Construção de Algoritmos e Programação

Aula 10 - 30/04/2025

Joice Otsuka
joice@ufscar.br

Programação C - Condicional

Estruturas condicionais

Comando if-else

```
if (condição) {
        Bloco caso verdadeira;
}
else {
        Bloco caso falsa;
}
```

Comando if-else

- Todo comando if-else requer **uma condição** que pode ser verdadeira ou falsa.

- Em C, não existe um tipo de dados específico para representar valores lógicos (V ou F).

- **Qualquer valor diferente de zero é interpretado como verdadeiro, enquanto zero é falso.**

Comando if-else

```
if (condição)
    comando;
else
    comando;
```

```
if (condição){
    comando;
    comando;
}
else {
    comando;
    comando;
}
```

- Um bloco de instruções começa com o símbolo { e termina com o símbolo }. Caso o bloco contenha apenas uma instrução, as chaves são opcionais.

Comando if-else

- Exemplo:

```
if (delta >=0)
{
    x1 = (-b + sqrt(delta))/(2*a);
    x2 = (-b - sqrt(delta))/(2*a);
}
else
{
    printf("Sem raízes reais.");
}
```

Comando if-else

- Qualquer instrução pode fazer parte de um conjunto de instruções, inclusive um comando if-else.

```
if (delta >=0)
{
    x1 = (-b + sqrt(delta))/(2*a);
    if (delta == 0)
        x2 = x1;
    else
        x2 = (-b - sqrt(delta))/(2*a);
}
else
{
    printf("Sem raízes reais.");
}
```

A importância dos recuos

- Programas mais complexos são mais difíceis de ler e compreender.
- Uma forma de melhorar a legibilidade do programa é usar recuos.
- Os recuos devem ser usados sempre após o símbolo {, sendo as instruções recuadas à direita.
- O símbolo } deve estar alinhado ao abre-chaves correspondente.

Recuos não resolvem ambiguidades

- Exemplo:

```
if (nota >= 9)
   if (nota_anterior < nota)
      printf("Você está melhorando.");
else
   printf("Sem estudo é difícil ser aprovado.");
```

- De quem é o else acima?
  - O compilador sempre associa um else ao "if anterior mais próximo que ainda não possui um else."
- Como associar o else à instrução if (nota >= 9)?

A importância dos recuos

- Exemplo:

```
if (nota >= 9)
{
   if (nota_anterior < nota)
      printf("Você está melhorando.");
}
else
   printf("Sem estudo é difícil ser aprovado.");
```

- Neste caso, as chaves, em vez de opcionais, serão obrigatórias, pois apenas os recuos não resolvem.

Operadores relacionais

- Para escrever condições, são utilizados os operadores relacionais e os operadores lógicos.

Operador	Significado
>	Maior do que
<	Menor do que
>=	Maior do que ou igual a
<=	Menor do que ou igual a
==	Igual a
!=	Diferente de

Operadores relacionais

Condição	Valor lógico
(a != x)	Verdadeiro
(a/2.0 == x)	Verdadeiro
(a/2 == x)	Falso
(a/x < 2)	Falso
(a)	Verdadeiro
(a – 2*x)	Falso

```
int a = 3; float x = 1.5;
```

Na linguagem C o valor lógico **falso é dado pelo inteiro 0** e o valor lógico **verdadeiro é dado qualquer valor diferente de 0**

Operadores lógicos

- Os operadores lógicos permitem combinar várias condições em uma única expressão lógica.

Operador	Significado
&&	Conjunção lógica (“and”)
||	Disjunção lógica (“or”)
!	Negação lógica (“not”)

Atribuição e teste de igualdade

- Um erro comum em linguagem C é usar o operador de atribuição (=) em vez do operador relacional (==) em condições que testam igualdade.

```
int fator = 3;
if (fator == 1)
{
   printf("O fator e' unitario\n");
}
printf("fator = %d\n", fator);
```

Imprime:
fator = 3
pois:
(fator == 1) é falso!

```
int fator = 3;
if (fator = 1)
{
   printf("O fator e' unitario\n");
}
printf("fator = %d\n", fator);
```

Imprime:
O fator e' unitario
fator = 1
pois:
(fator = 1) retorna 1, logo é verdadeiro!!

Problema

- Escreva um programa em C que receba um inteiro representando o número de um mês e escreva o número de dias desse mês (considere que fevereiro tem sempre 28 dias).

Solução com if-else

```c
int main()
{
    int mes;
    int dias;

    scanf("%d", &mes);

    if (mes==1||mes==3||mes==5||mes==7||mes==8||mes==10||mes==12)
        dias= 31;
    else if (mes==2)
        dias = 28;
    else
        dias = 30;

    printf("%d - %d", mes,dias);
    return 0;
}
```

Comando switch

```
switch (expressão)
{
    case constante-1:
        comando;
        comando;
    case constante-2:
        comando;
    ...
    default:
        comando;
        comando;
}
```

Comando switch

- Com o comando switch, a função dias_do_mes pode ser reescrita como:

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int mes;
    int dias;
    scanf("%d", &mes);
    switch (mes)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            dias = 31;
            break;
        case 2:
            dias = 28;
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            dias = 30;
            break;
        default:
            dias = 0;
    }
    printf("%d - %d", mes,dias);
    return 0;
}
```

Comando switch

- O **caso** cuja constante for igual ao valor da expressão será selecionado para execução.

- Os comandos associados a este caso **e todos os comandos seguintes** serão executados em sequência até o final do comando switch.

- Para evitar a execução de todos os comandos seguintes, usa-se o comando **break**.

Comando switch

- Como assim?

Para mes = 2:

Sem o uso de break:

```
dias = 28;
dias = 30;
dias = 0;
```

Para mes = 2:

Com o uso de break:

```
dias = 28;
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int mes;
    int dias;
    scanf("%d", &mes);
    switch (mes)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            dias = 31;
            break;
        case 2:
            dias = 28;
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            dias = 30;
            break;
        default:
            dias = 0;
    }
    printf("%d - %d", mes,dias);
    return 0;
}
```

Operador condicional

- O operador condicional na linguagem C tem a seguinte sintaxe:

```
(condição) ? resultado-se-condição-verdadeira : resultado-se-condição-falsa
```

- Os resultados podem ser de qualquer tipo (int, float, char, double) e mesmo strings.

- Exemplos:

```
(b != 0) ? a/b : 0
(peso <= 75) ? "ok" : "deve emagrecer"
```

Operador condicional

- O operador condicional pode ser usado em atribuições.

- Exemplo:

```
float nota1 = 5.0, nota2 = 4.0;

media = ((nota1 >= 3) && (nota2 >= 5)) ?
      (nota1 + 2*nota2)/3 :
      (nota1 + nota2)/2;
```

media recebe o valor 4.5

Qual seria o valor de média se:

```
float nota1 = 5.0;
float nota2 = 6.5;
```

Processamento condicional

- Exemplo

Determine as raízes da equação $ax^2 + bx + c = 0$

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char *argv[])
{
  int a = 2, b = 3, c = 1;
  float delta,x1,x2;

  delta = b*b - 4*a*c;
  printf("A equacao %s\n", (delta >=0) ? "possui raizes reais" :
                                         "nao possui raizes reais");
  if (delta >= 0)
  {
    printf("As raizes sao %s\n", (delta > 0)? "diferentes" : "iguais");
    x1 = (-b + sqrt(delta))/(2*a);
    x2 = (-b - sqrt(delta))/(2*a);
    printf("Raiz x1 = %f\n", x1);
    printf("Raiz x2 = %f\n", x2);
  }


  return 0;
}
```

Referências

- Senne, E.L.F. Primeiro Curso de Programação em C. Editora: Visual Books; edição: 3; ano:2009; número de páginas: 318. Disponível na Biblioteca

- BACKES, André. **Linguagem C:** completa e descomplicada. Rio de Janeiro: Elsevier, 2013. 371 p. ISBN 978-85-352-6855-3.Disponível na Biblioteca