

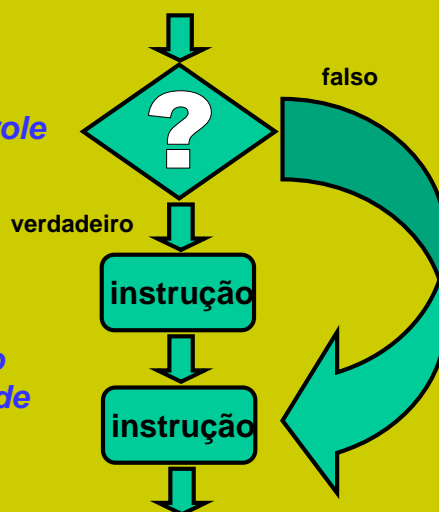
Laboratório de Programação I

Estruturas de Controle em C

Estruturas de Controle

Uma **estrutura de controle** é uma **instrução de controle** e sua coleção de comandos cuja execução ela controla.

O exemplo ao lado ilustra um desvio de fluxo de execução simples.



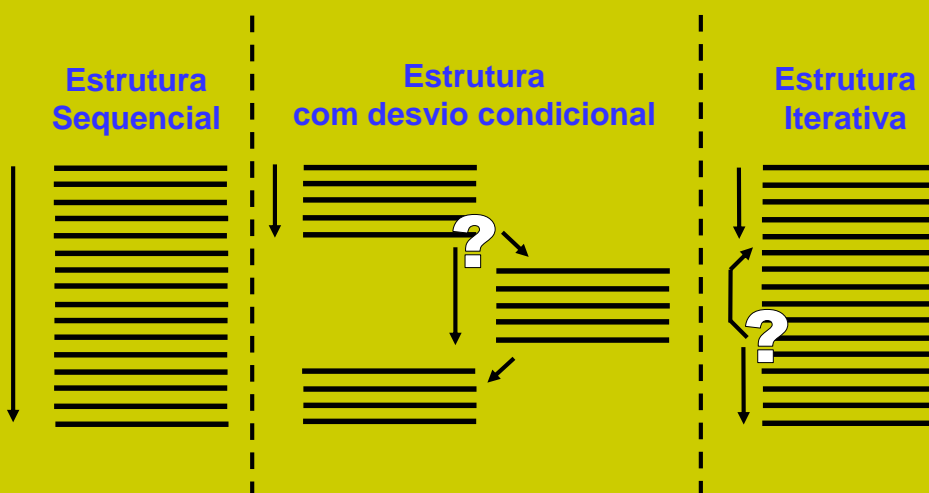
Estruturas de Controle

As computações em programas de linguagens imperativas são realizadas avaliando-se expressões e atribuindo-se os valores resultantes a variáveis. Entretanto, há somente alguns programas úteis que consistem inteiramente em instruções de atribuição.

Pelo menos dois mecanismos lingüísticos adicionais são necessários para tornar flexíveis e poderosas as computações em programas:

- ⇒ Meios de selecionar entre caminhos de execução.
- ⇒ Meios de provocar a execução repetida de certos conjuntos de instruções.

Estruturas de Controle



Instruções de Seleção

Selecionando os caminhos de execução em um programa

- ⇒ Instruções compostas
- ⇒ Desvios incondicionais
- ⇒ Seleção Unidirecional
- ⇒ Seleção Bidirecional
- ⇒ Aninhando Seletores
- ⇒ Exercício de Fixação
- ⇒ Seleção Múltipla
- ⇒ Exercícios de Fixação

Instruções Compostas

Um dos recursos de linguagem que ajuda a tornar o projeto de instruções mais fácil é um método para formar coleções de instruções.

```
C ou C++  
{  
    <instrução 1>;  
    .  
    .  
    .  
    <instrução n>;  
}
```

Seleção Unidirecional

Todas as linguagens imperativas incluem um seletor unidirecional. O seletor unidirecional do C, chamado instrução lógica IF, tem a forma:

IF (<expressão booleana>) <instrução>

Neste caso apenas um comando está sujeito à condição do comando IF.

Seleção Unidirecional

Outra forma de se vincular mais de uma instrução em um comando de seleção é utilizar instruções compostas. Abaixo está descrita a sintaxe para uma seleção bidirecional em C/C++:

```
if (<expressão booleana>) {  
    { <instrução> }  
} else {  
    { <instrução> }  
}
```

Caso o resultado da expressão booleana seja verdade a 1ª instrução será executada e a 2ª instrução não. E caso seja falsa a 1ª não será executada e a 2ª sim.

*No lugar destas instruções poderiam ser colocadas **instruções compostas** para que mais de um comando estivesse vinculado ao comando if em cada situação.*

Seleção Bidirecional

Veja um exemplo de utilização de uma seleção Bidirecional e instruções compostas:

```
#include <stdio.h>
void main() {
    int x, totPAR=0, totIMPAR=0;
    printf("Entre com um número:");
    scanf("%d");
    if (x % 2){
        printf("O número é par!");
        totPAR += x;
    } else {
        printf("O número é ímpar!");
        totIMPAR += x;
    }
    printf("Total par:%d, total ímpar:%d",totPAR,totIMPAR);
}
```

Aninhando Seletores

É possível colocar um seletor como instrução de um outro seletor. Isto pode gerar dúvidas dependendo da construção do programa.

Pode haver duas interpretações deste programa, uma que a cláusula else é referente ao primeiro if e a outra é que ela seja referente ao segundo if (exemplo A).

Muitas linguagens implementam que o else estará vinculado ao comando if ... then não-emparelhada mais recentemente, portanto, no nosso exemplo seria ao 2º if. Porém pelo fato de este ser um fator que pode gerar dúvidas no momento da leitura do código, é recomendável que seja explicitado o vínculo da cláusula else com seu respectivo if ... then através da utilização de instruções compostas (exemplo B).

Aninhando Seletores

A

```
if (soma == 0)
    if (cont == 0)
        resultado = 0;
else
    resultado = 1;
```

B

```
if (soma == 0) {
    if (cont == 0)
        resultado = 0
}
else
    resultado = 1;
```

Exercícios de Fixação

1) Escreva um programa em C que utilize três seletores aninhados cada um com uma expressão booleana diferente, e que o 2º seletor seja bidirecional.

Se a expressão lógica do 1º seletor for verdadeira imprima na tela “**seletor 1 verdade!**”; se a expressão do 2º seletor for verdadeira imprima na tela “**seletor 2 verdade!**”; se a expressão lógica do 3º seletor for verdadeira imprima na tela “**seletor 3 verdade!**”; e se a expressão lógica do 2º seletor for falsa imprima “**seletor 2 falso!**”. Caso contrário nada deverá ser impresso.

Seleção Múltipla

De acordo com o resultado da expressão inteira uma das instruções será executada. Se o resultado for 1, a primeira instrução é executada, se for 2 a segunda e assim por diante.

A seleção múltipla em C tem a seguinte forma:

```
switch (<expressão>) {  
    { case <expressão constante 1>: { <instrução 1> } }  
    default:  
        { <instrução> }  
}
```

Seleção Múltipla

*Um problema com esta estrutura é que os blocos de instruções em cada “case” iniciam sua execução e terminam ao final do switch, então neste caso deveremos utilizar o comando **break** que desviará a execução do código para a próxima instrução depois do fim do bloco.*

*Introduzindo o comando **break** ao final de cada bloco de instruções do comando **switch** teremos como resultado:*

```
switch (<expressão>) {  
    { case <expressão constante>: { <instrução> } ; break; }  
    default:  
        <instrução>  
}
```

Exercícios de Fixação

- 1) *Escreva um programa em C, utilizando o comando switch, que dado o número de um mês, ele retorne a quantidade de dias supondo que fevereiro tenha sempre 28 dias.*
- 2) *Altere o seu programa para que este leve em conta anos bissextos. Sabendo-se que um ano é bissexto quando o ano for divisível por 4 e não por 100 ou então que o ano seja divisível por 400.*

Estrutura de Controle

Instruções Iterativas

Estruturas de Controle

Instruções Iterativas

Uma instrução iterativa faz com que uma ou mais instruções sejam executadas zero, uma ou mais vezes. A iteração é a própria essência do poder dos computadores. Sem este recurso, programas úteis teriam um código-fonte muito grande e demorariam muito para serem escritos.

Existem algumas formas de se implementar instruções iterativas, e estas formas podem ser categorizadas em alguns aspectos:

- Como a iteração é controlada*
- Onde o mecanismo de controle deve aparecer no laço*

Estruturas de Controle

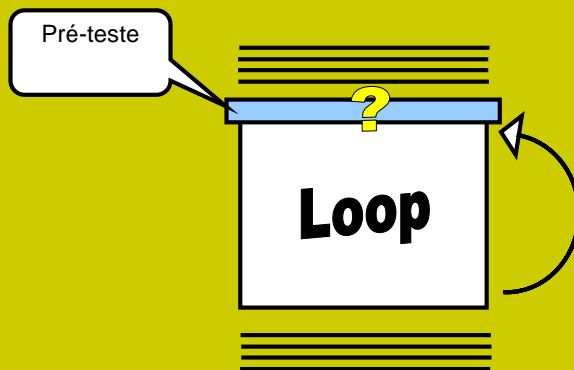
Instruções Iterativas

As principais possibilidades de controle de iteração são: lógica; contagem; ou a combinação das duas. E as possibilidades para a localização do mecanismo de controle são a parte superior e a parte inferior do laço.

*O **corpo** de um laço é a coleção de instruções cuja instrução de iteração controla a execução. Utilizamos o termo **pré-teste** querendo dizer que o teste para a finalização do laço ocorre antes que o corpo do laço é executado. E **pós-teste** significa que ele ocorre depois que o corpo do laço é executado. A instrução de iteração e o corpo do laço associado formam, juntos, uma **construção de iteração**.*

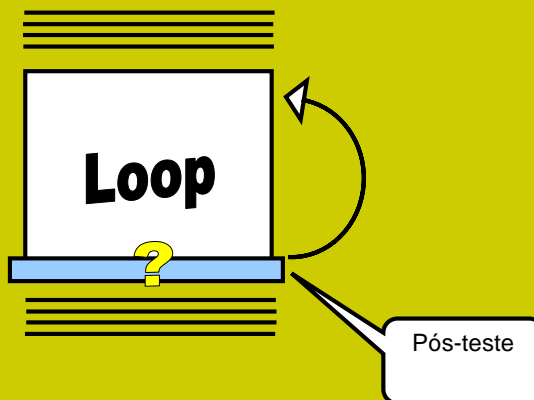
Estruturas de Controle

Instruções Iterativas lógicas com pré-teste



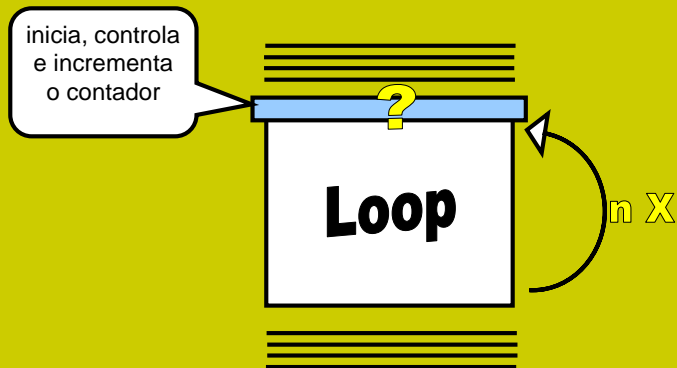
Estruturas de Controle

Instruções Iterativas lógicas com pós-teste



Estruturas de Controle

Instruções Iterativas com contagem



Estruturas de Controle

Laços Controlados por Contador

Uma instrução iterativa de controle de contagem tem uma **variável de laço**, na qual o valor da contagem é mantido. Ela também possui alguns meios de especificar os valores **inicial** e **terminal** da variável de laço e a diferença entre seus valores sequenciais, freqüentemente chamados **stepsize**. As especificações iniciais, as terminais e os **stepsize** de um laço são chamados de **parâmetros do laço**.

Estruturas de Controle

Laços Controlados por Contador

Questões de projeto

- Qual é o tipo e o escopo da variável de laço?*
- Que valor a variável de laço tem na sua finalização?*
- Deve ser legal que a variável de laço ou os seus parâmetros sejam mudados no laço, e se assim for, a mudança afeta o seu controle?*
- Os parâmetros de laço devem ser avaliados somente uma vez para cada iteração.*

Estruturas de Controle

Laços Controlados por Contador

Muitas linguagens implementaram suas instruções iterativas de contagem de formatos distintos entre si, porém o objetivo de todas elas é executar uma ou mais instruções um certo número de vezes.

Estruturas de Controle

Laços Controlados por Contador

Na linguagem C ou C++ a forma geral é:

```
for (<expressão1>; <expressão2>; <expressão3>)  
    <corpo do laço>
```

O corpo do laço pode ser: uma instrução única; uma instrução composta ou uma instrução nula.

A primeira expressão é avaliada uma única vez quando se inicia o loop. Normalmente é utilizada para inicializações.

A segunda expressão é o controle do laço avaliada antes de cada execução do corpo do mesmo.

A terceira expressão é executada depois da execução do corpo do laço. Normalmente utilizada para incrementos de variáveis.

Estruturas de Controle

Laços Controlados por Contador

Exemplo típico da utilização do laço de contagem:

```
for (indice = 0; indice <= 10; indice++)  
    soma = soma + lista[indice];
```

Pode-se utilizar instruções múltiplas no caso da linguagem C e C++. Veja o exemplo:

```
for ( cont1 = 0, cont2=0.0;  
      cont1 <= 10 && cont2 <= 100.0;  
      soma = ++cont1 + cont2, cont2 *= 2.5);
```

Estruturas de Controle

Laços Controlados Logicamente

Muitas vezes não sabemos exatamente a quantidade de vezes que um determinado laço será executada, portanto, a utilização de laços controlados por contador não são os mais adequados.

Nestes casos, são utilizados os laços controlados logicamente. A continuidade da repetição do laço depende de uma expressão lógica.

Estruturas de Controle

Laços Controlados Logicamente

Questões de projeto

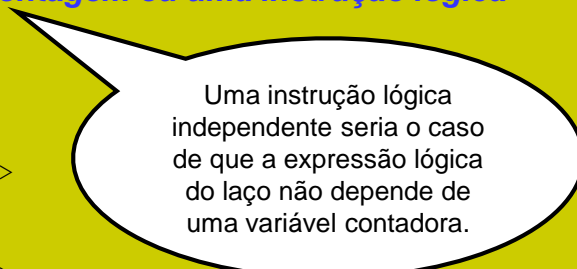
- ✓ O controle deverá ser de pré ou pós teste?
- ✓ O laço controlado logicamente deverá ser uma forma especial de laço de contagem ou uma instrução lógica independente?

Exemplos:

```
while (<expressão>)  
    <corpo do laço>
```

ou

```
do  
    <corpo do laço>  
while (<expressão>)
```



Uma instrução lógica independente seria o caso de que a expressão lógica do laço não depende de uma variável contadora.

Estruturas de Controle

Laços Controlados Logicamente

Localização do controle do laço

Quando devemos colocar o controle do laço no início ou no final do corpo? Ao colocar o controle no início, o teste será efetuado antes da execução do corpo, e por isso mesmo, corre o risco de não ser executado nenhuma vez. No caso de colocar o controle na base, o corpo do laço será executado pelo menos uma vez.

Existem situações em que é conveniente para o programador posicionar o controle do laço no interior do corpo ao invés de no início ou na base. Por isso, algumas linguagens oferecem essa capacidade.

Estruturas de Controle

Laços Controlados Logicamente

O C e o C++ possuem o comando `continue` que desvia a execução do programa da sua localização dentro do corpo do laço até o final do corpo. Este comando não encerra a execução do loop.

No caso do comando `break` a execução do código também é desviada para o final do corpo do laço, porém, o laço será terminado.

Estruturas de Controle

Laços Controlados Logicamente

Os exemplos abaixo ilustram a utilização do **continue** e do **break**. No primeiro caso um valor negativo que seja lido a partir da função **getnext** será ignorado e portanto, não acumulado na variável **soma**. No segundo caso um valor negativo encerra o loop.

1º

```
while (soma < 1000) {  
    getnext(valor);  
    if (valor<0) continue;  
    soma+=valor;  
}
```

2º

```
while (soma < 1000) {  
    getnext(valor);  
    if (valor<0) break;  
    soma+=valor;  
}
```

Estruturas de Controle

Exercício de Fixação

1) Construa um programa que calcule o resultado da seguinte série:

$$Y = 1/x + 1/(x+1) + 1/(x+2) + 1/(x+3) + \dots + 1/(x+1000)$$

2) Construa um programa que alimente com valores inteiros uma matriz **NxM** de tamanho máximo 20x20.