AULA 09 – PARADIGMAS DE PROGRAMAÇÃO
01/04/2013

PROFESSOR GLAUBER FERREIRA CINTRA
SEMESTRE 2012.2 – ENGENHARIA DE COMPUTAÇÃO IFCE

ESTRUTURAS DE CONTROLE

O fluxo natural de execução de um programa é sequencial. No entanto, em diversas situações, o fluxo sequencial em execução não atende as nossas necessidades. Às vezes precisamos desviar de certos trechos do programa. Outras vezes precisamos repetir a execução de partes do programa. Para isso, fazemos uso de estruturas de controle.

As estruturas de controle são dividas em duas categorias:

- Estruturas de **desvio**;

- Estruturas de **repetição**;

Desvios

Podem ser *condicionais* ou *incondicionais*.

Desvios incondicionais

- Goto (“vá para”): Desvia o fluxo de execução para uma determinada instrução.

Exemplo:

```
Leitura;

scanf(“%d”, &sexo);

if(sexo !=’M’ && sexo !=’F’){

     goto Leitura;

}
```

- Next (ou continue): Desvio para o início do laço, dando início a uma nova iteração do laço.

- Redo: Desvia para o início do laço, reiniciando a iteração corrente do laço.

- Break: Desvia o fluxo de execução para a primeira instrução após o laço Switch que o contém. O uso do break dentro do switch é aceitável.

- Return: Finaliza a execução do subprograma se for uma especificada expressão após o return, seu valor será devolvido para o chamador do subprograma. O uso do return é aceitável.

Desvios condicionais

- Se

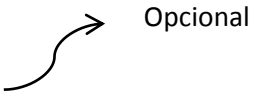
Sintaxe:

Se <condição>

<bloco 1>

Se não

<bloco 2>



A condição é verificada. Se for verificada o bloco 1 é executado. Caso contrário apenas o bloco 2 é executado.

Exemplo 1: Seleção Simples

```
if(x < y){
      swap(x, y);
}
```

Exemplo 2: Seleção dupla

```
if(x < y){
      swap(x, y);
}else{
      x--;
      y++;
}
```

Exemplo 2: Seleção múltipla

```
if(EC == 'S'){
     printf("Solteiro");
}else{
     if(EC == 'S'){
          printf("Casado");
     }else{
          if(EC == 'D'){
               printf("Divorciado");
          }else{
               printf("Outro");
          }
     }
}
```

A estrutura apresentada nesse último exemplo aparece com frequência nos programas e é conhecido como **case**.

Sintaxe:

case

<condição 1> <bloco 1>

<condição 2> <bloco 2>

.
.
.
<condição n> <bloco n>

[<Bloco default>]

As condições são verificadas na ordem em que aparecem no case. Se cada uma delas for verdadeira, o bloco correspondente é executado e o fluxo de execução é desviado para a primeira instrução após o case. Se o bloco default for alcançado ele será executado.

Exemplo:

```
switch(EC){
     case ‘S’: printf(“Solteiro”); break;
     case ‘C’: printf(“Casado”); break;
     case ‘V’: printf(“Viuvo”); break;
     case ‘D’: printf(“Divorciado”); break;
     default: printf(“Outro”);
}
```

Laços condicionais

- Enquanto

Sintaxe: Enquanto <condição> <bloco>

No início de cada iteração a condição é verificada, se for verdadeira, o bloco é executado e uma iteração é novamente iniciada. Se for falso o fluxo de execução é desviado para a primeira iteração após o laço.

Exemplo:

```
i = 1;
while(i<=10){
     printf(“%d ”, i);
     i++;
}
```

- Faça enquanto (do...while)

Sintaxe: Faça <bloco> Enquanto <condição>

Similar ao enquanto, sendo que a condição é verificada no final de cada iteração.

Exemplo:

```
i = 0;
do{
     printf(“%d ”, i);
```

```
    i++;
} while(i<=10);
```

Laços contados

- Faça (for)

Sintaxe: Para <contador > = <início> até <fim> [passo n]

Para cada valor entre início e fim será executada uma iteração do laço. Em linguagem C, o para tem a seguinte sintaxe:

for(<EXP1>;<EXP2>;<EXP3>)

<bloco>

Antes da primeira iteração a EXP1 é avaliada no início de cada iteração, a EXP2 é avaliada e se for verdadeira o bloco é executado, a EXP3 é computada e uma nova iteração é iniciada. Se falsa, o laço é finalizado.

Exemplo:

```
for(i=1;i<=10;i++) printf("%d", i);

for(i=1;i<=10; printf("%d", i++)) ;

for(;;) printf("Tempo");
```