5. Estruturas condicionais

DIM0321

2015.1

Outline

1. Comandos e blocos
2. If-then[-else]
3. Condicional alternativo `switch...case`

Transformar uma expressão

- O ponto-e-vírgula ; transforma uma expressão em **comando**
- Ele é o marcador de **sequência** de C
- É um **terminador** de comando (e não um separador)

Exemplo

```c
x = 0; // valor de x = 0 e 0
x++; // valor retornado e 0
++x; // valor retornado e 2
printf("foo"); //retorna o numero de caracteres imprimido
```

Blocos

Chaves

- { começa um bloco
- } termina um bloco
- As chaves agrupas declarações e comandos a fim de formar 1 **comando composto**
- Um bloco é **sintaticamente** a um comando
- Não tem ; no fim de um bloco
- Variáveis podem ser declaradas no início de qualquer bloco

Observação

- Chaves têm 2 usos

```
int main(int argc, char* argv[])
{ // obrigatório
     if (*argv[0] == 'a') { // opcional
  return 1;
     }
     return 0;
}
```


Exercício

Atenção !

Tem um erro no código abaixo. Qual é ?

```
#include <stdio.h>
int main(void)
{
    int x;
    {
int y = 0;
x = 1;
y = 2;
    }
    printf("%d, %d", x, y);
}
```

Resultado

```
block_error.c: In function ‘main’:
block_error.c:11:25: error: ‘y’ undeclared (first use in this function)
     printf("%d, %d", x, y);
 ^
block_error.c:11:25: note: each undeclared identifier is reported only once
for each function it appears in
```


Exemplo introdutivo

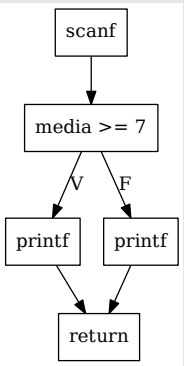
Exemplo

```
#include <stdio.h>

int main(void)
{
     unsigned int media;
     scanf("%u", &media);

     if (media >= 7) printf("Aprovado!\n");
     else printf("Não sei\n");

     return 0;
}
```



Observações

- A sequência a ser executada **depende** de uma condição.
- A condicional é uma estrutura de controle básica.

Condicional (composto)

Sintaxe

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

- A parte do **else** é opcional
- A condição fica entre `(...)`

Semântica

- se `expressao` for verdadeiro (i.e. $\neq 0$)
  - execute `comando1`
- senão:
  - se houver uma parte `else`, execute `comando2`
  - senão continue a execução do programa

Condicional simples

Descrição

- A forma de um condicional simples é

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

Exemplo

```
#include <stdio.h>

int main(void) {
  int media;

  printf("Qual é a média? ");
  scanf("%i", &media);
  if (media >= 7) printf("Aprovado\n");
  else printf("Não aprovado\n");
  return(0);
}
```


Aninhamento de condicionais

Observações

- Uma condicional é um comando **como os outros**
- Pode ocorrer dentro de uma outra condicional

Exemplo

- Alunos com uma média maior que 5 tem direito a mais um exame de recuperação. Isto pode ser programado com mais um comando

```
#include <stdio.h>

int main(void) {
  int media;

  printf("Qual é a média? ");
  scanf("%i", &media);
  if (media >= 7) printf("Aprovado\n");
  else
    if (media >= 5) printf("Recuperação\n");
    else printf("Não aprovado\n");
  return(0);
}
```


Exercício

Exemplo

Qual é o valor de `z` apos a execução do seguinte trecho de código?

```
n = 1;
a = 2;
b = 3;
if (n > 0) if (a > b) z = a; else z = b;
```

Observações

- A parte do `else` é associada ao `if` mais próximo (o último aberto).
- Indentação ajudaria!
- Use **blocos** para desambiguar condicionais aninhadas.

`if-else if`

Exercício

Assunto

Escrever um programa que:

- lê dois valores inteiros a e b e
- imprime a saída "Menor:" + o menor dos números e "Maior" + o maior dos números.
- **Operadores relacionais** !==, !=, >, >=, <, <=

Entrada	Saída
3 4	Menor: 3 Maior: 4
1 1	Menor: 1 Maior: 1

Operadores lógicos

Características

Em resumo

Símbolo	Significado
!	não
&&	e
||	ou

Lembrete

- Os operadores podem ter operandos inteiros
  - 0 ≡ `false`
  - !0 ≡ `true`
- `stdbool.h` apenas define nomes
  - `false` para `(char) 0`
  - `true` para `(char) 1`

```
1 int x = 5;
2 if (x) printf("foo");
```

- É melhor **explicitar** o sentido

```
1 int x = 5;
2 if (x != 0) printf("foo");
```


Conjunção &&

Tabela de verdade

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

Observação

Para qualquer proposição P

- $V \wedge P = P$
- $F \wedge P = P \wedge F = F$

Disjunção ||

Tabela de verdade

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

Observação

Para qualquer proposição P

- $F \vee P = P$
- $V \vee P = P \vee V = V$

Negação !

Tabela de verdade

p	$\neg p$
V	F
F	V

Notações

- ¬ p
- $\bar{p}$

Avaliação

Preguiça

- Fato: $p \vee q = \top \iff p = \top \vee q = \top$ ou $p \wedge q = \bot \iff p = \bot \vee q = \bot$.
- A linguagem C usa esse fato na avaliação das expressões booleanas.
  - Para avaliar a `&&` b, C avalia
    1. a
    2. se for verdadeiro, C avalia (e retorna) b
    3. senão retorna falso

Exercício

Avaliar

1. `!2 > 3`
2. `2 < 3 && !4 > 5 || !3 = -1 + 4=`
3. `0 > !2 * 0 || 3 > 6 * !7=`

Precedências

Operadores
++, --, **!**, +, - (unários)
*, /, %
+, - (binários)
<, >, <=, >=
==, **!=**
&&
||

Descrição

Uso

- O `switch...case` é um outro comando condicional
- Ele permite facilitar a leitura de programa quando tem muitos casos

Sintaxe

```
switch (c) {
case v1:
     do_something1(); // break ?;
case v2:
     do_something2();
default:
     // do something or assert false
}
```


Semântica

- Avaliar c
- Se c for igual a `v1`, execute a partir de `do_something1()`.
  - Para terminar a execução, use o comando `break;`
- Se c for igual a `v2`, execute a partir de `do_something2()`.
  - Para terminar a execução, use o comando `break;`
- Senão, execute o bloco `default`

break

O comando break permite:

- sair diretamente do bloco do `switch`
- levar a execução do programa para o primeiro comando fora do bloco

Exemplo (semântica)

```
#include <stdio.h>

int main(void)
{
     int n;
     scanf("%d", &n);
     switch (n) {
     case 1:
  printf("1\n");
     case 2:
  printf("2\n");
  break;
     case 3:
  printf("3\n");
     default:
  printf("%d\n", n);
     }
     return 0;
}
```

Exemplo

```c
#include <stdio.h>
int main(void)
{
     unsigned int digits_less5, puncts, others;
     int c;
     digits_less5 = 0;
     puncts = 0;
     others = 0;
     while ((c = getchar()) != EOF) {
  switch (c) {
  case '0': case '1': case '2':
  case '3': case '4': case '5':
       digits_less5++;
       break;
  case '!': case '?': case ':':
  case ';': case '.': case ',':
       puncts++;
       break;
  default:
       others++;
       break;
  }
     }
}
```


Exemplo (versão if-then)

```c
#include <stdio.h>
int main(void)
{
     unsigned int digits_less5, puncts, others;
     int c;
     digits_less5 = 0;
     puncts = 0;
     others = 0;
     while ((c = getchar()) != EOF) {
  /* The line below works with if the encoding
     encodes digits are in increasing order starting from 0.
     This works in ASCII.
     Would work in EBCDIC or UTF-8 too.
  */
  if (c >= '0' && c <= '5') digits_less5++;
  else if (c == '!' || c == '?' || c == ':' ||
   c == ';' || c == '.' || c == ',') puncts++;
  else others++;
     }
}
```


Exemplo2

- Se omitir `break`, o fluxo da execução continua.

```c
#include <assert.h>
#include <stdio.h>
#include <stdbool.h>
int main(void)
{
     unsigned int digits_less5, digits;
     int c;
     digits_less5 = 0;
     digits = 0;

     while ((c = getchar()) != EOF) {
  switch (c) {
  case '0': case '1': case '2':
  case '3': case '4': case '5':
       digits_less5++;
  case '6': case '7': case '8':
  case '9':
       digits++;
       break;
  default:
       assert(false);
       break;
  }
     }
}
```


Referências

Precisões

[KR88] Seções 3.1 – 3.4

[Bac13] Cap. 4

André Backes, *Linguagem C completa e descomplicada*, Elsevier, 2013.

Brian W. Kernighan and Dennis M. Ritchie, *The (ANSI) C Programming Language*, 2nd ed., Prentice Hall Professional Technical Reference, 1988.

Perguntas ?

http://dimap.ufrn.br/~richard/dim0321