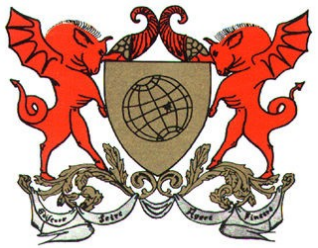
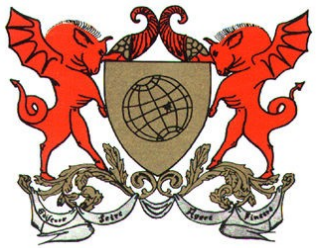


Universidade Federal de Viçosa
Campus Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas

SIN 110
Programação

Sistemas de Informação
Prof. Guilherme C. Pena
guilherme.pena@ufv.br

Le Rodrigo Smarzaro :)



Universidade Federal de Viçosa
Campus Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas

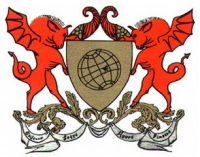
Aula de Hoje

Estruturas de Repetição em C

Estruturas de Repetição

Até agora sempre foi possível resolver os problemas com uma sequência de instruções onde todas eram necessariamente executadas **uma única vez**.

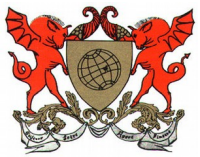
Os algoritmos que escrevemos seguiam uma sequência **linear** de operações.



Estruturas de Repetição

Suponha que você tenha que fazer um programa que entre com três notas de um aluno e imprima o valor da média aritmética na tela. Como seria codificado?

```
#include <stdio.h>
int main(){
        float nota1, nota2, nota3, media;
        printf("Digite as notas do aluno:");
        scanf("%f%f%f", &nota1, &nota2, &nota3);
        media = (nota1 + nota2 + nota3)/3;
        printf("Media do aluno e : %f\n", media);
        system("pause");
        return 0;
}
```

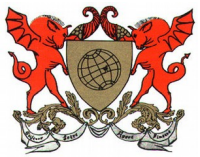


Estruturas de Repetição

Suponha que o mesmo cálculo tenha que ser feito agora para uma turma de 50 alunos. Como esse programa seria reescrito?

```
#include <stdio.h>
int main(){
    float nota1, nota2, nota3, media;
    printf("Digite as notas do aluno 1:");
    scanf("%f%f%f", &nota1, &nota2, &nota3);
    media = (nota1 + nota2 + nota3)/3;
    printf("Media do aluno 1 e : %f\n", media);
```

Aluno 1



Estruturas de Repetição

```
...
    printf("Digite as notas do aluno 2:");
    scanf("%f%f%f", &nota1, &nota2, &nota3);
    media = (nota1 + nota2 + nota3)/3;
    printf("Media do aluno 2 e : %f\n", media);
    printf("Digite as notas do aluno 3:");
    scanf("%f%f%f", &nota1, &nota2, &nota3);
    media = (nota1 + nota2 + nota3)/3;
    printf("Media do aluno 3 e : %f\n", media);
    printf("Digite as notas do aluno 4:");
    scanf("%f%f%f", &nota1, &nota2, &nota3);
    media = (nota1 + nota2 + nota3)/3;
    printf("Media do aluno 4 e : %f\n", media);
```

Aluno 2

Aluno 3

Aluno 4

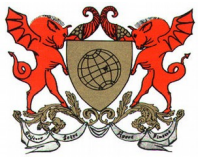
Estruturas de Repetição

```
…
    printf("Digite as notas do aluno 5:");
    scanf("%f%f%f", &nota1, &nota2, &nota3);
    media = (nota1 + nota2 + nota3)/3;
    printf("Media do aluno 5 e : %f\n", media);
    printf("Digite as notas do aluno 49:");
    scanf("%f%f%f", &nota1, &nota2, &nota3);
    media = (nota1 + nota2 + nota3)/3;
    printf("Media do aluno 49 e : %f\n", media);
    printf("Digite as notas do aluno 50:");
    scanf("%f%f%f", &nota1, &nota2, &nota3);
    media = (nota1 + nota2 + nota3)/3;
    printf("Media do aluno 50 e : %f\n", media);
    return 0;
}
```

Aluno 5

Aluno 49

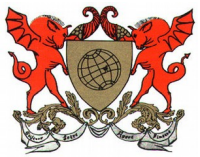
Aluno 50



Estruturas de Repetição

Da para notar que a codificação do programa fica extremamente trabalhosa, inclusive um fluxograma para este algoritmo ficaria imenso.

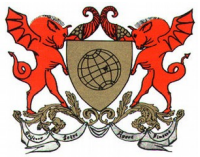
Existe um conjunto de estruturas **sintáticas** que permitem que um trecho de um algoritmo (lista de comandos) seja **repetido** um determinado **número de vezes**, sem que o código correspondente tenha que ser escrito mais de uma vez.



Estruturas de Repetição

Estruturas de repetição, **laços de repetição** ou **comandos de iteração**, permitem que um conjunto de instruções seja executado repetidamente até atingir uma certa **condição.**

Condição de parada que interrompe a execução da estrutura de repetição.

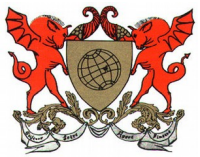


Estruturas de Repetição

Assim como as estruturas de seleção, as estruturas de repetição também **alteram o fluxo de execução** de um algoritmo e consistem de testes de **expressões lógicas**.

Tipos de estruturas de repetição:

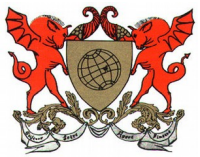
- Estrutura de repetição com **teste no início**
- Estrutura de repetição com **teste no final**
- Estrutura de repetição com **variável de controle embutida**



Estruturas de Repetição

Quando falamos de **estruturas de repetição** ou **laços de repetição**, surgem alguns novos conceitos importantes:

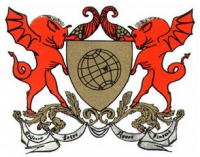
- Variável de Controle
- Incremento/Decremento
- Iteração
- Laços Infinitos (Loops)



Variável de Controle

Na maioria das vezes, as **expressões lógicas** que são testadas em estruturas de repetição contém variáveis de controle.

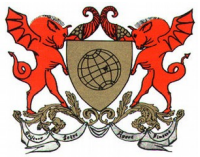
Uma variável de controle é uma variável **declarada e inicializada fora** do corpo da estrutura de repetição e sempre **atualizada dentro** do corpo da estrutura.



Variável de Controle

Quando se usa uma variável de controle, ela tem o objetivo de colocar um **fim na repetição**.

As variáveis de controle podem assumir **qualquer tipo**, no entanto seu uso mais comum são com os tipos numéricos (inteiros ou reais).



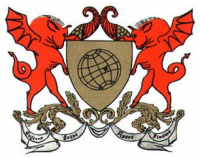
Incremento/Decremento

Os incrementos/decrementos ocorrem geralmente quando estamos atualizando uma variável de controle.

Exemplo:

```
int i;
i = 1;
```

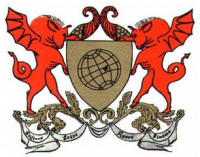
- **INCREMENTO**: i = i + 1; (Aumento do valor da Var.)
- **DECREMENTO**: i = i – 3; (Diminuição do valor da Var.)



Iteração

Damos o nome de iteração a cada vez que todos os comandos dentro de uma estrutura de repetição forem executados.

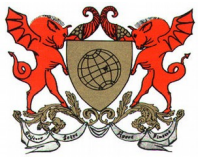
Neste caso, também pode-se dizer que a iteração foi completa.



Estruturas de Repetição

Em C, as estruturas de repetição são o **while**, **do… while** e **for** as quais nos permitem executar um **trecho de algoritmo** **zero ou mais vezes**.

E também existem alguns comandos de desvio de fluxo incondicional: **break, continue**.

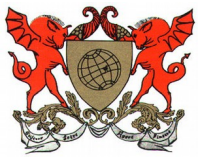


Operadores Especiais

A linguagem C fornece operadores adicionais que são muito usados com estruturas de repetição:

- Operadores de Incremento/Decremento

- Operadores de Atribuição Especiais



Operadores de Incremento/Decremento

C fornece quatro operadores unários que **somam** ou **subtraem** apenas **uma unidade** da respectiva variável:

Operador	Nome	Expressão	Explicação
++	Pré-incremento	++i	Incrementa i em 1, então utiliza o novo valor de i na expressão em que reside.
++	Pós-incremento	i++	Utiliza o valor atual de i na expressão em que reside e depois incrementa i em 1.
--	Pré-decremento	--j	Decrementa i em 1, então utiliza o novo valor de j na expressão em que reside.
--	Pós-decremento	j--	Utiliza o valor atual de j na expressão em que reside e depois decrementa i em 1.

Operadores de Atribuição Especiais

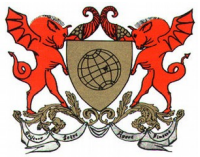
Qualquer expressão na forma:

```
variável = variável operador expressão;
```

Pode ser escrita na forma:

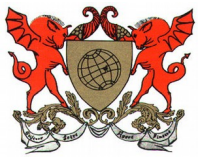
```
variável operador= expressão;
```

Operador	Expressão	Explicação	Supondo int a = 5;
+=	a += 5	a = a + 5	a = 10
-=	a -= 5	a = a - 5	a = 0
*=	a *= 5	a = a * 5	a = 25
/=	a /= 5	a = a / 5	a = 1
%=	a %= 5	a = a % 5	a = 0



Precedência dos Operadores

Prioridade	Operador(es)	Exemplo
1	++ -- (pós)	i++
2	! ++ -- (pré) - (unário)	++i
3	* / %	x / y
4	+ -	x - y
5	< <= > >=	x < y
6	== !=	x == y
7	&&	a && b
8	||	a || b
9	= += -= *= /= %=	x = 2



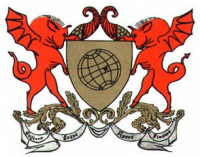
Estruturas de Repetição

Estruturas de repetição para um **"número indefinido"** de repetições:

- Comando **while**
- Comando **do..while**

Em geral, são utilizadas quando **não** se sabe o número de vezes que um trecho de código deve ser repetido.

Embora, também podem ser utilizadas quando se sabe esse número.

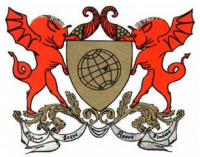


Estruturas de Repetição

Estruturas de repetição para um **"número indefinido"** de repetições:

- Comando **while**
- Comando **do..while**

Uma diferença importante entre os laços **while** e **do-while** envolve o **momento** em que a condição é verificada.



Estrutura de repetição com teste no início

A condição é verificada **antes** que o corpo do laço seja executado.

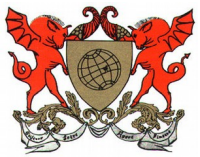
```
while (<condição>) {
    instrução 1;
    ...
    instrução n;
}
```

While significa "**enquanto**". Enquanto a condição for verdadeira ele repete.

Estruturas de Repetição (**while**)

```
while (<condição>) {
    instrução 1;
    ...
    instrução n;
}
```

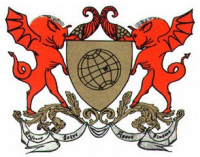
- A <condição> é avaliada e, se ela for **verdadeira**, a <lista de comandos> é executada.
- Os comandos serão executados **enquanto** a condição for **verdadeira**. Se ela for **falsa**, a repetição para.
- Existe a possibilidade da <lista de comandos> **nunca ser executada**.



Estruturas de Repetição (**while**)

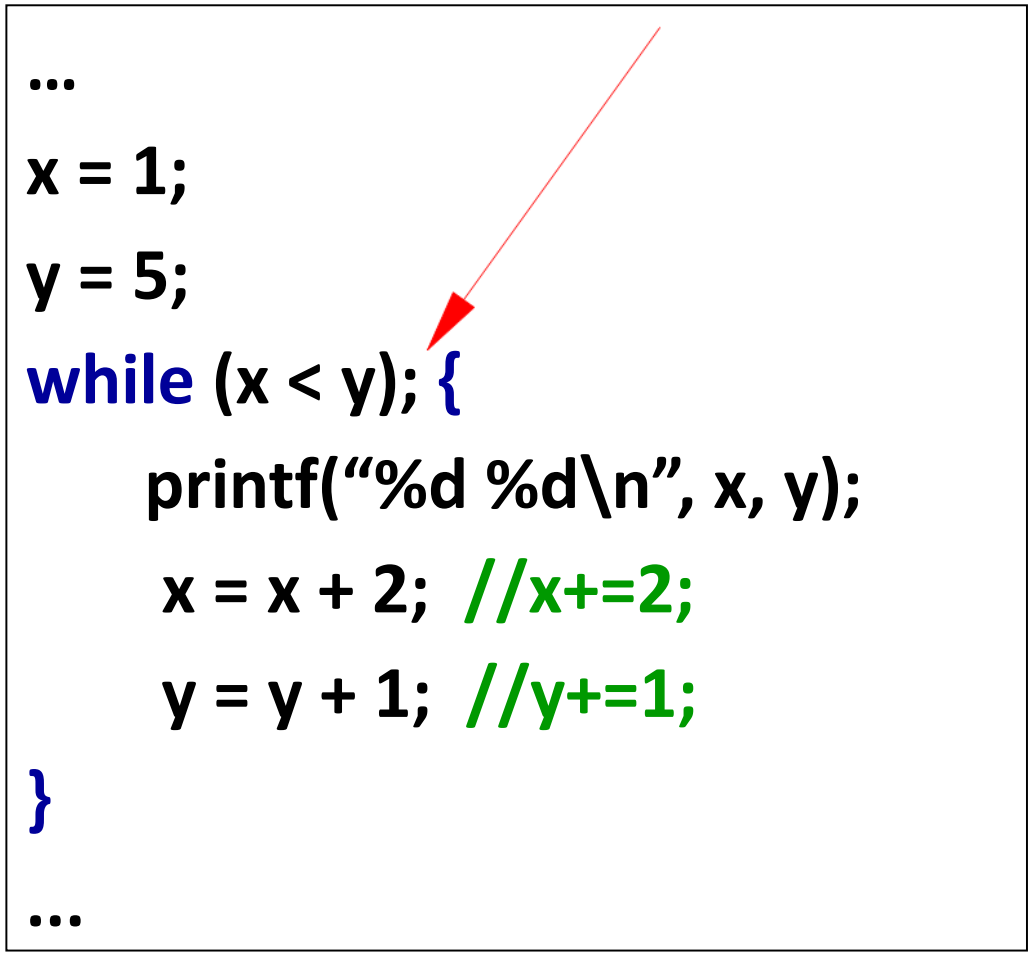
Exemplo:

```
...
x = 1;
y = 5;
while (x < y) {
        printf("%d %d\n", x, y);
        x = x + 2;  //x+=2;
        y = y + 1;  //y+=1;
}
...
```



Estruturas de Repetição (**while**)

Exemplo (**Erro Comum**):

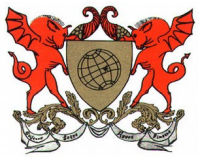


```
...
x = 1;
y = 5;
while (x < y); {
        printf("%d %d\n", x, y);
        x = x + 2;  //x+=2;
        y = y + 1;  //y+=1;
}
...
```

Estruturas de Repetição (**while**)

Exemplo: Calcula a média de 1 aluno

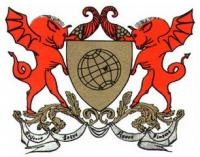
```
#include <stdio.h>
int main(){
        float nota1, nota2, nota3, media;
        printf("Digite as notas do aluno 1:");
        scanf("%f%f%f", &nota1, &nota2, &nota3);
        media = (nota1 + nota2 + nota3)/3;
        printf("Media do aluno 1 e : %f\n", media);
        system("pause");
        return 0;
}
```

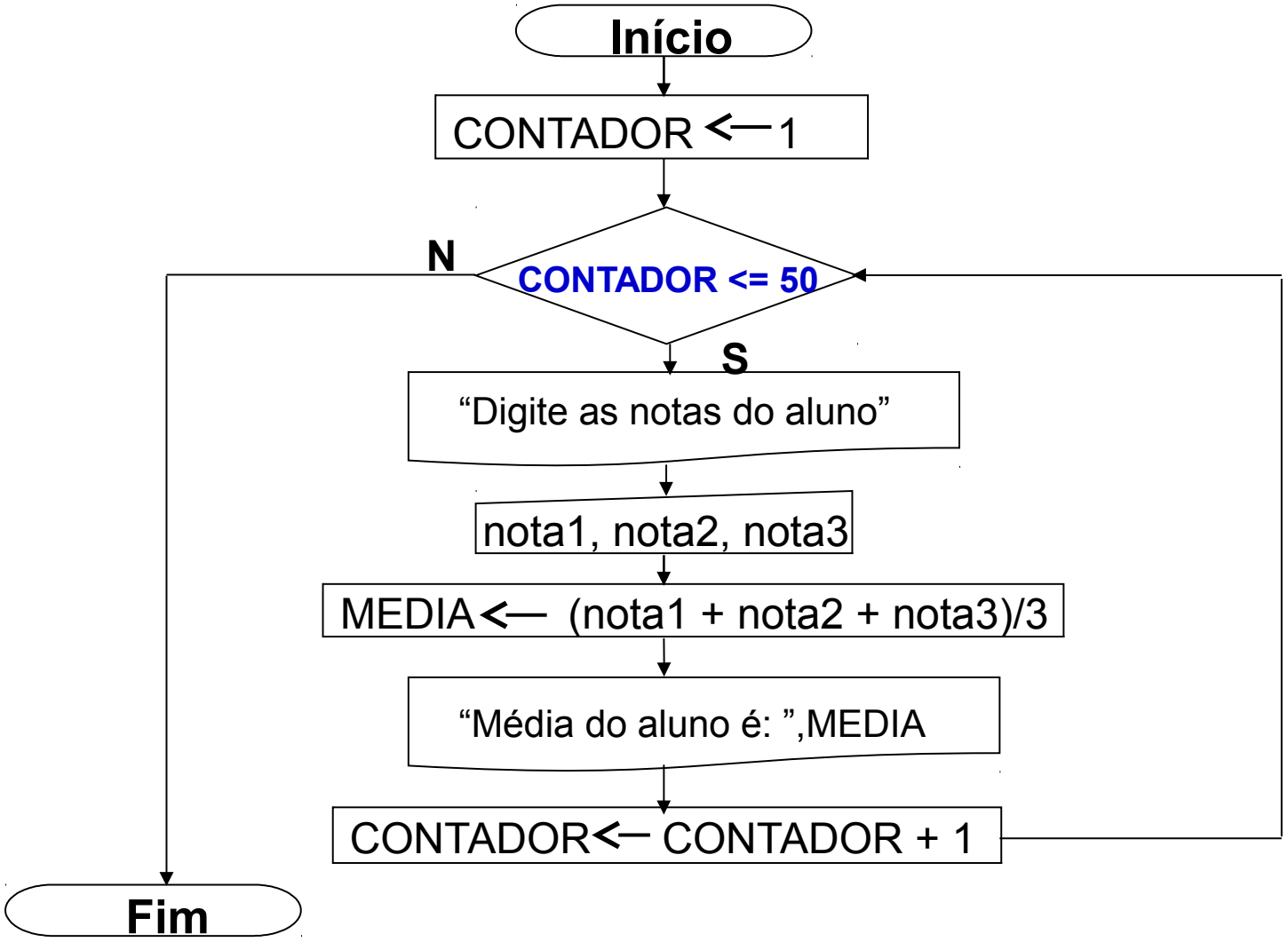
Estruturas de Repetição (**while**)

Exemplo: **Calcula a média de 50 alunos**

```
#include <stdio.h>
int main(){
        float nota1, nota2, nota3, media;
        int contador;
        contador = 1;
        while (contador <= 50) {
             printf("Digite as notas do aluno %d:",  contador);
             scanf("%f%f%f", &nota1, &nota2, &nota3);
             media = (nota1 + nota2 + nota3)/3;
             printf("Media do aluno %d e : %f\n", contador, media);
             contador++;
        }
        system("pause");
        return 0;
}
```

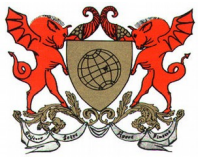


Estrutura de repetição com teste no início



Estrutura de repetição

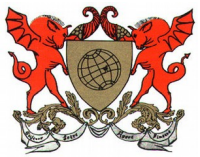
Qual **modificação** no código devemos fazer para que o cálculo da média seja feito para uma turma genérica (qualquer) que possui **N** alunos?



Estruturas de Repetição (**while**)

Exemplo:

```
#include <stdio.h>
int main(){
        float nota1, nota2, nota3, media;
        int contador, n;
        printf("Digite a quantidade de alunos: ");
        scanf("%d", &n);
        contador = 1;
        while (contador <= n) {
             printf("Digite as notas do aluno %d:",  contador);
             scanf("%f%f%f", &nota1, &nota2, &nota3);
             media = (nota1 + nota2 + nota3)/3;
             printf("Media do aluno %d e : %f\n", contador, media);
             contador++;
        }
        system("pause");
        return 0;
}
```

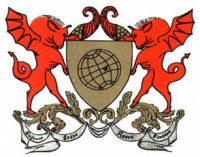


Estruturas de Repetição (**while**)

Exemplo:

Escreva um programa em C que mostre todas as letras do alfabeto na tela.

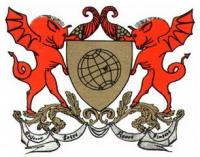
Dica: o tipo char é armazenado como um 1 byte que representa um símbolo na tabela ASCII e também representa um número inteiro (-128..127)



Estruturas de Repetição (**while**)

Exemplo:

```
#include <stdio.h>
int main(){
    int i = 0;
    while(i<26){
      printf("%c\n", i+'a');
      i++;
    }
    system("pause");
    return 0;
}
```

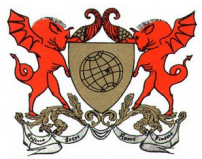


Estruturas de Repetição (**while**)

Exemplo:

```
#include <stdio.h>
int main(){
    int i = 0;
    while(i<26){
      printf("%c\n", i+'a');
      i++;
    }
    system("pause");
    return 0;
}
```

começa a imprimir a partir do 'a' em diante.

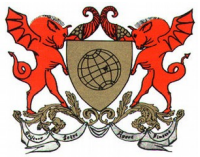


Estruturas de Repetição (**while**)

Exemplo:

Escreva um algoritmo que leia números positivos digitados pelo usuário e escreva, para cada um, o seu dobro.

Qual é a condição?



Estruturas de Repetição (**while**)

Exemplo:

```
#include <stdio.h>
int main(){
    int num;
    scanf("%d", &num);
    while(num > 0){
      printf("%d\n", num*2);
      scanf("%d", &num);
    }
    system("pause");
    return 0;
}
```

quando vai parar?

Estruturas de Repetição (**do..while**)

A condição é verificada **depois** que o corpo do laço seja executado.

```
do {
    instrução 1;
    ...
    instrução n;
} while (<condição>);
```

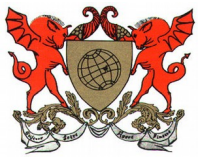
Do..While significa "**Faça..enquanto**". Faça/repita a lista de comandos enquanto a condição for verdadeira.

Estruturas de Repetição (**do..while**)

```
do {
    instrução 1;
    ...
    instrução n;
} while (<condição>);
```

Apenas no **do..while** deve-se obrigatoriamente terminar com ponto-e-vírgula.

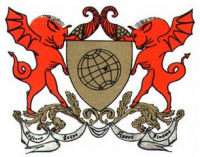
- A <condição> é avaliada e, se ela for **verdadeira**, a <lista de comandos> continua sendo executada.
- Os comandos serão executados **enquanto** a condição for **verdadeira**. Se ela for **falsa**, a repetição para.
- Nesse caso a <lista de comandos> é **executada pelo menos uma vez**.



Estruturas de Repetição (**do..while**)

Exemplo:

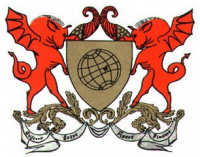
```
...
x = 1;
y = 5;
do{
      printf("%d %d\n", x, y);
       x = x + 2;  //x+=2;
       y = y + 1;  //y+=1;
}while (x < y) ;
...
```



Estruturas de Repetição (**while**)

Exemplo:

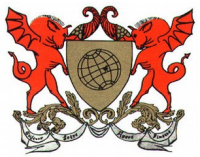
```
#include <stdio.h>
int main(){
        float nota1, nota2, nota3, media;
        printf("Digite as notas do aluno 1:");
        scanf("%f%f%f", &nota1, &nota2, &nota3);
        media = (nota1 + nota2 + nota3)/3;
        printf("Media do aluno 1 e : %f\n", media);
        system("pause");
        return 0;
}
```



Estruturas de Repetição (**do..while**)

Exemplo:

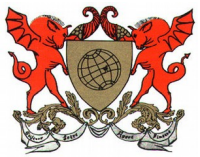
```
#include <stdio.h>
int main(){
        float nota1, nota2, nota3, media;
        int contador;
        contador = 1;
        do {
             printf("Digite as notas do aluno %d:",  contador);
             scanf("%f%f%f", &nota1, &nota2, &nota3);
             media = (nota1 + nota2+nota3)/3;
             printf("Media do aluno %d e : %f\n", contador, media);
             contador++;
        } while (contador <= 50);
        system("pause");
        return 0;
}
```



Estruturas de Repetição (**do..while**)

Exemplo:

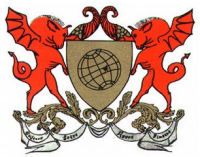
Escreva um programa que imprima a soma de dois números inteiros **positivos**, isto é, valide a entrada de dados apenas para números positivos, caso o usuário digite um número negativo, peça-o para repetir a entrada.



Estruturas de Repetição (**do..while**)

Exemplo:

```
#include <stdio.h>
int main(){
    int a, b;
    do{
        printf("Digite dois numeros positivos:\n");
        scanf("%d%d", &a, &b);
    }while(a <= 0 || b <= 0);
    printf("%d + %d = %d\n", a, b, a+b);
    system("pause");
    return 0;
}
```



Estruturas de Repetição (**do..while**)

Exemplo:

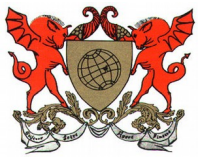
```
#include <stdio.h>
int main(){
    int a, b;
    do{
        printf("Digite dois numeros positivos:\n");
        scanf("%d%d", &a, &b);
    }while(a <= 0 || b <= 0);
    printf("%d + %d = %d\n", a, b, a+b);
    system("pause");
    return 0;
}
```

quando termina?

Estrutura de repetição com variável de controle embutida

Este tipo de estrutura de repetição é provavelmente o tipo mais usado dentre os programadores.

A estrutura **te obriga** a ter uma variável de controle e **te obriga** a controlar o seu intervalo (início e fim) e o incremento/decremento de uma forma bem simples.



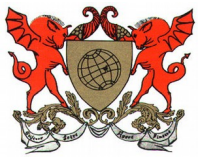
Estrutura de repetição com variável de controle embutida

Em geral é usada para um **"número definido"** de repetições:

- Comando **for**

Tem seu funcionamento controlado por uma **variável** que conta o número de vezes que o comando é executado.

Embora, também pode ser utilizada quando **não** se sabe esse número.



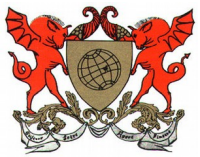
Estruturas de Repetição (**for**)

A condição também é verificada **antes** que o corpo do laço seja executado.

Forma Geral:

```
for (inicialização; condição; alteração) {
        instrução 1;
        ...
        instrução n;
}
```

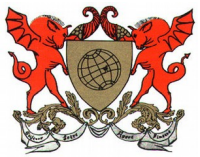
For significa "**para**". Em outras palavras, **para** uma variável alterando-se de um valor inicial até um valor final, repita a sequência de comandos.



Estruturas de Repetição (**for**)

```
for (inicialização; condição; alteração) {
        instrução 1;
        ...
        instrução n;
}
```

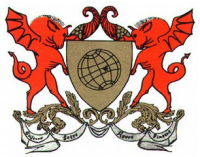
- Na primeira iteração é feito a **inicialização**, o teste da **condição** e se for verdadeira, a **execução dos comandos**.
- Sempre ao final da execução da <lista de comandos> é realizada a **alteração**.
- Na próxima iteração é feito o teste da **<condição>** novamente.



Estruturas de Repetição (**for**)

```
for (inicialização; condição; alteração) {
        instrução 1;
        ...
        instrução n;
}
```

- Quando a <condição> é avaliada, se ela for **verdadeira**, a <lista de comandos> é executada.
- Os comandos serão executados **enquanto** a condição for **verdadeira**. Se ela for **falsa**, a repetição para.
- Existe a possibilidade da <lista de comandos> **nunca ser executada**.

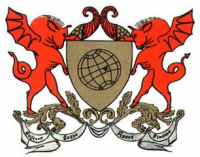


Estruturas de Repetição (**for**)

```
for (inicialização; condição; alteração) {
        instrução 1;
        ...
        instrução n;
}
```

- Na **inicialização** e na **alteração**, pode haver várias variáveis que devem estar separadas por **vírgula**.

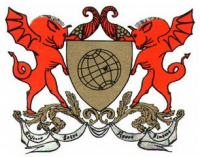
- No entanto, a **condição** deve ser única.



Estruturas de Repetição (**for**)

Exemplo:

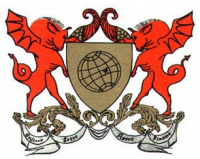
```
...
int i;
for (i = 1; i <= 5; i++) {
      printf("%d\n", i);
}
...
```



Estruturas de Repetição (**for**)

Exemplo(**Erro Comum**)::

```
...
int i;
for (i = 1; i <= 5; i++); {
      printf("%d\n", i);
}
...
```

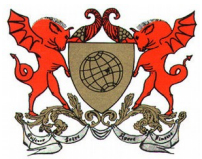


Estruturas de Repetição (**for**)

Exemplo:

```
...
int x, y;
for (x=1, y=5 ; x < y;  x+=2, y+=1) {
      printf("%d %d\n", x, y);
}
...
```

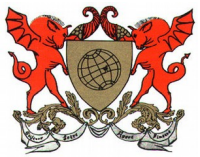
o que será impresso?



Estruturas de Repetição (**for**)

Exemplo:

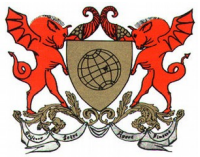
```
#include <stdio.h>
int main(){
     int up, down;
     for (up = 0, down=10; up < down; up++, down--){
          printf("up = %d, down= %d\n",up,down);
     }
     system("pause");
     return 0;
}
```



Estruturas de Repetição (**for**)

Exemplo: **Média das notas de 50 alunos**

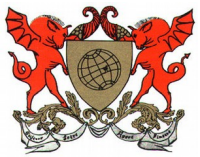
```
#include <stdio.h>
int main(){
        float nota1, nota2, nota3, media;
        int contador;
        for (contador = 1 ; contador <= 50 ; contador++) {
             printf("Digite as notas do aluno %d:",  contador);
             scanf("%f%f%f", &nota1, &nota2, &nota3);
             media = (nota1 + nota2 + nota3)/3;
             printf("Media do aluno %d e : %f\n", contador, media);
        }
        system("pause");
        return 0;
}
```



Estruturas de Repetição (**for**)

Exemplo:

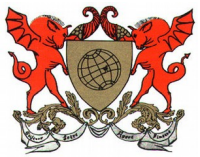
Escreva um programa em C que imprima os números no intervalo [1, n], sendo n um valor lido pelo programa.



Estruturas de Repetição (**for**)

Exemplo:

```
#include <stdio.h>
int main(){
    int i, n;
    printf("Digite um valor inteiro:\n");
    scanf("%d", &n);
    for(i=1; i<=n; i++){
      printf("%d ", i);
    }
    system("pause");
    return 0;
}
```



Estrutura de repetição com variável de controle embutida

- Exemplos:

Faça um algoritmo que leia 10 números inteiros e apresente na tela apenas aqueles que forem positivos.

Estrutura de repetição com variável de controle embutida

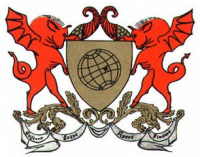
- Exemplos:

```
#include <stdio.h>
int main(){
   int i, n;
   for(i=1; i<=10; i++){
     printf(“Digite um valor inteiro:\n”);
     scanf(“%d”, &n);
     if(n > 0){
       printf("O numero digitado foi: %d\n", n);
     }
   }
   system("pause");
   return 0;
}
```

Estrutura de repetição com variável de controle embutida

- Exemplos:

Faça um algoritmo que mostre o resultado do somatório dos números de 1 até um valor N digitado pelo usuário.



Estrutura de repetição com variável de controle embutida

- Exemplos:

```
#include <stdio.h>
int main(){
        int i, num, soma;
        printf ("Digite um numero: ");
        scanf("%d", &num);
        soma = 0;
        for( i = 1;  i <= num; i++){
                soma = soma + i;
        }
        printf("A Soma de 1 até %d e: %d", num, soma);
       return 0;
}
```

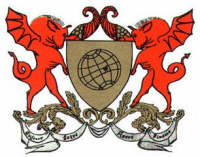
Estrutura de repetição com variável de controle embutida

- Exemplos:

Quais a modificações que devem ser feitas no algoritmo anterior para que o mesmo passe a calcular o **fatorial** de um número digitado pelo usuário?

lembrete:

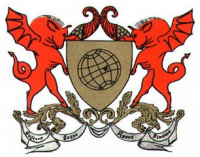
```
n! = 1 * 2 * 3 * 4 * ... * n-1 * n
```



Estrutura de repetição com variável de controle embutida

- Exemplos:

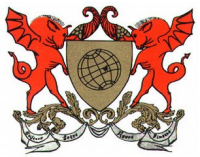
```
#include <stdio.h>
int main(){
        int i, num, soma;
        printf ("Digite um numero: ");
        scanf("%d", &num);
        soma = 1;
        for( i = 1;  i <= num; i++){
                soma = soma * i;
        }
        printf("O Fatorial de %d e: %d", num, soma);
       return 0;
}
```



Estruturas de repetição

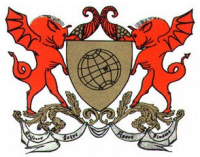
Existem diversas maneiras de implementar o mesmo **laço**, mas todo laço com variável de controle deve conter:

- inicialização da variável de controle
- incremento (aumento do valor) ou decremento (diminuição do valor) da variável de controle
- teste de valor da variável de controle



Estruturas de repetição

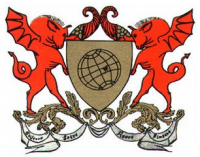
Um cuidado fundamental que o programador do algoritmo (**principalmente nós iniciantes**) deve ter é o de certificar-se que a condição para que sejam mantidas as iterações torne-se, em algum momento, **falsa ou verdadeira (depende da lógica usada)**, para que o algoritmo não entre em um **laço infinito**.



Estruturas de repetição

Exemplo: `loop (laço) infinito`

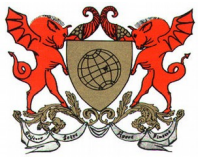
```
#include <stdio.h>
int main(){
    int contador = 0;
    while (contador != 10) {
        contador = 1;
        contador++;
    }
    return 0;
}
```



Estruturas de repetição

Exemplo:

```
#include <stdio.h>
int main(){
    int soma = 0;
    while (soma != 10) {
        soma = soma + 2;
    }
    return 0;
}
```



Estruturas de repetição

Exemplo:

```
#include <stdio.h>
int main(){
    int soma = 0;
    while (soma != 10) {
        soma = soma + 2;
    }
    return 0;
}
```

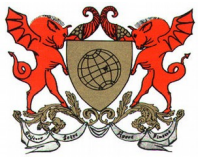
Soma: 0, 2, 4, 6, 8, 10

Estruturas de repetição

Exemplo:

Faça um algoritmo que escreva os números pares no intervalo de 2 a 10 em ordem decrescente.

- while
- do... while
- for

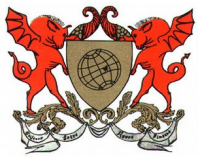


Comandos de Desvio Incondicional

Existem certas ocasiões que é preciso alterar o fluxo de controle de uma estrutura de laço antes do seu término.

Para isso, existem duas formas de se alterar o fluxo de controle:

- **break**;
- **continue**;

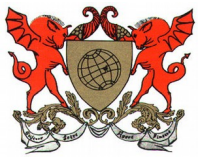


Comandos de Desvio Incondicional (**break**)

O comando **break** tem um papel semelhante ao exemplo do **switch**, isto é, ele encerra a estrutura de repetição por completo.

```
#include <stdio.h>
int main(){
        ...
        while (<condição>) {
                ...
            break;
            ...
        }
        ...
}
```

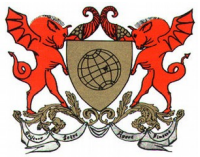
O fluxo de execução vai para o próximo comando depois do fim da repetição.



Comandos de Desvio Incondicional (**break**)

Exemplo:

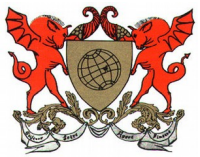
```
int main(){
        int x = 0;
        while (x < 5){
              printf("Valor de x: %d\n", x);
              if(x == 3){
                    printf("x igual a 3.\n");
                    break;
              }
              x = x + 1;
        }
        printf("Fora do laço while.\n");
}
```



Comandos de Desvio Incondicional (**break**)

Exemplo:

```
int main(){
    int x = 0;
    while (x < 5){
        printf("Valor de x: %d\n", x);
        x = x + 1;
        break;
        printf("Depois do incremento.\n");
    }
    printf("Fora do laço while.\n");
}
```



Comandos de Desvio Incondicional (**break**)

Obs:

O **switch** utiliza o comando **break** em sua estrutura, logo o que acontece se eu tiver um switch e break dentro de um laço??

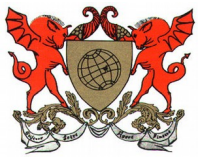
A resposta é simples, no caso se eu usar break dentro de um switch, o break só termina o switch e mas o laço continua normal.

Comandos de Desvio Incondicional (**continue**)

O comando **continue** força a próxima iteração do laço e pula o código que estiver abaixo dele.

```
#include <stdio.h>
int main(){
     ...
     while (<condição>){
          ...
        continue;
        ...
     }
     ...
}
```

A execução do programa vai diretamente para o teste condicional e depois continua o processo do laço.



Comandos de Desvio Incondicional (**continue**)

O comando **continue** força a próxima iteração do laço e pula o código que estiver abaixo dele.

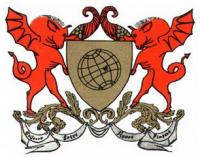
```
#include <stdio.h>
int main(){
    ...
    for(<ini>; <cond>; <alteração>){
        ...
        continue;
        ...
    }
    ...
}
```

No **for**, a execução vai para o incremento/decremento do laço, depois a condição, e finalmente faz com que o laço continue.

Comandos de Desvio Incondicional (**continue**)

Exemplo:

```
int main(){
    int x = 0;
    while (x < 5){
        printf("Valor de x: %d\n", x);
        x = x + 1;
        continue;
        printf("Depois do incremento.\n");
    }
    printf("Fora do laço while.\n");
}
```

Comandos de Desvio Incondicional (**continue**)

Exemplo:

```
int main(){
        int x = 0;
        while (x < 5){
            printf("Valor de x: %d\n", x);
            if(x == 3){
                printf("x igual a 3.\n");
                continue;
            }
            x = x + 1;
        }
        printf("Fora do laço while.\n");
}
```

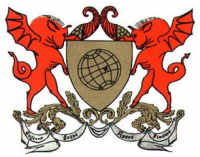
O que acontece?

Comandos de Desvio Incondicional (**continue**)

Obs:

O comando **continue** é usado de fato em situações muito específicas, isto é, apenas quando o programador julgar necessário.

No caso de usar **continue** dentro do **switch**, ele funciona normalmente voltando ao laço como foi dito anteriormente.

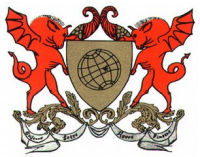


Laços aninhados

Assim como as estruturas de decisão aninhadas, a linguagem C não restringe o aninhamento de laços de repetição.

Isso é bom para facilitar a lógica de programação de muitos problemas.

No entanto, quando fizer isto, lembre-se de que qualquer comando **break** ou **continue** se aplica ao laço em que ele pertencer.

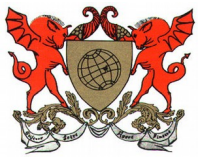


Laços aninhados

Como as estruturas de repetição são capazes de repetir qualquer comando, podemos repetir uma outra estrutura de repetição, que por si, repete uma série de comandos.

Exemplo:

```
for(i=0;i<10;i++){
    for(j=0;j<10;j++)
        printf("%d ",j);
    printf("\n");
}
```

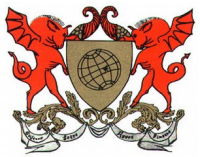


Laços aninhados

Como as estruturas de repetição são capazes de repetir qualquer comando, podemos repetir uma outra estrutura de repetição, que por si, repete uma série de comandos.

Exemplo:

```
int x = 0;
int y = 5;
while (x < 5){
   printf("Valor de x: %d", x);
   while (y > 0){
      printf("Valor de y: %d", y);
      y = y - 1;
   }
   x = x + 1;
}
```



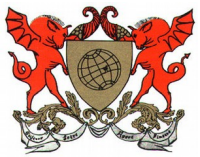
Laços aninhados

Exemplo:

Faça um programa que imprima na tela um quadrado de dimensões **n X n** formado apenas por uma letra qualquer. O tamanho do quadrado é digitado pelo usuário.

Qual seria a alteração para fazer um quadrado oco?

Altere o programa para fazer um retângulo n X m.



Exercícios

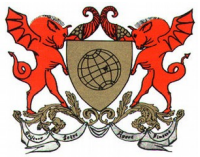
1) Faça um programa que calcule a área de um triângulo, cuja base e altura são fornecidas pelo usuário. Esse programa **não** pode permitir a entrada de dados inválidos, isto é, medidas menores ou iguais a 0.

2) (a) Faça um programa que pede para o usuário digitar um número e, então, testa se o número digitado é um número primo. (b) Adapte este programa para que ele possa testar a primalidade de vários números. Use o número 0 para indicar o término do programa.

Exercícios

3) Crie um programa em C que exiba o menu de opções abaixo. O programa deve receber como entrada vários números até que o valor 0 seja digitado. Calcule e mostre o resultado de cada uma das operações seguintes:

(1) Soma dos números digitados
(2) Quantidade de números digitados
(3) Média dos números digitados
(4) Maior número digitado
(5) Menor número digitado
(6) Média dos números pares

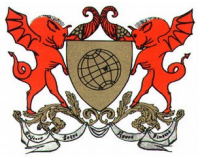


Exercícios

4) Faça um programa que receba um valor N inteiro e positivo, calcule o mostre o valor E, conforme a fórmula a seguir **E = 1 + 1/1! + 1/2! + 1/3! + ... + 1/N!**

5) Faça um programa que apresente os quadrados dos números inteiros de 15 a 200.

6) Faça um programa que determine e mostre os cinco primeiros múltiplos de 3, considerando números maiores do que 0.



Exercícios

7) Faça um programa que calcule e mostre a soma dos **50 primeiros números pares.**

8) Faça um programa que mostre na tela a tabuada completa da multiplicação dos números de 1 a 10.