Chapter 3
Flow of Control

Control Structures
All programs written in terms of 3 control structures:
- Sequence structures: Built into C. Programs executed sequentially by default.
- Selection structures: C has three types: `<if`, `if/else`, and `switch`
- Repetition structures: C has three types: `while`, `do/while` and `for`

Relational Operators
- The relational operators are `<`, `>`, `<=`, and `>=`.
- They take 2 expressions as operands and yield either the `int` value 0 (false) or the `int` value 1 (true).
- Valid
  - `a < 3`
  - `a <= b`
  - `a >= b`
- Invalid
  - `a > b`
  - `a < = b`
- Examples: Assume `a = 1`, `b=2`.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a &lt;= b</code></td>
<td>1</td>
</tr>
<tr>
<td><code>a =&lt; b</code></td>
<td>0</td>
</tr>
<tr>
<td><code>a + 10 / b &lt;= -3 + 8</code></td>
<td>0</td>
</tr>
</tbody>
</table>

Equality Operators
- The equality operators are `==` and `!=`.
- Yield either the `int` value 0 or the `int` value 1.
- Valid
  - `x != -2.77`
  - `x == 3.3/x`
  - `ch == '*'`
- Invalid
  - `x = = y-1`
  - `x = 44`
  - `ch = 'c'`
- Examples: Assume `a=1`, `b=2`, `ch = 'A'`

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a == b</code></td>
<td>0</td>
</tr>
<tr>
<td><code>a != b</code></td>
<td>1</td>
</tr>
<tr>
<td><code>ch &lt; 'B'</code></td>
<td>1</td>
</tr>
<tr>
<td><code>a+b == -2 * 3</code></td>
<td>0</td>
</tr>
</tbody>
</table>
Equality Operators

- Note carefully that the two expressions 
  \[ a == b \quad \text{and} \quad a = b \]

  are visually similar.

- The expression \( a == b \) is a test for equality and \( a = b \) is an assignment expression.

Logical Operators

- The logical operators are &&, ||, and !.
- Expressions connected by && or || are evaluated left to right.
- Logical negation: 
  
  \[
  \begin{array}{c|c}
  \text{Value of expression} & \text{expression} \\
  \hline
  \text{zero} & 0 \\
  \text{nonzero} & 1 \\
  \end{array}
  \]

- Examples

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>!(a&lt;b) &amp;&amp; c</td>
<td>1</td>
</tr>
<tr>
<td>ch &gt;= 'a' &amp;&amp; ch &lt;= 'z'</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td></td>
</tr>
<tr>
<td>a &lt; b &amp;&amp; x &lt; y</td>
<td>0</td>
</tr>
<tr>
<td>a &lt; b</td>
<td></td>
</tr>
</tbody>
</table>

The precedence of && is higher than ||, but both operators are of lower precedence than all unary, arithmetic and relational operators. Their associativity is left to right.

Examples

- Given declarations:
  
  ```
  int a =3, b = 3, c =3;
  double x = 0.0, y =2.5;
  char ch = 'g'
  ```

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>!(a&lt;b) &amp;&amp; c</td>
<td>1</td>
</tr>
<tr>
<td>ch &gt;= 'a' &amp;&amp; ch &lt;= 'z'</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td></td>
</tr>
<tr>
<td>a &lt; b &amp;&amp; x &lt; y</td>
<td>0</td>
</tr>
<tr>
<td>a &lt; b</td>
<td></td>
</tr>
</tbody>
</table>
### Short-Circuit Evaluation

- For the expressions that contain the operands of `&&` and `||`, the expression process stops as soon as the outcome true or false is known.
- Suppose `expr1` is 0.
  
  \[ expr1 \land expr2 = 0 \]  
  (expr2 will not be evaluated.)
- Suppose `expr1` is nonzero.
  
  \[ expr1 \lor expr2 = 1 \]  
  (expr2 will not be evaluated.)

### Compound and Empty Statements

- \{ 
  
  \begin{align*}
  &a = 1; \\
  &\{b = 2; \quad c = 3;\} \\
  \}
  
- a = b;
- a + b + c;
- 
- \textbf{printf("Hello");}

### The `if` Selection Structure

- Selection structure:
  - Used to choose among alternative courses of action
  - e.g.
    
    \textit{If student’s grade is greater than or equal to 60} 
    
    \textit{Print “Passed”}
  - If condition **true**
    
    - Print statement executed and program goes on to next statement
    
    - If **false**, print statement is ignored and the program goes onto the next statement

### if Statement

- A decision can be made on any expression.

- \textbf{false}
  - \textbf{true}

- Example:
  
  \textbf{3 = 4: true}

- \textbf{printf("Passed\n");}

  \textbf{printf("Bye!\n");}
/* Determines if a number is even */
#include <stdio.h>
int main(void)
{
  int value;
  printf("Enter a number.\n");
  scanf(%d", &value);
  if (value % 2 == 0)
    printf("%d is an even number.\n";)
  return 0;
}
Examples

- Compound statement:
  ```c
  if ( grade >= 60 )
  printf("Passed.\n");
  else {
    printf("Failed.\n");
  printf("You must take this course again.\n");
  }
  ```

- Dangling else: an else attaches to the nearest if.
  ```c
  if (a == 10)
  if (b==20)
  printf("***\n");
  else
  printf("###\n");
  ```

Problem Solving

// Find the minimum of three values.
#include <stdio.h>
int main()
{
    int a, b, c, min;
    printf("Enter three numbers:\n");
    scanf("%d%d%d", &a,&b,&c);
    if (a < b)
    min = a;
    else
    min = b;
    if (c < min)
    min = c;
    printf("The minimum value is %d\n", min);
    return 0;
}

Nested if/else structures

- Test for multiple cases by placing if/else selection structures inside if/else selection structures.
- Once condition is met, the rest of statements skipped.
- Its general form is:
  ```c
  if (expr1)
    statement1
  else if (expr2)
    statement2
  else if (expr3)
    statement3
  ... else if (exprN)
    statementN
  else
    default statement
    next statement
  ```

Nested if's

```c
  if (grade >= 90)
    printf("A\n");
  else if (grade >= 80)
    printf("B\n");
  else if (grade >= 70)
    printf("C\n");
  else if (grade >= 60)
    printf("D\n");
  else
    printf("F\n");
```
The Ternary Conditional Operator (?)

- Takes three arguments (condition, value if true, value if false)
- e.g:
  
  \[
  \text{printf("%s\n", grade >= 60 ? "Passed" : "Failed" ;)}
  \]
- Or it could have been written:
  
  \[
  \text{grade >= 60 ? printf( "Passed\n" ) : printf( "Failed\n" );}
  \]
- Or it could be used in an assignment statement:
  
  \[
  \text{letter = (grade >= 60) ? 'S' : 'U';}
  \]

The switch Multiple-Selection Structure

- **switch**
  - Useful when a variable or expression is tested for all the values it can assume and different actions are taken
- **Format**
  - Series of case labels and an optional default case
  
  \[
  \text{switch ( a_variable ){
  case value1:
  actions
  case value2:
  actions
  ...
  default:
  actions
  }}
  \]
  
  - **break**: exits from structure

/* Counting letter grades */

```
int grade;
int aCount = 0, bCount = 0, cCount = 0, dCount = 0, fCount = 0;
printf("Enter the letter grade.\n"  );
scanf("%c",&grade);
switch ( grade ) {
    case 'A':
    case 'a': ++aCount;
    break;
    case 'B':
    case 'b': ++bCount;
    break;
    case 'C':
    case 'c': ++cCount;
    break;
    case 'D':
    case 'd': ++dCount;
    break;
    case 'F':
    case 'f': ++fCount;
    break;
    default:    /* catch all other characters */
    printf( "Incorrect letter grade entered.\n" );
    printf( " Enter a new grade.\n" );
    break;
}
```

/* Counting letter grades */

```
int grade;
int aCount = 0, bCount = 0, cCount = 0, dCount = 0, fCount = 0;
printf("Enter the letter grade.\n"  );
scanf("%c",&grade);
switch ( grade ) {
    case 'A':
    case 'a': ++aCount;
    break;
    case 'B':
    case 'b': ++bCount;
    break;
    case 'C':
    case 'c': ++cCount;
    break;
    case 'D':
    case 'd': ++dCount;
    break;
    case 'F':
    case 'f': ++fCount;
    break;
    default:    /* catch all other characters */
    printf( "Incorrect letter grade entered.\n" );
    printf( " Enter a new grade.\n" );
    break;
}
```
```c
int main()
{
    int month, year, days, leapyear;

    printf("Enter a month and a year:");
    scanf("%d%d", &month, &year);
    if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
        leapyear = 1;
    else
        leapyear = 0;

    switch (month) {
    case 9: case 4: case 6: case 11:
        days = 30;
        break;
    case 2:
        days = (leapyear == 1)? 29: 28;
        break;
    default:
        days = 31;
    }
    printf("There are %d days in that month in that year.\n", days);
}
```