Chapter 3: Control Flow

- Control flow statements specify the order in which computations are performed.

3.1 Statements and Blocks

- An expression becomes a *statement* when it is followed by a semicolon.
    - Example: `x = 4` is an expression, `x = 4;` is a statement.
- In C the semicolon is a statement terminator.
- Braces are used to group declarations and statements together into a *compound statement*, also known as a *block*.
    - This makes them syntactically equivalent to a single statement.
    - There is no semicolon after the right brace that ends a block.

3.2 If-Else

- The if-else statement is used to express decisions.
- Syntax:

```
if (expression)
    statement1
else
    statement2
```

- The else part is optional.
- Operation:
    - The expression is evaluated and if true then statement1 is executed.
    - If the else is included then if expression is false statement2 is

    executed.

- The else is associated with the closest previous if statement lacking an else.
    - If the else is needed for a different if then brackets must be used to group the blocks appropriately.

3.3 Else-If

- Add an if after the else in the statement above for multiple conditions.

3.4 Switch

- Multi-way decision that tests if an expression matches a constant integer value.

3.5 Loops - While and For

- Syntax for a while statement:

```
while (expression)
    statement
```

- The *expression* is evaluated and if it is non-zero the *statement* is executed.
    - The process is repeated until the *expression* is zero.
- Syntax of the for statement

```
for (expr1; expr2; expr3)
    statement
```

- The for statement is equivalent to a while statement with expr2 as the expression and expr3 as the control statement.

- We can omit the expr2 control in the for statement `for(;;)` which will create an infinite loop unless it is broken otherwise.
- We can also use the comma operator ','
    - most often used in for statements.
    - a pair of expressions separated by a comma is evaluated left to right.
        - the type and value of the output matches the right operand.
    - We can use this to place multiple conditions in an expression of the for loop.

3.6 Loops - Do-While

- Tests an expression after the statement is executed to restart the loop.

3.7 Break and Continue

- *break* allows exiting a loop without testing the expression again.
- *continue* causes the next iteration of a loop to begin immediately.

3.8 Goto and Labels

- *goto* allows moving to a different portion of the code.
- *labels* allow marking the portions to move to.
- Code using a goto statement can always be written without one. *