controle de fluxo - condicionais

controle de fluxo - condicionais

tópicos

- if
- if … else
- if … else if … else
- expressões booleanas
- switch

referências

- Capítulo 4 da apostila
- Capítulo 3 do livro

condicionais (tomada de decisão) – if

```
if ( expressão booleana ) /* se expressão for verdadeira */
{
    bloco de comandos 1  /* ... Executa o bloco de comandos 1, */
}

próximo comando ....     /* prossegue para o próximo comando após o if */
```

exemplo

```
if ( nota < 5.0 )          /* se expressão for verdadeira */
{
    printf("Reprovado");  /* ... Executa o bloco de comandos 1, */
    ....
}

printf ("\nfim"); ..       /* prossegue para o próximo comando após o if */
```

condicionais (tomada de decisão) – if ... else

```
if ( expressão booleana ) /* se expressão for verdadeira */
{
    bloco de comandos 1  /
    ....
}
else                     /* senao */
{
    bloco de comandos 2  /
    ....
}

próximo comando ....     /* prossegue para o próximo comando após o if */
```

verdadeiro: expressao != 0

falso: expressao == 0

exemplo

```
if ( media >= 6.0 )              /* se expressão for verdadeira */
{
    printf("Aprovado");          /* ... Executa o bloco de comandos 1, */
}
else                             /* senao */
{
    printf("Em prova final\n");  /* ... Executa o bloco de comandos 2, */
}
printf ("\nfim"); ..           /* prossegue para o próximo comando após o if */
```

exemplo

```
#include <stdio.h>

int main (void)
{
  float media;
  printf("Digite sua media: ");
  scanf ("%f", &media);
  if (media >= 6.0)
  {
      printf ("Aprovado\n");
  }
  else
  {
      printf ("Em prova final");
  }
  return 0;
}
```

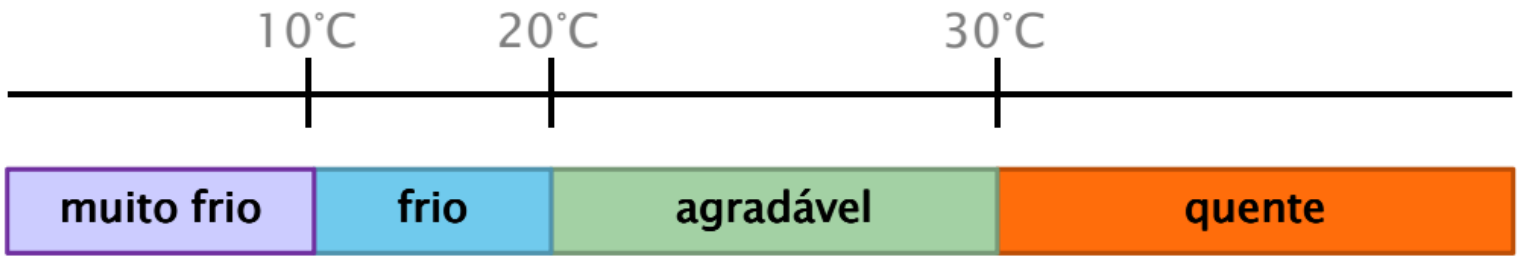
```
#include <stdio.h>

int main (void)
{
  float media;
  printf("Digite sua media: ");
  scanf ("%f", &media);
  if (media >= 6.0)
      printf ("Aprovado\n");
  else
      printf ("Em prova final");
  return 0;
}
```

Quando só tem um comando, não precisa marcar com bloco entre { e }.

tomada de decisão

exemplo (1): qualificando a temperatura



Casos de teste

- 5ºC 15ºC 25ºC 35ºC

cuidado!

```c
/* temperatura (versao 1 - INCORRETA) */
#include <stdio.h>
int main (void)
{
     int temp;
     printf("Digite a temperatura: ");
     scanf("%d", &temp);
     if (temp < 30)
          if (temp > 20)
               printf("Temperatura agradavel \n");
     else
          printf("Temperatura quente \n");
     return 0;
}
```

O que acontece quando a temperatura for:

- 5ºC ?
- 15ºC ?
- 25ºC ?
- 35ºC ?

Em C, um else está associado ao último if "sem else".

cuidado!

```c
/* temperatura (versao 1 - INCORRETA) */
#include <stdio.h>
int main (void)
{
     int temp;
     printf("Digite a temperatura: ");
     scanf("%d", &temp);
     if (temp < 30)
          if (temp > 20)
               printf("Temperatura agradavel \n");
     else
          printf("Temperatura quente \n");
     return 0;
}
```

O que acontece quando a temperatura for:

- 5ºC ?
- 15ºC ?
- 25ºC ?
- 35ºC ?

ainda incompleto...

```c
/* temperatura (versao 2 - INCOMPLETA) */
#include <stdio.h>
int main (void)
{
     int temp;
     printf("Digite a temperatura: ");
     scanf("%d", &temp);
     if (temp < 30)
     {
          if (temp > 20)
               printf("Temperatura agradavel \n");
     }
     else
          printf("Temperatura quente \n");
     return 0;
}
```

O que acontece quando a temperatura for:

- 5ºC ?
- 15ºC ?
- 25ºC ?
- 35ºC ?

condicionais (tomada de decisão) – if

expressão booleana

- é uma expressão que, quando avaliada, resulta no valor **falso** ou **verdadeiro**.
- a linguagem C **não** tem um tipo de dado específico para armazenar valores *booleanos*:
  - Em C, o valor booleano é representado por um valor inteiro:
    - 0 significa falso e qualquer outro valor diferente de zero significa verdadeiro.
    - Em geral, usa-se 1 para representar o valor verdadeiro, e qualquer expressão booleana que resulta em verdadeiro resulta no valor 1.

operadores relacionais

> < <= >= == !=

numa expressão booleana, a comparação de dois operandos resulta em

0, caso a expressão seja **falsa**, ou

1, caso a expressão seja **verdadeira**

Exemplo:

```
int a=10, b=10, c=5;
a > b     resulta em 0 (falso)
b > c     resulta em 1 (verdadeiro)
a == b    resulta em 1 (verdadeiro)
b != c    resulta em 1 (verdadeiro)
```

operadores lógicos

combinam expressões ou valores booleanos

conjunção (and)	(operador binário)
&&	a && b
disjunção (or)	(operador binário)
||	a || b
negação (not)	(operador unário)
!	!a

operadores lógicos - conjunção

&& (and) || (or) ! (not)

true && true resulta em true

true && false resulta em false

false && true resulta em false

false && false resulta em false

exemplos

(1 > 0) && (2 > 0) é avaliada como 1 && 1, e resulta em 1 (verdadeiro)

(1 > 0) && (2 < 0) é avaliada como 1 && 0, e resulta em 0 (falso)

(2 < 0) && (1 > 0) resulta em 0 (false)

(1 < 0) && (2 < 0) resulta em 0 (false)

operadores lógicos – conjunção – exemplo

```
if ( media >= 5.0 && nota1 >= 3.0 && nota2 >= 3.0 && nota3 >= 3.0)
{
    printf("Aprovado\n");
}
```

operadores lógicos – disjunção

&& (and) **|| (or)** ! (not)

true || true resulta em true

true || false resulta em true

false || true resulta em true

false || false resulta em false

exemplos

(1 > 0) || (2 > 0) resulta em 1 (verdadeiro)

(1 > 0) || (2 < 0) resulta em 1 (verdadeiro)

(2 < 0) || (1 > 0) é avaliada como 0 || 1, e resulta em 1 (verdadeiro)

(1 < 0) || (2 < 0) é avaliada como 0 || 0, e resulta em 0 (falso)

operadores lógicos – disjunção – exemplo

```
if ( media < 5.0 || nota1 < 3.0 || nota2 < 3.0 || nota3 < 3.0)
{
    printf("Em prova final\n");
}
```

operadores lógicos – negação

&& (and) || (or) **! (not)**

!true resulta em false

!false resulta em true

exemplos

!(1 > 0) é avaliada como !1, e resulta em 0 (falso)

!(1 < 0) é avaliada como !0, e resulta em 1 (verdadeiro)

operadores lógicos – negação – exemplo

```
if ( !( media < 5.0 || nota1 < 3.0 || nota2 < 3.0 || nota3 < 3.0) )
{
    printf("Aprovado\n");
}
```

condicionais (tomada de decisão) – encadeamento if ... else if ... else

```
if ( expressao1 )          /* se expressão1 for verdadeira */
{
    bloco de comandos 1    /* ... Executa o bloco de comandos 1, */
}
else if ( expressao2 )     /* senao  se expresão2 for verdadeira */
{
    bloco de comandos 2    /* ... Executa o bloco de comandos 2, */
}
else if ( expressao3 )     /* senao  se expresao3 for verdadeira */
{
    bloco de comandos 3    /* ... Executa o bloco de comandos 3, */
}
else                       /* senao  */
{
    bloco de comandos n    /*executado quando todas as expressões são falsas*/
}

próximo comando ....      /* prossegue para o próximo comando após o if */
```

dúvidas?

exemplo (2)

- Implemente e teste um programa para converter o critério de avaliação de alunos em escolas brasileiras para o critério utilizado em escolas americanas. Nas escolas brasileiras, a avaliação dos alunos é reportada por uma nota que varia de 0 a 10. Nas escolas americanas, a avaliação dos alunos é baseada em conceitos: A, B, C, D, ou F. Assuma a seguinte equivalência:
  - **A** (9.0 a 10.0),
  - **B** (8.0 a 8.9),
  - **C** (7.0 a 7.9),
  - **D** (5.0 a 6.9), e
  - **F** (menor que 5.0)

exemplo (2): solução (1)

```c
#include <stdio.h>
int main (void) {
   float nota;
   printf("Entre com a nota: ");
   scanf("%f", &nota);

   if (nota >= 9.0) {
      printf("A");
   }
   if (nota >= 8.0 && nota < 9.0) {
      printf("B");
   }
   if (nota >= 7.0 && nota < 8.0) {
      printf("C");
   }
   if (nota >= 5.0 && nota < 7.0) {
      printf("D");
   }
   if (nota < 5.0) {
      printf("F");
   }
   return 0;
}
```

exemplo (2): solução (2)

```c
/* solução mais estruturada e mais eficiente */
#include <stdio.h>

int main (void) {
   float nota;
   printf("Entre com a nota: ");
   scanf("%f",&nota);
   if (nota >= 9.0) {
      printf("A");
   } else if (nota >= 8.0) {
      printf("B");
   } else if (nota >= 7.0) {
      printf("C");
   } else if (nota >= 5.0) {
      printf("D");
   } else {
      printf("F");
   }
   return 0;
}
```

bloco de comandos

- Na linguagem C, podemos agrupar comandos em blocos, envolvendo-os com abre e fecha chaves ({...}), como fizemos para delimitar o bloco de comando **if** e **else** nas construções para tomada de decisões.

- Na verdade, podemos criar blocos de comandos em qualquer ponto do programa, bastando envolver comandos com chaves.

- Uma variável declarada dentro de um bloco existe enquanto os comandos do bloco estiverem sendo executados. Quando o bloco chega ao fim, as variáveis declaradas dentro dele deixam de existir.

blocos de comandos

- Segundo o padrão C89 da linguagem C, uma variável só pode ser declarada no início de um bloco de comandos (mudou no padrão C99).

- Nas construções do comando **if**, os blocos são importantes para identificar o conjunto de comandos cuja execução está submetida à avaliação da expressão booleana.

- No entanto, se um "bloco de comandos" for constituído por apenas um único comando, as chaves podem ser omitidas.

bloco de comandos – voltando ao exemplo (2)

```c
#include <stdio.h>

int main (void){
  float nota;

  printf("Entre com a nota: ");
  scanf("%f",&nota);

  if (nota >= 9.0)
    printf("A");
  else if (nota >= 8.0)
    printf("B");
  else if (nota >= 7.0)
    printf("C");
  else if (nota >= 5.0)
    printf("D");
  else
    printf("F");
  return 0;
}
```

exercício

- Escreva um programa em C que leia uma média e exiba o status de um aluno:
  - AP se o aluno está aprovado (média final >= 6)
  - RM se o aluno está reprovado (média final < 3)
  - PF se o aluno está em prova fina (caso contrário)

exercício

- Escreva um programa em C que leia uma média e exiba o status de um aluno:
  - AP se o aluno está aprovado (média final >= 6)
  - RM se o aluno está reprovado (média final < 3)
  - PF se o aluno está em prova fina (caso contrário)

```c
#include <stdio.h>
int main (void)
{
   float media;
   printf("Digite a media: ");
   scanf("%f", &media);
   if (media >= 6)
      printf("AP");
   else if (media < 3)
      printf("RM");
   else
      printf("PF");
   return 0;
}
```

o que tem de errado com esse código?

```
if ( media >= 6.0 )
{
    printf (“Aprovado.\n”);
}
else if ( media < 6.0 )
{
    printf(“Em prova final.\n”);
}
printf(“fim.\n”);
```

Isso é conceitualmente ruim. Por quê?

```
if ( media >= 6.0 )
{
    printf (“Aprovado.\n”);
}
else
{
    printf(“Em prova final.\n”);
}
printf(“fim.\n”);
```

exemplo (3): cálculo das raízes em uma equação do 2º grau

Como primeiro exemplo “mais complexo”, vamos discutir a construção de um programa para calcular as raízes de uma equação do segundo grau.

Sabemos que as raízes de uma equação na forma ax2+bx+c=0 são dadas por: $\frac{-b \pm \sqrt{b^2 - 4*a*c}}{2*a}$

Este seria um problema de codificação direta de uma expressão matemática se não fosse pelo fato das raízes poderem não existir. Na verdade, a raiz quadrada só é definida para valores positivos.

exemplo (3): cálculo das raízes em uma equação do 2º grau

- Se, dentro de um programa, tentarmos avaliar uma expressão matemática cujo resultado é indefinido, o resultado do programa certamente não será o desejado.

- Isto inclui ações como:
  - tentar extrair a raiz quadrada de um número negativo,
  - calcular o logaritmo de um número negativo,
  - ou mesmo fazer uma divisão por zero.

- Por este motivo, devemos avaliar estas expressões apenas após certificarmos que os operandos são válidos.

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

int main (void){
   double a, b, c;  /* coeficientes */
   double x1, x2;   /* raízes */
   double delta;
   printf("Entre com os coeficientes (a b c):");
   scanf("%lf", &a);
   scanf("%lf", &b);
   scanf("%lf", &c);
   if (a == 0.0) {
      printf("Valor de 'a' nao pode ser zero.");
      return 1;
   }
   delta = b*b - 4*a*c;
   if (delta < 0) {
      printf("Raizes reais inexistentes.");
   }
   else if (delta == 0.0) {
      x1 = -b / (2*a);
      printf("Uma raiz real: %f", x1);
   }
   else {
     delta = sqrt(delta);
     x1 = (-b + delta) / (2*a);
     x2 = (-b - delta) / (2*a);
     printf("Duas raizes reais: %f e %f", x1, x2);
   }
   return 0;
}
```

dúvidas?

seleção – comando switch

- Seleciona um dentre vários casos
- (opk deve ser um inteiro ou caractere)

```
switch ( expr )
{
    case op1: bloco de comandos 1; break;
    case op2: bloco de comandos 2; break;
    case op3: bloco de comandos 3; break;
...
    default: bloco de comandos default; break;
}
próximo comando ....     /* prossegue para o próximo comando após o if */
```

O switch seleciona apenas o ponto de entrada. O comando break é necessário para prosseguir a partir do comando que sucede o switch, pulando os cases seguintes.

exemplo – calculadora

```c
/* calculadora de quatro operações */
#include <stdio.h>
int main (void)
{
  float num1, num2;
  char op;
  printf("Digite uma expressao: numero operador numero\n");
  scanf ("%f %c %f", &num1, &op, &num2);

  switch (op)
  {
    case '+': printf(" = %f\n", num1 + num2); break;
    case '-': printf(" = %f\n", num1 - num2); break;
    case '*': printf(" = %f\n", num1 * num2); break;
    case '/': printf(" = %f\n", num1 / num2); break;
    default : printf("Operador invalido!\n"); break;
  }
  return 0;
}
```

dúvidas?