

# Linguagem C

## Estruturas de Repetição

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro  
Prof. Edwar Saliba Júnior  
Janeiro de 2018

# Estruturas de Repetição

- Também são conhecidas como: laços ou *loops*;
- Estrutura essencial na programação estruturada. Ela possibilita a execução de uma determinada parte do código-fonte várias vezes, ou seja, repetidamente;
- Existem três tipos de estruturas de repetição:
  - *for*
  - *while*
  - *do ... while.*

## Problema Inicial

- Imprimir na tela do computador, os números de 1 até 1000.

Possível.  
Mas absurdo!  
Mil linhas de  
Código.

```
main.c *
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("1");
6      printf("2");
7      printf("3");
8      printf("4");
9      .
10     .
11     .
12     printf("1000");
13
14     return 0;
15 }
16
```

## *for*

- Comando `for` sintaxe:

```
for(inicialização; condição de parada; incremento) {  
    bloco de comandos  
}
```

- Onde:
  - **inicialização** – local onde deve-se por os comandos de inicialização das variáveis. Este local é executado uma única vez;
  - **condição de parada** – local de teste onde deve-se ter uma condição que interrompa o *loop*. A cada iteração esta condição é testada, e enquanto seu resultado lógico for **verdadeiro**, segue-se com a execução do `bloco de comandos`;
  - **incremento** – área que possui comandos que serão executados ao final da iteração. Nesta área geralmente são colocados comandos que incrementam variáveis.

# Resolução do Problema Inicial Utilizando *for*

```
main.c [x]
1  #include <stdio.h>
2
3  int main()
4  {
5      int cont;
6
7      for(cont = 0; cont <= 1000; cont++)
8          printf("\n %i", cont);
9
10     return 0;
11 }
12
```

## *while*

- Comando **while** sintaxe:

```
while(condição) {  
    bloco de comandos  
}
```

- O comando `while` é mais utilizado quando não se pode determinar previamente, quantas vezes um bloco de comandos será executado;
- Inicialmente a **condição** é testada. Caso seja **falsa**, o programa não executará o bloco de comandos e continuará no comando após a `}` do comando `while`.
- Caso a **condição** seja **verdadeira**, o bloco de comandos é executado. Ao final da execução do bloco de comandos, volta-se a testar a **condição**.
- O bloco de comandos, portanto será executado até que a **condição** torne-se **falsa**. Ou seja, o bloco de comandos será executado enquanto a **condição** for **verdadeira**.

# Resolução do Problema Inicial Utilizando *while*

```
main.c [x]
1  #include <stdio.h>
2
3  int main()
4  {
5      int cont = 0;
6
7      while(cont <= 1000){
8          printf("\n %i", cont);
9          cont++;
10     }
11
12     return 0;
13 }
14
```

## *do ... while*

- Comando `do ... while` sintaxe:

```
do{  
    bloco de comandos  
}while(condição);
```

- O comando `do ... while` diferencia-se do comando `while` somente em um detalhe. O bloco de comandos é sempre executado pelo menos uma vez.
- Após a execução do bloco de comandos a **condição** é testada. Caso seja **verdadeira**, o bloco de comandos será executado novamente.
- A execução sairá da estrutura de repetição somente quando a **condição** retornar **false**.

# Resolução do Problema Inicial Utilizando *do ... while*

```
main.c [x]
1  #include <stdio.h>
2
3  int main()
4  {
5      int cont = 0;
6
7      do{
8          printf("\n %i", cont);
9          cont++;
10     }while(cont <= 1000);
11
12     return 0;
13 }
14
```

# Outros Exemplos

- Imprimir na tela, os números pares de 2 a 1000.

*do ... while*

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int cont = 2;
6
7      do{
8          printf("\n %i", cont);
9          cont += 2;
10     }while(cont <= 1000);
11
12     return 0;
13 }
14
    
```

*for*

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int cont;
6
7      for(cont = 2; cont <= 1000; cont += 2)
8          printf("\n %i", cont);
9
10     return 0;
11 }
12
    
```

*while*

```

1  #include <stdio.h>
2
3  int main()
4  {
5      int cont = 2;
6
7      while(cont <= 1000){
8          printf("\n %i", cont);
9          cont += 2;
10     }
11
12     return 0;
13 }
14
    
```

Saída					
2	4	6	8	....	1000

## Outras Exemplos

- *Loop* infinito;
- Condição de parada do *loop* não existe ou é sempre verdadeira;
- Casos específicos.

do ... while

```
1 #include <stdio.h>
2
3 int main()
4 {
5     do{
6         bloco de comandos
7     }while(1);
8
9     return 0;
10 }
11
```

for

```
1 #include <stdio.h>
2
3 int main()
4 {
5     for(;;){
6         bloco de comandos
7     }
8
9     return 0;
10 }
11
```

while

```
1 #include <stdio.h>
2
3 int main()
4 {
5     while(1){
6         bloco de comandos
7     }
8
9     return 0;
10 }
11
```

# Flexibilidade do *for*

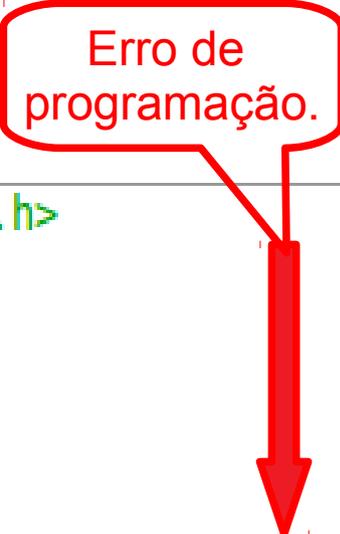
- Qualquer expressão de um laço `for` pode conter várias instruções separadas por vírgula;
- Inicializa `x` e `y` com 0;
- Testa se `x + y < 100`;
- Executa comando o comando `printf`;
- Incrementa 1 a `x`, e `x` a `y`;
- Raramente utilizada.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int x, y;
6
7      for (x=0, y=0; x+y < 100; ++x, y=y+x)
8          printf("%d \n",x+y);
9
10     return 0;
11 }
12
```

## Erro Comum de Programação

- A estrutura de repetição `for` **não** termina com “;”.
- Se você colocar um “;” no final do `for`, isto não causa um erro de sintaxe;
- Dado o problema: imprima na tela os número de 1 até 99. Para a resolução deste problema, foi desenvolvido o código-fonte ao lado;
- O que será impresso na tela do computador?

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int y;
6
7     for (y=0; y < 100; y++);
8         printf("%d \n", y);
9
10    return 0;
11 }
12
```



# Exemplo de *do ... while* para Restringir Valores

- No exemplo ao lado, criamos um programa que simula um *menu* de 4 opções;
- Pergunta-se: Para que serve a estrutura de repetição neste programa?

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int opcao = 0;
7
8      do{
9          printf("Escolha uma Opção:\n");
10         printf("1 - Inserir\n");
11         printf("2 - Excluir\n");
12         printf("3 - Alterar\n");
13         printf("4 - Consultar\n");
14         printf("Opção: ");
15         scanf("%i",&opcao);
16     }while((opcao < 1) || (opcao > 4));
17
18     return 0;
19 }
20
```

# Exemplo de *do ... while* com Término Definido pelo Usuário

- Neste programa o usuário do *software* define quando o mesmo terminará.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int valor = 0;
6
7      do{
8          printf("\nDigite um valor ou 0 para terminar: ");
9          scanf("%i",&valor);
10     }while(valor != 0);
11
12     return 0;
13 }
14
```

# *Break e Continue*

- ***break***
  - Força a saída de um laço (*for*, *while*, *do ... while*) antes do término previsto. Desaconselhável o uso do comando para este fim;
  - Também utilizado no comando *switch*;
- ***continue***
  - Volta ao início do laço.

## Exercícios

- A espessura de uma folha de papel é de  $X$  mm ( $X$  é definida pelo usuário). Forma-se uma pilha de folhas com uma quantidade  $Y$ . Sendo que  $Y$  também será definido pelo usuário. Imprima a altura desta pilha de folhas, a partir dos valores de  $X$  e  $Y$ , na construção deste *software*, utilize a estrutura de repetição *do ... while*).
- Faça um programa para calcular o fatorial de um número inteiro digitado pelo usuário. Lembre-se das restrições que o cálculo do fatorial tem. (Para resolver este problema, use a estrutura *while*).
- Faça um programa para calcular  $X^Y$ . Sendo que  $X$  e  $Y$  serão números inteiros escolhidos pelo usuário. Lembre-se das restrições do cálculo de potência. (Para resolver este problema, use a estrutura *for*).

# Bibliografia

- LAUREANO, Marcos. **Programação em C para ambiente Linux**. Disponível em: <<http://br-c.org/doku.php>>. Acesso em: 06 fev. 2011.
- MURTA, Cristina Duarte. *Slides da disciplina de Programação de Computadores I*. CEFET-MG, 2010.
- SENNE, Edson Luiz França. **Primeiro Curso de Programação em C**. 2. ed. Florianópolis: Visual Books, 2006.