CONTROL STRUCTURES

Prepared by Caroline Natalie M. Peralta

Control-Flow Statements

The simplest C programs usually consist of a main function containing all the statements to be executed. These statements are executed ***sequentially***. However, for more complex algorithms, there is a need to execute statements only if a certain condition is fulfilled, or to execute a set of statements repeatedly for as long as a condition is true. For these purposes, the C programming language has a set of ***control-flow statements*** that can alter the default order of execution of statements of a program.

Control Flow

Control flow is the order in which statements are executed in a program. There are three types of control flow: ***sequence***, ***selection***, and ***iteration***.

Sequence

This type of control flow means that the statements are executed one after another. This is the default control flow for C programs. Figure 1 shows a flowchart of a set of statements executed sequentially.

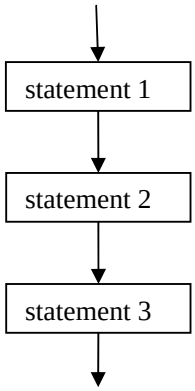

Figure 1

Selection

Selection is the type of control flow wherein the next statement to be executed is dependent on the outcome of a certain condition. There are two selection control-flow statements in C: the ***`if-else`*** statement and the ***`switch`*** statement.

If-Else Statement

The **`if-else`** statement evaluates a conditional expression before choosing the next statement to be executed based on that conditional expression.

The format of an `if-else` statement is as a follows:

```
if(expression) {
  //if body
} else {
  //else body
}
```

The `else` clause of an `if-else` statement is optional. Therefore, **an `if` clause can exist without an `else` clause, but an `else` clause cannot exist without an `if` clause**. Also, both the `if` body and the `else` body can contain numerous statements to be executed.

Nested If-Else Statements

`if-else` statements can exist within the `if` and `else` bodies of another `if-else` statement.

```
if(expression1) {
  //statements
  if(expression2) {
    //statements
  }
  //statements
} else {
  //statements
  if(expression3) {

  } else {
    //statements
  }
  //statements
}
```

If-Else-If Statements

Additional conditions may be added to `else` clauses if they are needed. For example:

```
if(expression1) {
  //statements
} else if(expression2) {
  //statements
} else if(expression3) {
  //statements
} ...
```

Unlike simple `if-else` statements, it is possible for none of the branches of the `if-else-if` statement to execute if none of the conditional expressions are true.

Switch Statement

The **`switch`** statement tests a variable or expression against a number of possible values or test cases. A `switch` statement can only test **equality** between the variable or expression being tested and its test cases. The format of a `switch` statement is as follows:

```
switch(expression or variable) {
  case a:
    //statements
    break;
  case b:
    //statements
    break;
  default:
    //statements
}
```

Each test case can only take the form of the **`int`** and **`char`** data types. The default clause is only executed when none of the test cases is true. Also, a **`break`** statement should be issued after the execution of the statements under one test case, otherwise the execution will continue onto the next case without testing the expression again.

Iteration

Iteration is the type of control flow that allows sets of statements to be executed repeatedly based on the outcome of a conditional expression. C provides three different iterative statements (also known as loops): the ***`while`*** loop, the ***`do-while`*** loop, and the ***`for`*** loop.

For each iterative statement, there should be a **control variable**. The execution of the iterative statement will depend on the value of the control variable. Therefore, each iterative statement should always have three different parts:

1. Initialization – the starting point for the loop's control variable.
2. Condition – the condition that must be true for the loop to continue executing.
3. Increment – the change that must be applied to the loop's control variable so that it will eventually make the condition false.

while Statement

The **`while`** loop repeatedly executes a set of statements for as long as the conditional expression is evaluated to be true. The expression is evaluated first before the statements in the loop's body are executed. Therefore, **it is possible for the `while` loop to not execute at all** if the condition is immediately evaluated to be false. The format of the `while` loop is as follows:

```
//initialization
while(condition) {
  //statements
  //increment
}
```

do-while Statement

The **`do-while`** loop functions in the same way as the while loop, except that it executed the loop body's statements first before evaluating the expression. Therefore, a `do-while` loop **will always execute at least once**. The format of the `do-while` loop is as follows:

```
//initialization
do {
  //statements
  //increment
} while (condition);
```

for Statement

The **`for`** loop is a specialized loop which is especially useful if the number of iterations for the loop can be determined before the execution of the loop. In a `for` loop, the initialization, condition, and increment are all centralized at the start of the loop. Like the while loop, **it is possible for the `for` loop's body to not execute at all** if the condition is immediately evaluated to be false. The format of the `for` loop is as follows:

```
for(initialization; condition; increment) {
  //statements
}
```

break Statement

The ***`break`*** statement stops the execution of a control structure. This is mostly used for loops and only when certain conditions are met that will require the immediate termination of the loop's execution.

continue Statement

The ***`continue`*** statement stops the current iteration of a loop and moves onto the next iteration. If the continue statement is used within a for loop, the increment is still applied. Before starting the next iteration, the condition is still tested. Like the `break` statement, it is only used when certain conditions are met.