Comandos de Controle

Comandos de Teste

Blocos de Comandos

Blocos de comando são grupos de comandos que devem ser tratados como uma unidade lógica. O início de um bloco em C é marcado por uma **{** e o término por uma **}**. O bloco de comando serve para criar um grupo de comandos que devem ser executados juntos. Normalmente usa-se bloco de comandos quando se usa comandos de teste em que deve-se escolher entre executar dois blocos de comandos. Um bloco de comandos pode ser utilizado em qualquer trecho de programa que se pode usar um comando C.

Comando if

O comando `if` é utilizado quando for necessário escolher entre dois caminhos, ou quando se deseja executar um comando sujeito ao resultado de um teste. A forma geral do comando `if` é a seguinte:

```
if (expressão)
  comando1;
else
  comando2;
```

Neste comando a expressãp é avaliada e caso o resultado seja `VERDADE` (qualquer resultado diferente de zero) o `comando1` é executado, caso contrário o comando2 é que é executado. Pela definição do comando a expressão deve ter como resultado um valor diferente de zero para ser considerada verdade. Observar que somente um dos dois comandos será executado.

Como a cláusula `else` é opcional a forma abaixo do comando `if` é perfeitamente válida.

```
if (expressão)
  comando;
```

Nos dois casos acima os comandos sempre podem ser substituídos por um bloco de comandos, e neste caso os comandos

ficariam da seguinte maneira:

```
if (expressão)
{
        /* lista de comandos a serem executados */
}
else
{
        /* lista alternativa de comandos a serem executados */
}
```

ou

```
if (expressão)
{
        /* lista de comandos a serem executados */
}
```

A seguir mostramos alguns trechos de programas com exemplos de uso de comando `if`:

1. ```
   scanf("%d", &dia);
   if ( dia > 31 && dia < 1 )
       printf("Dia invalido\n");
   ```

2. ```
   scanf("%d", &numero);
   if ( numero >= 0 )
     printf("Numero positivo\n");
   else
     printf("Numero negativo\n");
   ```

3. ```
   scanf("%f", salario);
   if (salario < 800.00)
   {
     printf("Aliquota de imposto = 0.1\n");
     imposto = salario * 0.1;
   }
   else
   {
     printf("Aliquota de imposto = 0.25\n");
     imposto = salario * 0.25;
   }
   ```

Uma construção que pode aparecer são os comandos `if's` em escada, cuja forma geral é a seguinte:

```
if (expressão)
   comando;
else if (expressão)
        comando;
     else if (expressão)
             comando;
                ...
          else comando;
```

O programa calc.c mostra um exemplo com `if's` em escada e aninhados.

Para evitar que o recuo da escada seja muito profundo o comando if em escada pode ser escrito da seguinte maneira:

```
if (expressão)
  comando;
else if (expressão)
  comando;
else if (expressão)
  comando;
  ...
else comando;
```
```

Comando switch

O comando `if`, em todas suas formas, é suficiente resolver problemas de seleçao de comandos. Porém em alguns casos, como no exemplo acima (ccalc.c) o programa se torna mais trabalhoso para ser escrito. O comando `switch` facilita a escrita de trechos de programa em que a seleção deve ser feita entre várias alternativas. A forma geral do comando `switch` é a seguinte:

```
switch (expressão) {
case constante1:
  sequência de comandos;
  break;
case constante2:
     sequência de comandos;
     break;
case constante3:
     sequência de comandos;
     break;
     ...
default:
    sequência de comandos;
}
```

A execução do comando segue os seguintes passos:

1. A expressão é avaliada;
2. O resultado da expressão é comparado com os valores das constantes que aparecem nos comandos `case`;
3. Quando o resultado da expressão for igual a uma das constantes, a execução se inicia a partir do comando associado com esta constante. A execução continua com a execução de todos os comandos até o fim do comando `switch`, ou até que um comando `break` seja encontrado;
4. Caso não ocorra nenhuma coincidência o comando default é executado. O comando default é opcional e se ele não aparecer nenhum comando será executado.

O comando `break` é um dos comandos de desvio da linguagem C. O `break` se usa dentro do comando `switch` para interromper a execução e pular para o comando seguinte ao comando `switch`.

Há alguns pontos importantes que devem ser mencionados sobre o comando `switch`.

- O resultado da expressão deve ser um tipo compatível com um inteiro, isto é, expressões com resultados tipo `char` também podem ser usadas;
- Notar que caso não apareca um comando de desvio todas as instruções seguintes ao teste `case` que teve sucesso serão executadas, mesmo as que estejam relacionadas com outros testes `case`;
- O comando switch só pode testar igualdade;
- Não podem aparecer duas constantes iguais em um `case`;

O programa calsw.c mostra um exemplo de uso de comandos `switch`.

Comando Ternário

O comando ternário tem este nome porque necessita de três operandos para ser avaliado. O comando ternário tem a seguinte forma:

```
expressão1 ? expressão2 : expressão3
```

Para avaliar o resultado da expressão primeiro expressão1 é avaliada. Caso este resultado seja correspondente ao valor VERDADE então resultado da expressão será igual ao valor da expressão2, no caso contrário expressão3 é avaliada e se torna o resultado.

O programa tern.c mostra um exemplo de uso de comando ternário.

Laços

Estes comandos permitem que trechos de programa sejam repetidos um certo número de vezes controlado pelo programa. O número de vezes que um laço será executado pode ser fixo ou depender de condições.

Comando for

Este comando aparece em várias linguagens de programação, mas na linguagem C ele apresenta uma grau maior de flexibilidade. A idéia básica do comando for é a seguinte. Uma variável de controle, geralmente um contador, recebe um valor inicial. O trecho de programa que pertence ao laço é executado e ao final a variaável de controle é incrementada ou decrementada e comparada com o valor final que ela deve alcançar. Caso a condição de término tenha sido atingida o laço é interrompido.

A forma geral do comando `for` é a seguinte:

```
for (expressão1; expressão2; expressão3) comando;
```

As três expressões geralmente tem os seguintes significados:

1. A expressão1 é utilizada para inicializar a variável de controle do laço;
2. A expressão2 é um teste que controla o fim do laço;
3. A expressão3 normalmente faz um incremento ou decremento da variável de controle.

A execução do comando `for` segue os seguintes passos:

1. A expressão1 é avaliada;
2. A expressão2 é avaliada para determinar se o comando deve ser executado;
3. Se o resultado da expressão2 for VERDADE o comando é executado caso contrário o laço é terminado;
4. A expressão3 é avaliada;
5. Voltar para o passo onde 2.

Exemplo Imp100f: O trecho de programa abaixo imprime todos os números entre 1 e 100.

```
for (i = 0; i <= 100; i++)
     printf("Numero %d\n", i);
```

É possível omitir qualquer uma das expressões. Por exemplo, se a expressão2 for omitida o programa assume que ela é sempre verdade de modo que o laço só termina com um comando de desvio como o `break`

```
for (i = 0; ; i++){
    printf("numero %d\n", i);
    if (i == 5) break;
}
```

O programa fat.c mostra como se pode calcular o fatorial de um número usando-se o comando `for`.

Laços `for` com mais de um comando por expressão
Outra possibilidade que o comando `for` em C permite e a inclusão de mais de um comando, separados por vírgulas, nas expressões. O programa for1.c mostra um exemplo de uso de comando `for` com vários comandos nas expressões.

Laços `for` com testes com outras variáveis
A expressão de controle não precisa necessáriamente envolver somente um teste com a variável que controla o laço. O teste de final do laço pode ser qualquer expressão relacional ou lógica. No exemplo linf.c o laço pode terminar porque a variável de controle já chegou ao seu valor limite ou foi batida a tecla '*'.

Laços `for` com expressões faltando
Um outro ponto importante do for é que nem todas as expressões precisam estar presentes. No exemplo, fors.c a variável de controle não é incrementada. A única maneira do programa terminar é o usuário bater o número -1.

Laço infinito
Uma construção muito utilizada é o laço infinito. No laço infinito o programa para quando se usa o comando break. No exemplo finf.c o programa só para for digitada ou a tecla 's' ou 'S '.

Laços aninhados
Uma importante construção aparece quando colocamos como comando a ser repetido um outro comando `for`. Esta construção aparece quando estamos trabalhando com matrizes. O exemplo ftab.c mostra um programa que imprime uma tabuada.

Comando while

O comando while tem a seguinte forma geral:

```
while (expressão)
    comando;
```

A expressão pode assumir o valor FALSO (igual a 0) ou VERDADE (diferente de 0). Os passos para execução do comando são os seguintes:

1. A expressão é avaliada;
2. Se a expressão for VERDADE então o comando é executado, caso contrário a execução do comando é terminada;
3. Voltar para o passo 1.

Uma característica do comando while, como pode ser visto dos passos acima, é que o comando pode não ser executado.

O exemplo de comando `for` Imp100f escrito com comando `while` fica da seguinte maneira:

```
i = 1;
while (i <= 100){
    printf("Numero %d\n", i);
    i++;
}
```

O trecho acima escrito de forma mais compacta fica:

```
i = 1;
while (i++ <= 100)
    printf("Numero %d\n", i);
```

A expressão do comando pode incluir chamada de função. Lembrar que qualquer atribuição entre parênteses é considerada como uma expressão que tem o valor da atribuição sendo feita. Por exemplo, o trecho abaixo repete um bloco de comandos enquanto o usuário usar a tecla 'c' para continuar, qualquer outra tecla o bloco é interrompido.

```
#include <stdio.h>
#include <conio.h>
void main()
```

```
{
    int c;
    puts("Tecle c para continuar.\n");
    while ((c=getche()) == 'c'){
        puts("\nNão Acabou.\n");
    }
    puts("\nAcabou.\n");
}
```

Comando do-while

A forma genérica do comando é a seguinte:

```
do {
    comando;
} while (expressão);
```

Observar que neste comando a expressão de teste está após a execução do comando, portanto o comando é executado pelo menos uma vez. A execução do comando segue os seguintes passos:

1. Executa o comando;
2. Avalia a expressão;
3. Se a expressão for VERDADE então volta para o passo 1, caso contrário interrompe o `do-while`

O exemplo de comando `for` Imp100f escrito com comando `do-while` fica da seguinte maneira:

```
i = 1;
do   {
  printf("Numero %d\n", i);
  i++;
} while (i <= 100);
```

Comandos de Desvio

Comando break

O comando `break` pode ser tanto usado para terminar um teste `case` dentro de um comando `switch` quanto interromper a execução de um laço.

Quando o comando é utilizado dentro de um comando `for` o laço é imediatamente interrompido e o programa continua a execução no comando seguinte ao comando `for`.

No trecho de programa abaixo o comando for deve ler 100 números inteiros positivos. No entanto, se for digitado um número negativo o comando `for` é interrompido imediatamente sem que o número seja impresso.

```
for (i = 0; i <100; i++)
{
   scanf("%d", &um);
   if (num < 0) break;
```

```
   printf("%d\n", num);
}
```

Comando continue

O comando `continue` é parecido com o comando `break`. A diferença é que o comando `continue` interrompe a execução da iteração corrente passando para a próxima iteração do laço, se houver uma. No comando `for` o controle passa para o teste e o incremento do laço sejam executados, nos comandos `while` e `do-while` o controle passa para a fase de testes.

No trecho de programa abaixo o laço lê 100 números inteiros, caso o número seja negativo ele um novo número é lido.

```
for (i = 0; i < 100; i++)
{
  scanf("%d", &num);
  if (num < 0) continue;
  printf("%d\n", num);
}
```

Comando goto

O comando `goto` causa um desvio incondicional para um outro ponto da função em que o comando está sendo usado. O comando para onde deve ser feito o desvio é indicado por um rótulo, que é um identificador válido em C seguido por dois pontos.

É importante notar que o comando `goto` e o ponto para onde será feito o desvio devem estar dentro da mesma função. A forma geral deste comando é:

```
goto rótulo;

.

.

.

rótulo:
```

Este comando durante muito tempo foi associado a programas ilegíveis. O argumento para esta afirmação se baseia no fato de que programas com comandos `goto` perdem a organização e estrutura porque o fluxo de execução pode ficar saltando erraticamente de um ponto para outro. Atualmente as restrições ao uso do comando tem diminuído e seu uso pode ser admitido em alguns casos.

Função exit()

A função exit provoca a terminação de um programa, retornando o controle ao sistema operacional. O protótipo da função é a seguinte:

```
void exit( int codigo);
```

Observar que esta função interrompe o programa como um todo. O código é usado para indicar qual condição causou a interrupção do programa.

Comando return

O comando `return` é usado para interromper a execução de uma função e retornar um valor ao programa que chamou esta função. Caso haja algum valor associado ao comando `return` este é devolvido para a função, caso contrário um valor qualquer é retornado.

A forma geral do comando é:

```
return expressão;
```

Notar que a expressão é opcional. A chave que termina uma função é equivalente a um comando `return` sem a expressão correspondente. É possível haver mais de um comando `return` dentro de uma função. O primeiro que for encontrado durante a execução causará o fim da execução. Uma função declarada como do tipo `void` não pode ter um comando `return` que retorne um valor. Isto não faz sentido, já que funções deste tipo não podem retornar valores.

Exercícios

1. Escreva um programa que calcule x elevado a n. Assuma que n é um valor inteiro.
   Solução: potencia.c

2. Escreva um programa que exiba um menu com as opções "1-multiplicar" e "2-somar", leia a opção desejada, leia dois valores, execute a operação (utilizando o comando "if") e exiba o resultado.
   Solução: menu.c

3. Utilizando if's em escada, inclua, no programa do exercício anterior, as opções "3-Subtrair" e "4-Dividir".
   Solução: if.c

4. Simplifique os programas anteriores da seguinte forma:

   Reescreva o programa do exercício 1 substituindo o comando "if" pelo comando ternário.
   Solução: ternario.c

   Reescreva o programa do exercício 2 substituindo os if's em escada pelo comando "switch".
   Solução: switch.c

5. Utilizando um laço "while" e o comando "break", escreva um programa que exiba a mensagem "HA-HA-HA!! Você está preso." até que a senha "FUI!!" seja digitada.
   Solução: senha.c

6. Utilizando um laço "for" dentro de outro, escreva um programa que exiba as tabuadas de multiplicação dos números de 1 à 9.
   Solução: multi.c

7. Escreva um programa com menu de 5 opções que utilize o comando de desvio "goto" para executar a opção desejada e só saia do programa caso a opção "5-Sair" seja selecionada.
   Solução: goto.c

8. Escreva um programa que tenha um número (inteiro) como entrada do usuário e escreva como saída a sequencia de bits que forma esse numero. Por exemplo, após digitado o número 10, a saida deve ser 0000000000001010.
   Solução: converte.c Solução: converte1.c

9. Escreva um programa que imprima todos os numeros pares entre 0 e 50 e em seguida imprima todos os impares. Deixar um espaco entre os numeros.
   Solução: par.c

10. Escreva um programa que leia 10 numeros. O programa deve imprimir a media, o maior e o menor deles. Obs: Os numeros devem ser entre 0 e 10.
    Solução: media.c

11. Escreva um programa que leia 10 numeros. O programa deve imprimir a media, o maior e o menor deles. Obs: Considere agora que os números podem ser quaisquer.
    Solução: .c

12. Escreva um programa que exibe a tabela ascii.

Solução: ascii.c

13. Crie um programa para verificar se um número dado é primo.
    Solução: primo.c

14. Escreva um prgrama que leia um numero inteiro do teclado e acha se o numero e primo ou nao. O programa deve informar o menor divisor.
    Solução: primos.c

15. Escreva um programa que leia um numero do teclado e ache todos os seus divisores.
    Solução: divisores.c

16. Escreva um programa que imprima a seqüência "9876543218765432176543216543215432143213212 11", mas sem imprimir nenhuma constante, use apenas variáveis. Em outra linha imprima as letras maiúsculas de A até Z (ABCD...).
    Solução: imprime.c

17. Escreva um programa que conte de 100 a 999 (inclusive) e exiba, um por linha, o produto dois tres digitos dos numeros. Por exemplo, inicialmente o programa irá exibir:
    0 (1*0*0)
    0 (1*0*1)
    0 (1*0*2)
    (...)
    0 (1*1*0)
    1 (1*1*1)
    2 (1*1*2)
    até 9*9*9=729

    Faça seu programa dar uma pausa a cada 20 linhas para que seja possível ver todos os números pouco a pouco (solicitando que seja pressionada alguma tecla para ver a próxima sequencia de números, como o DIR/P do DOS).
    Solução: produto.c


Índice