

## Controle de Fluxo Estruturas de Repetição

Prof. Thiago Oliveira dos Santos  
Departamento de Informática  
Universidade Federal do Espírito Santo

2015

# Visão Geral da Aula

---

- Introdução
- Comandos de Iteração
- Comandos de desvio
- Comando Aninhados
- Buffers dos Comandos de Entrada e Saída

## Problema

- Vários problemas do dia a dia são possuem partes repetitivas

## Computadores

- Possuem alta capacidade de repetir tarefas
  - Essa é uma das razões de seu sucesso
- O controle das repetições é definido pelo programador
  - Através de um critério de parada

# Comandos de Iteração

---

## Características

- Permite que um número de comandos seja repetido
- Repetição ocorre enquanto uma condição é satisfeita

## Tipos

- Laço “while”
- Laço “do-while”
- Laço “for”

# Comandos de Iteração

## Comando de Repetição com precondição, “while”

- Equivalente ao “enquanto”

### Sintaxe

```
while(<condicao>)  
    <comando>;
```

- Ou, com bloco

```
while(<condicao>) {  
    <comandos>;  
}
```

### Semântica

- Verifica se a <condicao> é satisfeita (verdadeira)
  - Se sim, executa <comandos>
  - Se não, vai para o final do comando “while”

# Comandos de Iteração

## “while”

- Exemplo 1
  - Imprime os n primeiros números inteiros

```
#include <stdio.h>
int main() {
    int n, num;

    printf("Forneca 1 numero\n");
    scanf("%d", &n);

    num = 1;
    while (num <= n){
        printf("%d ", num);
        num = num + 1;
    }
    return 0;
}
```

**Muito Importante!**

Inicialização!

Condição de parada!

Incremento para parada!

# Comandos de Iteração

## “while”

- Exemplo 2

- Imprime os n primeiros números impares

```
#include <stdio.h>
int main() {
    int n, num, i;
    printf("Forneca 1 numero\n");
    scanf("%d", &n);

    i= 1;
    num = 1;
    while (i <= n){
        printf("%d ", num);
        num = num + 2;
        i= i + 1;
    }
    return 0;
}
```

# Conceitos Importantes

---

## Contadores

- Permitem contagem em laços **Muito Importante!**
  - Contar número de execuções cíclicas já concluídas
  - Contar número de ocorrências de um valor em uma sequência

## Acumuladores

- Permitem acumular valores em laços
  - Acumulação de resultados durante as iterações (ex. somatório)

## Mínimo e máximo

- Permitem identificar valores especiais em laços
  - Retenção do melhor resultado até o momento durante as iterações



# Comandos de Iteração

## “while”

- Exemplo de Contador
  - Obter a quantidade de números pares em uma sequência positiva

```
#include <stdio.h>
int main() {
    int numero, cont = 0;
    printf("Forneca um numero\n");
    scanf("%d", &numero);

    while (numero > 0){
        if ( !(numero % 2) ){
            cont = cont + 1;
        }
        printf("Forneca um numero\n");
        scanf("%d", &numero);
    }
    printf("A qtd eh: %d\n", cont);
    return 0;
}
```

# Comandos de Iteração

## “while”

- Exemplo de Acumulador
  - Obter a soma de uma sequência positiva

```
#include <stdio.h>
int main() {
    int numero, soma;

    printf("Forneca um numero\n");
    scanf("%d", &numero);

    soma = 0;
    while (numero > 0){
        soma = soma + numero;
        printf("Forneca um numero\n");
        scanf("%d", &numero);
    }
    printf("A soma eh %d\n", soma);
    return 0;
}
```

# Comandos de Iteração

## “while”

- Exemplo de Máximo
  - Obter o maior número de uma sequência positiva

```
#include <stdio.h>
int main() {
    int numero, maior=0;
    printf("Forneca um numero\n");
    scanf("%d", &numero);

    maior = numero;
    while (numero > 0){
        if (numero > maior){
            maior = numero;
        }
        printf("Forneca um numero\n");
        scanf("%d", &numero);
    }
    if (maior > 0){
        printf("O maior numero eh %d\n", maior);
    } else printf("Sem numeros\n");
    return 0;
}
```

# Comandos de Iteração

## “do-while”

- Equivalente ao “repita isso até que”
  - Continua enquanto a condição é verdadeira
- Ao contrario do “while”
  - Comandos executam pelo menos uma vez

## Sintaxe

```
do {  
    <comandos>;  
} while(<condicao>);
```

## Semântica

- Executa <comandos>;
- Verifica se a <condicao> é satisfeita
  - Se sim, executa <comandos> novamente
  - Se não, vai para o final do comando “do-while”

# Comandos de Iteração

## “do-while”

- Exemplo 1
  - Menu

```
#include <stdio.h>
int main() {
    int i;
    do {
        printf("1) Faca algo\n");
        printf("2) Faca algo diferente\n");
        printf("3) Saia\n");
        scanf("%d", &i);
        //Comandos para fazer algo ou
        //algo diferente entrariam aqui
    } while (i != 3);
    printf("Bye bye\n");
    return 0;
}
```

```
#include <stdio.h>
int main() {
    int i = 0;
    while (i != 3){
        printf("1) Faca algo\n");
        printf("2) Faca algo diferente\n");
        printf("3) Saia\n");
        scanf("%d", &i);
        //Comandos para fazer algo ou
        //algo diferente entrariam aqui
    }
    printf("Bye bye\n");
    return 0;
}
```

# Comandos de Iteração

---

## **“for”**

- Utilizado quando se conhece o número de repetições de antemão
- Permite agrupamento de comandos em blocos

## **Semântica**

- Faz <inicializacao> (somente uma vez)
  - Atribui valor inicial a variável
- Verifica se a <condicao> está satisfeita (verdadeira)
  - Se sim, executa <comandos>
  - Se não, vai para o final do comando “for”
- <incremento> permite mudar o valor da variável a cada repetição

# Comandos de Iteração

---

## Sintaxe

```
for(<inicializacao>; <condicao>; <incremento>)  
    <comando>;
```

- Ou, com bloco de comandos

```
for(<inicializacao>; <condicao>; <incremento>){  
    <comandos>;  
}
```

# Comandos de Iteração

---

## “for”

- Exemplo 1
  - Imprime números de 1 a 100 na tela

```
#include <stdio.h>
int main() {
    int x;
    for(x = 1; x <= 100; x = x + 1)
        printf("%d ", x);
    return 0;
}
```



# Comandos de Iteração

---

## “for”

- Exemplo 2
  - Imprime na tela os números de 100 a 66 com seus quadrados

```
#include <stdio.h>
int main() {
    int x, z;
    for(x = 100; x != 65; x = x - 1){
        z = x*x;
        printf("x = %d e z = %d\n", x, z);
    }
    return 0;
}
```

# Comandos de Iteração

---

## Variações do “for”

- Permite mais de um comando na <inicializacao> e no <incremento>

## Sintaxe

- for(<inicializacao1>, <inicializacao2>, ... <inicializacaoN>;  
<condicao>;  
<incremento1>, <incremento2>, ... <incrementoN>)  
    <comando>;

# Comandos de Iteração

---

## “for”

- Exemplo 1
  - Imprime na tela os números x de 100 a 0 e z de 0 a 100

```
#include <stdio.h>
int main() {
    int x, z;
    for(x = 100, z = 0; x >= 0; x--, z++)
        printf("x = %d e z = %d\n", x, z);
    return 0;
}
```

# Comandos de Iteração

---

## Uso do “For” como Temporizador

- Espera por algum tempo antes de continuar

```
#include <stdio.h>
int main() {
    unsigned int algumTempo = 1000000000, t;

    printf("Antes do for de tempo!");

    for(t = 0; t < algumTempo; t++) ;

    printf("Depois de algum tempo!");

    return 0;
}
```

# Comandos Aninhados

---

## Onde usar?

- Em situações em que existam sequências dentro de sequências

## Como usar?

- Colocando um comando de repetição dentro do bloco de outro

## Efeito

- Para cada item do comando de repetição externo
  - Ocorrerá novas repetições do comando de repetição interno

# Comandos Aninhados

---

## Exemplo de Uso

- Assumir uma disciplina com várias turmas, em que cada turma existem vários alunos que fizeram várias avaliações
- Fazer um programa para exibir a turma com maior percentual de aprovação na disciplina
- Os dados são fornecidos em lotes de:
  - Numero da turma 1
    - Matricula do aluno 1
      - Notas do aluno
    - Matricula do aluno 2
      - Notas do aluno
    - ...
      - ...
  - ...
- O programa para quando a nota, a matrícula e a turma forem respectivamente -1.0, -1 e -1

# Comandos Aninhados

---

## Exemplo de Uso

- Mostrar exemplo NotasDisciplina

# Comandos de Iteração

---

## Comando de Desvio

- Permitem ir para o começo ou final do laço
- Quando há aninhamento de laços
  - Só afeta o laço mais interno que o envolve

## Tipos

- break
- Continue



# Comando de Desvio

## “break”

- Pula diretamente para o final do laço
- Termina o laço prematuramente
- Permite finalização de laços infinitos
- Exemplo

```
int i = 0;
do {
    printf("1) Faca algo\n");
    printf("2) Faca algo diferente\n");
    printf("3) Saia\n");
    scanf("%d", &i);

    if (i == 3) break;

} while (1);
printf("Bye bye\n");
```

```
int i = 0;
for(;;) {
    printf("1) Faca algo\n");
    printf("2) Faca algo diferente\n");
    printf("3) Saia\n");
    scanf("%d", &i);

    if (i == 3) break;

}
printf("Bye bye\n");
```

# Comando de Desvio

## “break”

- Exemplo de break com laços aninhados

```
while(1) {  
    while(1) {  
        //...  
        break;  
    }  
}  
printf("Bye bye\n");
```

Para sair dos dois whiles  
pode-se usar uma flag

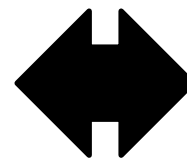
# Comando de Desvio

## “continue”

- Pula diretamente para o início
- Permite adiantar o laço sem executar os comandos
  - No “for”
    - O incremento é feito e a condição é re-testada
  - No “while”
    - A condição é re-testada
- Exemplo

```
int i = 0;
for(i=0;i<100;i++) {
    if (i % 10 == 0) continue;

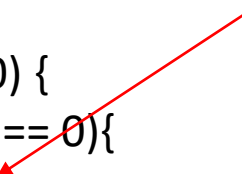
    printf("i = %d\n", i);
}
```



```
int i = 0;
while(i<100) {
    if (i % 10 == 0){
        i++;
        continue;
    }

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

**Cuidado para  
não esquecer!**



# Buffers dos Comandos de Entrada e Saída

---

## Geral

- Funcionam com mecanismo de buffer
- O buffer faz a intermediação dos dados

## Funcionamento do *printf*

- Joga os dados das variáveis para o buffer
- Joga os dados para um dispositivo de saída (padrão)

## Funcionamento do *scanf*

- Olha se tem dados no buffer e espera se tiver vazio
  - Pega os dados de um dispositivo de entrada (padrão)
- Coloca os dados lidos nas variáveis

# Buffers dos Comandos de Entrada e Saída

## “while”

- Exemplo com caracteres
  - Espera por uma determinada letra, ‘a’

```
char l = '\0';  
while (l != 'a'){  
    scanf("%c", &l);  
}  
printf("Finalmente voce digitou a!");
```

**Cuidado!**  
**Inicialização**  
**necessária!**

- Exemplo com caracteres

```
char l = '\0';  
while (l != 'a'){  
    printf("Digite uma letra!");  
    scanf("%c", &l);  
}  
printf("Finalmente voce digitou a!");
```

# Buffers dos Comandos de Entrada e Saída

## “while”

- Exemplo com caracteres
  - Espera por uma determinada letra, ‘a’

```
char l = '\0';  
while (l != 'a'){  
    scanf("%c", &l);  
}  
printf("Finalmente voce digitou a!");
```

**Cuidado!**  
**Inicialização**  
**necessária!**

- Exemplo com caracteres

```
char l = '\0';  
while (l != 'a'){  
    printf("Digite uma letra!");  
    scanf("%c", &l);  
}  
printf("Finalmente voce digitou a!");
```

```
//Leitura melhor  
scanf("%c", &l);  
scanf("%*[^\\n]");  
scanf("%*c");
```

# Scanf com While

---

- Le inteiros enquanto não encontrar algo diferente de número
  - White-spaces (spaces, newline and tab characters) são desconsiderados (ver o help do scanf para maiores detalhes)
- Scanf retorna o número de itens lidos para a lista de variáveis
- Leitura falha se não encontrar o padrão procurado
- Exemplo com scanf
  - Sai do while quando encontra algo diferente de número ou white-space

```
int n;  
  
while ( scanf("%d", &n) == 1 ){  
    ...  
}
```

# Pergunta???



**UFES**  
Informática

- 
- Fazer exercícios da lista 2!