Algoritmos e Programação

Linguagem C
Estruturas de Controle -

Eliane Pozzebon

Estruturas de Controle

- Estruturas Condicionais ou de Seleção
  - Simples
  - Composta
  - Múltipla escolha
- Estruturas de Repetição
  - Por condição no início
  - Por condição no fim
  - Por contagem

Estrutura Condicional ou de Seleçã

- Realizam a execução de um conjunto de comandos de acordo com o valor de uma expressão.

- Servem para alterar o fluxo de execução de um programa baseado no valor, verdadeiro ou falso, de uma expressão lógica.

Estrutura Condicional Simples :: Comando `if`

- Na linguagem C, a estrutura condicional simples possui a seguinte sintaxe:

```
if (<expressão>)
{
    <comandos>
}
```

Estrutura Condicional Composta :: Comando `if-else`

- Na linguagem C, a estrutura condicional composta possui a seguinte sintaxe:

```
if (<expressão>)
{
        <comandos>
}
else
{
        <comandos>
}
```

Estrutura Condicional Composta :: Comando `if-else`

- Quando temos estruturas condicionais presentes em blocos de comandos de outras estruturas condicionais, chamamos de estruturas aninhadas ou encadeadas:

```
if (<expressão>)
{
        if (<expressão>)
        {
                <comandos1>
        }
        else
        {
                <comandos2>
        }
}
else
{
        if (<expressão>)
        {
                <comandos3>
        }
        else
        {
                <comandos4>
        }
}
```

Estrutura Condicional Composta :: Comando `if-else`

- Ambigüidade: na falta de chaves, a linguagem C associa o **`else`** sempre ao último **`if`**.
- Portanto, para que a construção funcione como desejado, pode-se resolver o problema usando bloco.

```
x = 2;
if (x < 2)
        if (x >= -3)
                printf("\n x está entre [-
3;1]");
else printf("x é maior que 1");
```

```
x = 2;
if (x < 2)
{
        if (x >= -3)
                printf("\n x está entre [-
3;1]");
}
else printf("x é maior que 1");
```

Operador Condicional: ?

- O operador ternário **?** possui a seguinte sintaxe:

```
Exp1 ? Exp2 : Exp3;
```

- **Exp1** é avaliada:
  - Se for verdadeira, **Exp2** é avaliada se torna o seu valor da instrução.
  - Se for falsa, **Exp3** é avaliada se torna o seu valor da instrução.

Operador Condicional: ?

- Equivale ao seguinte código **if-else**:

```
Exp1 ? Exp2 : Exp3;
```

```
if (Exp1)
      Exp2;
else
      Exp3;
```

Exercício

- Considere um par de valores (x, y)
- Eles representam as coordenadas de um ponto no plano
- Determinar:
  - o quadrante ao qual pertence o ponto, ou
  - se está sobre um dos eixos cartesianos

Estrutura Condicional de Múltipla E
:: Comando switch-case

- Instrução de seleção de múltipla escolha CASE na linguagem C:

```
switch (<expressão_ou_valor>)
{
   case Constante1:
            <comandos1>
            break;
   case Constante2:
          <comandos2>
            break;
   ...
   default:
          <comandosN>
}
```

Estrutura Condicional de Múltipla E
:: Comando `switch-case`

- Funcionamento do comando **`switch`**:
  - Analisa a expressão ou valor.
  - Caso seja igual a Constante1, então executa "comandos1" e vai para o final do **`switch`**.
  - Caso seja Constante2, então executa "comandos2" e vai para o final do **`switch`**.
  - Assim sucessivamente...
  - Caso não seja nenhuma das alternativas, vai para o **`default`**, executa "comandosN" e encerra o **`switch`**.

Estrutura Condicional de Múltipla E
:: Comando `switch-case`

- O parâmetro do **`switch`** deve ser **`int`** ou **`char`**.
- O valor após o **`case`** dever ser uma constante (não pode ser uma variável).
- Instrução **`break`**:
  - Faz com que aconteça a saída imediata daquela estrutura (no caso, o comando **`switch`**).
  - A execução do programa continua com a primeira instrução depois da estrutura.
  - Sua ausência faz o controle passar ao próximo **`case`**, o que não é desejado.

Exercício

- Os funcionários de uma empresa receberam um aumento de salário:
  - Técnicos: 50%
  - Gerentes: 30%
  - Demais: 20%

- Escrever um programa que receba como entrada o salário atual e o cargo do funcionário, e apresente o novo salário após o aumento.

Repetição por condição no início :: Comando while

- Permite que um certo trecho de programa seja executado ENQUANTO uma certa condição for verdadeira.

```
while (<condição>)
{
      <comandos_a_repetir>
}
<comandos_após_laço_while>
```

Repetição por condição no início :: Comando `while`

- Funcionamento:
  1. Testar a condição.
  2. Se a condição for falsa, então sair do laço.
  3. Se a condição for verdadeira, então executar cada um dos comandos do interior do laço.
  4. Após executar o último comando do laço *while*, voltar ao passo 1.

Repetição por condição no início :: Comando `while`

- O **`while`** deve ser usado sempre que:
  - Não soubermos exatamente quantas vezes o laço deve ser repetido
  - O teste deva ser feito antes de iniciar a execução de um bloco de comandos
  - Houver casos em que o laço não deva ser repetido nenhuma vez antes do teste da condição
- Atenção:
  - As variáveis usadas no teste devem ser inicializadas antes do teste.

Repetição por condição no fim :: Comando do-while

- Permite que um certo trecho de programa seja executado ATÉ QUE uma certa condição deixe de ser verdadeira:

```
do
{
      <comandos_a_repetir>
} while (<condição>);

<comandos_após_laço_do-while>
```

Repetição por condição no fim :: Comando `do-while`

- Funcionamento:
  1. Executar os comando dentro do bloco **`do-while`**.
  2. Testar a condição.
  3. Se a condição for falsa, então sair do laço.
  4. Se a condição for verdadeira, então retornar para o passo 1.

Repetição por condição no fim :: Comando `do-while`

- O comando **`do-while`** deve ser usado sempre que:
  - Não soubermos exatamente quantas vezes o laço deve ser repetido
  - O teste deva ser feito depois da execução de um bloco de comandos
  - O bloco de comandos deve se executado pelo menos uma vez

Exercícios

- Ler um número e verificar se este é ou não um número primo.

- Dada uma dívida de 10 mil reais que cresce a juros de 2,5% ao mês e uma aplicação de 1.500 reais com rendimento de 4% ao mês, escrever um algoritmo que determine o numero de meses necessários para pagar a dívida.

Repetição por contagem :: Comando for

- Permite que um certo trecho de programa seja executado um número determinado de vezes.

```
for (comandos_inic; condição_teste; incre/decremento)
{
      <comandos_a_repetir>
}

<comandos_após_laço_for>;
```

Repetição por contagem :: Comando `for`

- Funcionamento:
  1. Executar os comandos de inicialização.
  2. Testar a condição.
  3. Se a condição for falsa então sair do laço **`for`**.
  4. Se a condição for verdadeira então executar os comandos que estão subordinados ao **`for`**.
  5. Executar os comandos de incremento/decremento.
  6. Voltar ao passo 2.

Repetição por contagem :: Comando `for`

- Observações:
  - Os comandos de inicialização são executados apenas uma vez.
  - O contador é incrementado/decrementado sempre ao final da execução do bloco.
  - O teste é feito sempre antes do início da execução do bloco de comandos.

Repetição por contagem :: Comando `for`

- O comando **`for`** deve ser usado sempre que:
  - Soubermos exatamente quantas vezes o laço deve ser repetido.
  - O teste deva ser feito antes da execução de um bloco de comandos.
  - Houver casos em que o laço não deva ser repetido nenhuma vez.

Repetição por contagem :: Comando `for`

- Flexibilidade do comando **`for`**
  - Se nenhuma inicialização é necessária, **`comandos_inic`** pode ser omitido.

```
for ( ; condição_teste; incre/decremento)
```

Repetição por contagem :: Comando `for`

- Flexibilidade do comando **`for`**
  - Se **`condição_teste`** é omitida, o comando **`for`** pode ficar em um laço infinito.

```
for (comandos_inic;  ; incre/decremento)
```

Repetição por contagem :: Comando `for`

- Flexibilidade do comando **`for`**
  - Se **`incre/decremento`** é omitido, deve-se, atualizar a variável de laço em um comando dentro do laço.

```
for (comandos_inic; condição_teste; )
```

Repetição por contagem :: Comando `for`

- Flexibilidade do comando **`for`**
  - Duas ou mais variáveis (separadas por vírgula) podem ser inicializadas e/ou incre/decrementadas ao mesmo tempo.

Repetição por contagem :: Comando `for`

- Comandos **`for`** aninhados
  - Um comando **`for`** pode ser utilizado dentro de outro comando **`for`**.
  - Cuidados devem ser tomados para não aumentar o tempo de execução do algoritmo.

Repetição por contagem :: Comando `for`

- Cálculo de séries
  1. Determinar o termo geral.
  2. Determinar valor inicial do contador.
  3. Determinar valor final do contador.

Repetição por contagem :: Comando `for`

- Exercícios

1. Calcular o valor de S:

$$S = \frac{2}{50} + \frac{2^2}{48} + \frac{2^3}{46} + \Lambda + \frac{2^{25}}{2}$$

2. Calcular o valor de π com precisão de 0,001:

$$S = 1 - \frac{1}{3^3} + \frac{1}{5^3} - \frac{1}{7^3} + \Lambda \ , \quad \pi = \sqrt[3]{S \times 32}$$

Referências:

Material do Prof. Leandro Galvão da Universidade Federal de Manaus

Livros:

- Fundamentos da programação de computadores
- C a linguagem de programação padrão ANSI
- C completo e total