# Linguagem C
# Estruturas de Repetição

Prof. Edwar Saliba Júnior
Fevereiro de 2011

# 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;
- Se subdividem em três estruturas:
  - *for*
  - *while*
  - *do … while*.

# Problema Inicial

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

Possível.
Mas absurdo!
Mil linhas de Código.

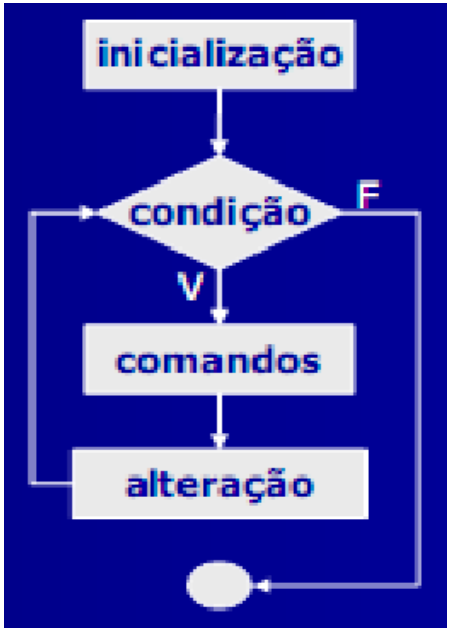
main.c

```c
#include <stdio.h>

int main()
{
    printf("1");
    printf("2");
    printf("3");
    printf("4");
    .
    .
    .
    printf("1000");

    return 0;
}
```

# *for*




- Comando `for` sintaxe:

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

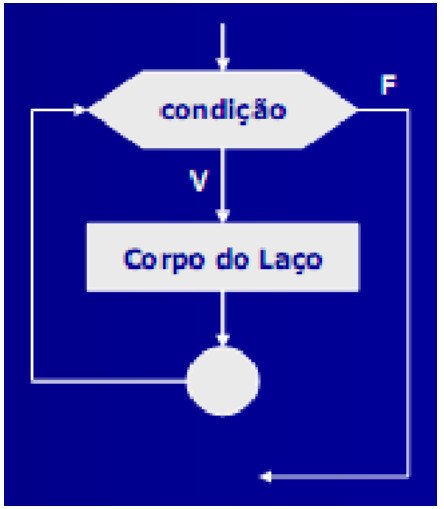
- Onde:
  - `inicialização` – área onde deve-se por os comandos de inicialização de variáveis. Esta área é executada uma única vez;
  - `condição de parada` – área de teste onde deve-se ter uma condição que interrompa o *loop*. A cada iteração esta condição é testada, e enquanto for `verdadeira`, segue-se 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

```
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
}
```

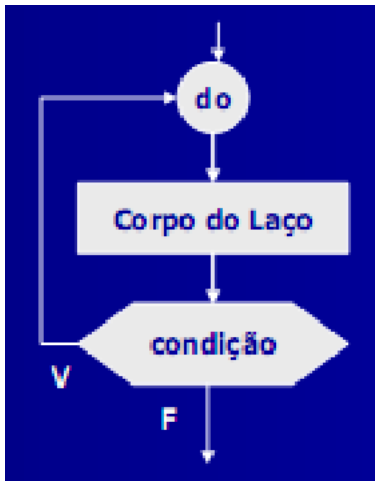
- O comando `while` é mais utilizado quando não se pode determinar previamente, quantas vezes um `bloco de comandos` será executado;
- Inicialmente a `condição` é testada. Caso seja `falsa`, o programa não executará o `bloco de comandos` e continuará no comando após a `}` do comando `while`.
- Caso a `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

```
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 `falso`.

# Resolução do Problema Inicial Utilizando *do … while*

main.c

```
1   #include <stdio.h>
2
3   int main()
4   {
5       int cont = 1;
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

```c
#include <stdio.h>

int main()
{
    int cont = 2;

    do{
        printf("\n %i", cont);
        cont += 2;
    }while(cont <= 1000);

    return 0;
}
```

for

```c
#include <stdio.h>

int main()
{
    int cont;

    for(cont = 2; cont <= 1000; cont += 2)
        printf("\n %i", cont);

    return 0;
}
```

while

```c
#include <stdio.h>

int main()
{
    int cont = 2;

    while(cont <= 1000){
        printf("\n %i", cont);
        cont += 2;
    }

    return 0;
}
```

**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*

```c
#include <stdio.h>

int main()
{
    do{
        bloco de comandos
    }while(1);

    return 0;
}
```

*for*

```c
#include <stdio.h>

int main()
{
    for(;;){
        bloco de comandos
    }

    return 0;
}
```

*while*

```c
#include <stdio.h>

int main()
{
    while(1){
        bloco de comandos
    }

    return 0;
}
```

# 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.

```c
#include <stdio.h>

int main()
{
    int x, y;

    for (x=0, y=0; x+y < 100; ++x, y=y+x)
        printf("%d \n",x+y);

    return 0;
}
```

# 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 100. Para a resolução deste problema, foi desenvolvido o código- fonte ao lado;
- O que sairá 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("Opcão: ");
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

- No programa ao lado, o usuário do *software* define quando o mesmo terminará.

```c
#include <stdio.h>

int main()
{
    int valor = 0;

    do{
        printf("\nDigite um valor ou 0 para terminar: ");
        scanf("%i",&valor);
    }while(valor != 0);

    return 0;
}
```

# *Break* e *Continue*

- ***break***
  - Força a saída de um laço (*for*, *while*, *do … while*) antes do término previsto;
  - 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.