CAPÍTULO 3

CONTROLE DE FLUXO

3.1 if

3.1.1 Sintaxe

```
if (expr) comando
```

Se *expr* for verdadeira (diferente de 0), *comando* é executado.

```
if (expr) comando1 else comando2
```

Se *expr* for verdadeira, *comando1* é executado; caso contrário, *comando2* é executado.

```
if (expr1) comando1 else if (expr2) comando2 else comando3
```

Se *expr1* for verdadeira, *comando1* é executado. Se for falsa, se *expr2* for verdadeira, *comando2* é executado; caso contrário, *comando3*

3.1.2 Exemplos

```
/* Exemplo 1 */
if (a == 3)
      b = 4;


/* Exemplo 2 */
if (a > b)
      c = a * a;
else
      c = b * b;


/* Exemplo 3 */
if (a == b)
      c = 0;
else if (a > b)
      c = a * a;
else
      c = b * b;
```

Atenção:

Exemplo	**Ação**
if (a == 3) b = 4;	Testa se o valor de *a* é 3. Se for, atribui o valor 4 a *b*
if (a = 3) b = 4;	Atribui 3 à variável *a*. Testa o valor 3 (verdadeiro); portanto, independente do valor inicial de *a* será atribuído 4 a *b*

3.2 while

3.2.1 Sintaxe

```
while (expr) comando
```

Enquanto *expr* for verdadeira (diferente de 0), *comando* é executado.

Quando o programa chega na linha que contém o teste do comando **while**, ele verifica se a expressão de teste é verdadeira. Se for, o programa executa o comando uma vez e torna a testar a expressão, até que a expressão seja falsa. Somente quando isso ocorrer, o controle do programa passa para a linha seguinte ao laço.

Se, na primeira vez que o programa testar a expressão, ela for falsa, o controle do programa passa para a linha seguinte ao laço, sem executar o comando nenhuma vez.

O corpo de um laço **while** pode ter um único comando terminado por ponto-e-vírgula, vários comandos entre chaves ou ainda nenhuma instrução, mantendo o ponto-e-vírgula.

3.2.2 Exemplo

```
int i;

i = 0;
while (i < 6)
{
    printf("%d\n", i);
    i++;
}
```

3.3 do-while

3.3.1 Sintaxe

```
do comando while (expr)
```

comando é executado enquanto *expr* for verdadeira (diferente de 0).

O laço **do-while** é bastante parecido com o laço **while**. A diferença é que no laço **do-while** o teste da condição é executado somente depois do laço ser processado. Isso garante que o laço será executado pelo menos uma vez.

3.3.2 Exemplo

```
int i;

i = 0;
do
{
    printf("%d\n", i);
    i++;
} while (i < 6)
```

3.4 for

3.4.1 Sintaxe

```
for (inicializacao, condicao, incremento) comando
```

O laço **for** é equivalente ao seguinte laço **while**:

```
inicializacao
while (condicao)
{
    comando
    incremento
}
```

3.4.2 Exemplos

```
int i;

for (i = 0; i < 6; i++)
{
    printf("%d\n", i);
}
```

Qualquer uma das expressões do laço **for** pode conter várias instruções separadas por vírgulas.

```
int i, j;

for (i = 0, j = 0; i + j < 100; i++, j+=2)
{
    printf("%d\n", i + j);
}
```

Qualquer uma das três partes de um laço **for** pode ser omitida, embora os ponto-e-vírgulas devam permanecer. Se a expressão de teste for omitida, é considerada verdadeira.

```
int i = 0;

for (; i < 100;)
{
    printf("%d\n", i++);
}
```

3.5 break

O comando **break** pode ser utilizado no corpo de qualquer estrutura de laço C (**while**, **do-while** e **for**). Causa a imediata saída do laço e o controle passa para o próximo estágio do programa.

3.5.1 Exemplo

```
int i = 0;

while(1)
{
    printf("%d\n", i++);
    if (i >= 6) break;
}
```

3.6 switch

3.6.1 Sintaxe

```
switch(expr)
{
    case constante1:
        comando1;       /*opcional*/
    case constante2:
        comando2;       /*opcional*/
    case constante3:
        comando3;       /*opcional*/
    default:            /*opcional*/
        comando4;       /*opcional*/
}
```

O comando **switch** verifica o valor de *expr* e compara seu valor com os rótulos dos casos. *expr* deve ser inteiro ou caractere.

Cada caso deve ser rotulado por uma constante do tipo inteiro ou caractere. Esta constante deve ser terminada por dois pontos (:) e não por ponto-e-vírgula.

Pode haver uma ou mais instruções seguindo cada **case**. Estas instruções não necessitam estar entre chaves.

O corpo de um **switch** deve estar entre chaves.

Se um caso for igual ao valor da expressão, a execução começa nele.

Se nenhum caso for satisfeito, o controle do programa sai do bloco **switch**, a menos que exista um caso *default*. Se existir, a execução começa nele.

Os rótulos dos casos devem ser todos diferentes.

O comando **break** causa uma saída imediata do programa. Se não houver um **break** seguindo as instruções do caso, o programa segue executando todas as instruções dos casos abaixo, até encontrar um **break** ou o fim do corpo do **switch**.

3.6.2 Exemplo

```
int mes;

 ...

switch(mes)
{
    case 1:
        printf("Janeiro\n");
        break;
    case 2:
        printf("Fevereiro\n");
        break;
    case 3:
        printf("Marco\n");
        break;
    case 4:
        printf("Abril\n");
        break;
```

```
        case 5:
            printf("Maio\n");
            break;
        case 6:
            printf("Junho\n");
            break;
        case 7:
            printf("Julho\n");
            break;
        case 8:
            printf("Agosto\n");
            break;
        case 9:
            printf("Setembro\n");
            break;
        case 10:
            printf("Outubro\n");
            break;
        case 11:
            printf("Novembro\n");
            break;
        case 12:
            printf("Dezembro\n");
            break;
        default:
            printf("O numero nao equivale a nenhum mes\n");
            break;
    }
```