Fluxo de execução

Condicionais

if

Executa um bloco de comandos se a condição vale `true`. Exemplo:

```
/* módulo do número */
if(x < 0) {
  x = -x;
}
```

Nota: caso haja apenas um comando a ser executado pelo `if`, os colchetes podem ser omitidos, logo o exemplo acima poderia ser escrito como

```
/* módulo do número */
if(x < 0)
  x = -x;
```

if - else

Caso a condição valha `true`, executa o bloco de comandos após o `if`; caso contrário, executa o bloco de comandos após o `else`. Exemplo:

```
if(x % 2 == 0) {
  printf("x eh par\n");
} else {
  printf("x eh impar\n");
}
```

if - else if - ... - else

Podemos encadear várias construções `if..else` de modo a obter:

```
if(condicao1) {
  comandos_se_condicao1_eh_verdadeira;
```

```
} else if(condicao2) {
  comandos_se_condicao1_eh_falsa_mas_condicao2_eh_verdadeira;
} else if(condicao3) {
  comandos_se_condicao1_e_2_sao_falsas_mas_condicao3_eh_verdadeira;
} /* mais else if se voce quiser */ else {
  comandos_se_nenhuma_das_condicoes_eh_verdadeira;
}
```

Exemplo (Fizzbuzz):

```
if(x % 15 == 0) {
  printf("FizzBuzz\n");
} else if(x % 5 == 0) {
  printf("Buzz\n");
} else if(x % 3 == 0) {
  printf("Fizz\n");
} else {
  printf("%d\n", x);
}
```

Nota: esse exemplo pode ser reescrito com um `if` a menos se usarmos uma váriavel extra:

```
bool fizz_or_buzz = false;
if(x % 3 == 0) {
  printf("Fizz");
  fizz_or_buzz = true;
}
if(x % 5 == 0) {
  printf("Buzz");
  fizz_or_buzz = true;
}
if(!fizz_or_buzz) {
  printf("%d", x);
}
printf("\n");
```

switch...case

A construção

```
switch(expr) {
  case VALOR1:
    comandos_se_expr_igual_a_VALOR1;
    break;
  case VALOR2:
    comandos_se_expr_igual_a_VALOR2;
    break;
  case VALOR3:
    comandos_se_expr_igual_a_VALOR3;
    break;
  ...
  default:
    comandos_se_variavel_diferente_de_todos_aqueles_valores;
}
```

É equivalente a

```
var = expr; /* expr é avaliada só uma vez */
if(var == VALOR1) {
    comandos_se_variavel_igual_a_VALOR1;
} else if(var == VALOR2) {
    comandos_se_variavel_igual_a_VALOR2;
} else if(var == VALOR3) {
    comandos_se_variavel_igual_a_VALOR3;
} ... else {
    comandos_se_variavel_diferente_de_todos_aqueles_valores;
}
```

Exemplo:

```
switch(x % 3) {
  case 0:
    printf("x eh divisivel por 3\n");
    break;
  case 1:
    printf("o resto de divisao de x por 3 eh 1\n");
    break;
  default:
```

```
    printf("o resto de divisao de x por 3 eh 2\n");
    break;
}
```

Operadores `?` e `:`

O código

```
sign = x >= 0 ? 0 : 1;
```

É equivalente a

```
if( x >= 0) {
  sign = 0;
} else {
  sign = 1;
}
```

Laços (loops)

while

```
while( condicao) {
  comando_executado_enquanto_condicao_eh_verdadeira;
}
```

A condição é testada, e se for verdadeira, os comandos dentro do `while` são executados. A cada vez que se chega ao final do `while`, a condição é testada novamente, e se ainda for verdadeira, os comandos dentro do `while` são executados de novo, e assim por diante.
Exemplo:

```
/* calcula x! */
int y = 1;
while( x > 0) {
  y *= x;
  x--;
}
```

Podemos escrever um loop infinito como

```
```

```
while(1) {
  printf("loop infinito! \n");
}
```

for

O código

```
for(inicializacao; condicao; transicao) {
  comandos;
}
```

É, por definição, equivalente a

```
inicializacao;
while(condicao) {
  comandos;

  transicao;
}
```

Por exemplo:

```
/* imprime os números de 0 a 9 */
int i;
for(i = 0; i < 10; i++) {
  printf("%d\n", i);
}
```

Um ou mais dos campos do `for` podem estar vazios. Por exemplo, podemos reescrever o exemplo do fatorial acima como

```
/* calcula x! */
int y = 1;
for(; x > 0; x--) {
  y *= x;
}
```

Uma condição vazia é sempre avaliada como verdadeira, de modo que podemos também fazer loops infinitos como

```
```

```
for(;;) {
  printf("loop infinito! \n");
}
```

do...while

Semelhante ao `while`, mas a condição só é testada depois de executar o que está dentro do `do...while` uma vez. Por exemplo,

```
/*
 * pede confirmação do usuário; queremos um loop que executa até o usuário
 * dar uma resposta válida.
 */
int c;
do {
  printf("deseja continuar? (s)im/(n)ao\n");
  c = getchar();
} while(c != 's' && c != 'n');
```

Temos que usar o `do...while` nesse caso porque `c` não tem um valor definido antes da primeira interação do loop.

Revision #1

Created Mon, Jan 28, 2019 11:36 PM by Luana

Updated Mon, Jan 28, 2019 11:37 PM by Luana