WHILE?
 - Sempre que quiser repetir expressões ou instruções, ou pretende fazer uma sequência.

- Sobre o CICLO WHILE,

DUVIDAS ou PERGUNTAS

DO... WHILE

- Uma outra instrução de ciclo que temos, é o
 - **DO... WHILE**
 - Este faz a mesma forma que o WHILE, permitindo que expressões ou sequências sejam repetidas N vezes.
 - **CONTUDO:**
 - Difere do WHILE simples, porque enquanto no WHILE a condição é testada primeiro, no DO... WHILE, a condição é testada no fim

DO... WHILE

- Por exemplo.
 - No WHILE,

```
int i=0;  
while(i != 0)  
    printf("Programa do WHILE");
```

- Quando compilarmos esse programa o que aparecerá no ecrã?
 - **NADA**

DO... WHILE

- Se fosse com DO... WHILE, a expressão seria mostrada uma vez, porque o DO...WHILE, faz a expressão a repetir e só depois verifica se a condição e verdadeira para mostrar mais vezes

- **SINTAXE**

```
do{  
    //expressão a repetir  
- }while(condição);
```

- **REPARE QUE TEM ; NO FIM**

DO... WHILE

- **Então!**

```
int i=0;  
do{  
    printf("Programa do DO...WHILE");  
}while(i=0);
```

DO...WHILE – Quando Utilizar

- Embora se possa fazer com o Do...While o mesmo que o While, é importante dizer que o DO...WHILE é usado mais em questões de ciclos infinitos, onde o valor da variável é fornecido dentro do ciclo pelo utilizador

– EXEMPLO

```
int i;  
do{  
    printf("Escreva 2 para sair\n");  
    scanf("%d", &i);  
}while(i==2);
```

- Sobre o CICLO DO... WHILE,

DUVIDAS ou PERGUNTAS

Ciclo FOR

- Por último temos o ciclo FOR, que basicamente permite fazer a mesma coisa que o ciclo WHILE...
 - E deve ser preferida quando a condição de paragem for simples.
- A sintaxe é dada por:

```
for(condição início;condição de paragem;incremento)
{
    //expressao a repetir
}
```

Análise comparativa entre WHILE e FOR

```
#include <stdio.h>
main()
{
    int i=0;
    while(i<10)
    {
        printf("%d", I);
        i++;
    }
}
```

```
#include <stdio.h>
main()
{    int i;
    for(i=0;i<10;i++)
        printf("%d", I);
}
```

WHILE e FOR

- Uma outra instrução de ciclo que temos, é o
 - **DO... WHILE**
 - Este faz a mesma forma que o WHILE, permitindo que expressões ou sequências sejam repetidas N vezes.
 - **CONTUDO:**
 - Difere do WHILE simples, porque enquanto no WHILE a condição é testada primeiro, no DO... WHILE, a condição é testada no fim