## ESTRUTURAS DE CONTROLE DE FLUXO

As estruturas de controle de fluxo são fundamentais para qualquer linguagem de programação. Sem elas só haveria uma maneira do programa ser executado: de cima para baixo comando por comando. Não haveria condições, repetições ou saltos. A linguagem C possui diversos comandos de controle de fluxo. É possível resolver todos os problemas sem utilizar todas elas, mas devemos nos lembrar que a elegância e facilidade de entendimento de um programa dependem do uso correto das estruturas no local certo.

**5 ESTRUTURAS CONDICIONAIS**

### O Comando if

O comando **if** representa uma tomada de decisão do tipo "SE isto ENTÃO aquilo". A sua forma geral é:

*if (condição) <ação>;*

A condição do comando **if** é uma expressão que será avaliada. Se o resultado **for zero** a <*ação>* não será executada. Se o resultado for qualquer coisa **diferente de zero** a <*ação>* será executada. A <*ação>* pode ser um bloco de código ou apenas um comando. Abaixo apresentamos um exemplo:

#include <stdio.h>

#include <stdlib.h>

int main ()

{

int num;

printf ("Digite um numero: ");

scanf ("%d",&num);

if (num>10) printf ("\n\nO numero e maior que 10");

if (num==10)

{

printf ("\n\nVoce acertou!\n");

printf ("O numero e igual a 10.");

}

if (num<10) printf ("\n\nO numero e menor que 10\n");

system("PAUSE");

return 0;

}

No programa acima a expressão **num>10** é avaliada e retorna um valor diferente de zero, se for verdadeira, e retorna zero se for falsa. No exemplo, se num for maior que 10, a seguinte frase será impressa: "O número e maior que 10".  Repare que, se o número for igual a 10, estamos executando dois comandos. Para que isto fosse possível, tivemos que agrupá-los em um bloco que se inicia logo após a comparação e termina após o segundo printf.

**IMPORTANTE:** Note também que quando queremos testar igualdades usamos o operador **==** e não **=**. Isto  porque o operador **=** representa *apenas* uma atribuição. Pode parecer estranho à primeira vista, mas se escrevêssemos:

if (num=10) ...       /\* Atenção: Isto está errado \*/

o compilador iria *atribuir* o valor 10 à variável **num** e a expressão **num=10** iria retornar 10, fazendo com que o nosso valor de **num** fosse modificado e fazendo com que a <*ação>* fosse executada sempre. Este problema gera erros frequentes entre iniciantes e, portanto, muita atenção deve ser tomada.

Os operadores de comparação são:

|  |  |
| --- | --- |
| **símbolo** | **Significado** |
| **==** | Igual |
| **!=** | Diferente |
| > | Maior que |
| < | Menor que |
| >= | Maior ou igual a |
| <= | Menor ou igual a |

#### - O else

Podemos pensar no comando **else** como sendo um complemento do comando [**if**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html#c410.html#c410.html). O comando [**if**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html#c410.html#c410.html) completo tem a seguinte forma geral:

*if (condição) <ação1>;*   
*else <ação2>;*

A expressão da condição será avaliada. Se ela for diferente de zero a ação\_1 será executada. Se for zero a ação\_2 será executada. É importante nunca esquecer que, quando usamos a estrutura **if-else**, estamos garantindo que uma das duas declarações será executada. Nunca serão executadas as duas ou nenhuma delas. Abaixo está um exemplo do uso do **if-else** que deve funcionar como o programa da seção anterior.

#include <stdio.h>

#include <stdlib.h>

int main ()

{

int num;

printf ("Digite um numero: ");

scanf ("%d",&num);

if (num==10)

{

printf ("\n\nVocê acertou!\n");

printf ("O numero e igual a 10.\n");

}

else

{

printf ("\n\nVoce errou!\n");

printf ("O numero e diferente de 10.\n");

}

system("PAUSE");

return 0;

}

#### - O if-else-if

A estrutura **if-else-if** é apenas uma extensão da estrutura [**if-else**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html#c411.html#c411.html). Sua forma geral pode ser escrita como sendo:

*if (condição\_1) <ação\_1>;*   
*else if (condição\_2) <ação\_2>;*   
*else if (condição\_3) <ação\_3>;*   
*.*   
*.*   
*.*   
*else if (condição\_n) <ação\_n>;*   
*else <ação\_default>;*

A estrutura acima funciona da seguinte maneira: o programa começa a testar as condições começando pela 1 e continua a testar até que ele ache uma expressão cujo resultado dê diferente de zero. Neste caso ele executa a ação correspondente. Só uma ação será executada, ou seja, só será executada a ação equivalente à *primeira* condição que der diferente de zero. A última ação (default) é a que será executada no caso de todas as condições darem zero e é opcional.

Um exemplo da estrutura acima pode ser visto no próximo programa:

#### #include <stdio.h>

#### #include <stdlib.h>

#### int main ()

#### {

#### int num;

#### printf ("Digite um numero: ");

#### scanf ("%d",&num);

#### 

#### if (num==10)

#### {

#### printf ("\n\nVoce acertou!\n");

#### printf ("O numero e igual a 10.\n");

#### }

#### else if (num>10) printf ("\n\nO numero e maior que 10\n");

#### else printf ("\n\nO numero e menor que 10\n");

#### 

#### system("PAUSE");

#### return 0;

#### }

#### - A expressão condicional

Quando o compilador avalia uma condição, ele quer um valor de retorno para poder tomar a decisão. Mas esta expressão não necessita ser uma expressão no sentido convencional. Uma variável sozinha pode ser uma "expressão" e esta retorna o seu próprio valor. Isto quer dizer que teremos as seguintes expressões:

                int num;

                if (num!=0) ....

                if (num==0) ....

for (i = 0; string[i] != '\0'; i++)

equivalem a

                int num;

                if (num) ....

                if (!num) ....

for (i = 0; string[i]; i++)

Isto quer dizer que podemos simplificar algumas expressões simples.

#### - ifs aninhados

O [**if**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html#c410.html#c410.html) aninhado é simplesmente um [**if**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html#c410.html#c410.html) dentro da declaração de um outro [**if**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html#c410.html#c410.html) externo. O único cuidado que devemos ter é o de saber exatamente a qual [**if**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html#c410.html#c410.html) um determinado [**else**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html#c410.html#c410.html) está ligado.

Vejamos um exemplo:

### #include <stdio.h>

### #include <stdlib.h>

### int main ()

### {

### int num;

### printf ("Digite um numero: ");

### scanf ("%d",&num);

### if (num==10)

### {

### printf ("\n\nVoce acertou!\n");

### printf ("O numero e igual a 10.\n");

### }

### else

### {

### if (num>10) printf ("O numero e maior que 10.\n");

### else printf ("O numero e menor que 10.\n");

### }

### 

### system("PAUSE");

### return 0;

### }

### - O Operador ?

Uma expressão como:

if (a>0) b=-150;

else       b=150;

pode ser simplificada usando-se o operador **?** da seguinte maneira:

b= a>0 ? -150 : 150;

De uma maneira geral expressões do tipo:

*if (condição) <ação\_1>;*

*else <ação\_2>;*

podem ser substituídas por:

*condição ? expressão\_1 : expressão\_2;*

O operador **?** é limitado (não atende a uma gama muito grande de casos) mas pode ser usado para simplificar expressões complicadas. Uma aplicação interessante é a do contador circular.

Veja o exemplo:

#include <stdio.h>

int main()

{

     int index = 0, contador;

     char letras[5] = "João";

     for (contador=0; contador < 1000; contador++)

{

printf("\n%c",letras[index]);

(index==3) ? index=0: ++index;

}

}

O nome João é escrito na tela verticalmente (mil vezes) até a variável contador determinar o término do programa. Enquanto isto a variável index assume os valores 0, 1, 2, 3, , 0, 1, ... progressivamente.

EXERCÍCIOS

* 1. Escrever um algoritmo que lê um número inteiro e verifica se é par ou impar (usar % e if/else).
  2. Escrever um programa que lê um número inteiro e verifica se é negativo, positivo ou zero (usar if/else if/else)

### O Comando switch

O comando [**if-else**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html#c411.html) e o comando **switch** são os dois comandos de tomada de decisão ou condicionais. Sem dúvida alguma o mais importante dos dois é o [**if**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html), mas o comando **switch** tem aplicações valiosas. Mais uma vez vale lembrar que devemos usar o comando certo no local certo. Isto assegura um código limpo e de fácil entendimento. O comando **switch** é próprio para se testar uma variável em relação a diversos valores pré-estabelecidos. Sua forma geral é:

switch (variável)

*{*

*case constante\_1:*   
*<ação\_1>;*   
*break;*   
*case constante\_2:*   
*<ação\_2>;*   
*break;*   
*...*  
*case constante\_n:*   
*<ação\_n>;*   
*break;*   
*default*   
*<ação\_default>;*   
*}*

Podemos fazer uma analogia entre o **switch** e a estrutura [**if-else-if**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html#c412.html) [apresentada anteriormente](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c410.html#c412.html). A diferença fundamental é que a estrutura **switch** *não* aceita expressões, aceita apenas constantes. O **switch** testa a variável e executa a <ação> cujo **case** corresponda ao valor atual da variável. A <ação **default>** é opcional e será executada apenas se a variável, que está sendo testada, não for igual a nenhuma das constantes.

O comando [**break**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c460.html) faz com que o **switch** seja interrompido assim que uma das ações seja executada. Mas ele não é essencial ao comando **switch**. Se após a execução da <ação> não houver um [**break**](../../Sistemas%20de%20Informação/Linguagem%20C/Programacao%20C/Tutorial%20UFMG/aulas/c460.html), o programa continuará testando. Isto pode ser útil em algumas situações, mas recomenda-se cuidado.

Veremos agora um exemplo do comando **switch**:

#include <stdio.h>

#include <stdlib.h>

int main ()

{

int num;

printf ("Digite um numero: ");

scanf ("%d",&num);

switch (num)

{

case 1:

printf ("\n\nO numero é igual a 1.\n");

break;

case 2:

printf ("\n\nO numero é igual a 2.\n");

break;

case 3:

printf ("\n\nO numero é igual a 3.\n");

break;

default:

printf ("\n\nO numero nao é nem 1, nem 2 e nem 3.\n");

}

system("PAUSE");

return 0;

}

Outro exemplo com mais de uma opção em conjunto:

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

int n;

printf("Informe um valor");

scanf("%d", &n);

switch (n) {

case 1: printf("Um");break;

case 2: case 3: case 4: printf("Dois, três ou quatro");break;

default: printf("Outros valores");

}

return 0;

}

Teste com caracteres:

#include <stdio.h>

int main()

{

char operacao;

float num1, num2;

printf("Escolha sua operação [+ - \* / ]: ");

scanf("%c",&operacao);

printf("Entre com o primeiro número: ");

scanf("%f",&num1);

printf("Entre com o segundo número: ");

scanf("%f",&num2);

switch( operacao ) {

case '+':

printf("%.2f + %.2f = %.2f", num1, num2, num1 + num2);

break;

case '-':

printf("%.2f - %.2f = %.2f", num1, num2, num1 - num2);

break;

case '\*':

printf("%.2f \* %.2f = %.2f", num1, num2, num1 \* num2);

break;

case '/':

printf("%.2f / %.2f = %.2f", num1, num2, num1 / num2);

break;

default: printf("Você digitou uma operação invalida.");

}

}

EXERCÍCIOS

* 1. Construir um programa que solicite ao usuário um número entre 1 e 12 e apresente o nome do mês equivalente.