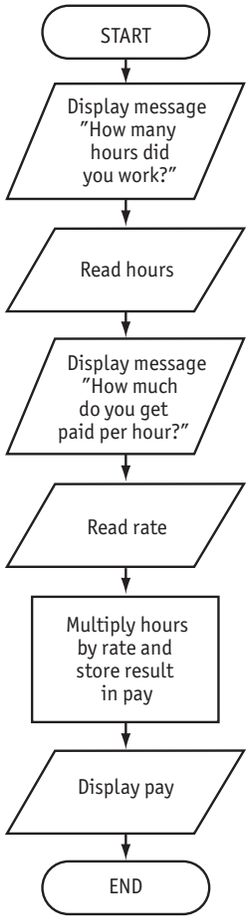
APPENDIX N

Introduction to Flowcharting

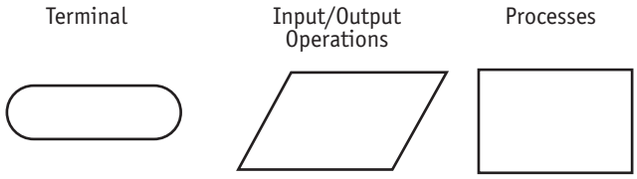
This appendix provides a brief introduction to flowcharting. It includes example flowcharts for some of the programs that appear in Chapters 1–6.

A flowchart is a diagram that depicts the "flow" of a program. It contains symbols that represent each step in the program. Figure N-1 is a flowchart for Program 1-1, the pay-calculating program in Chapter 1.

Figure N-1



Notice there are three types of symbols in this flowchart: rounded rectangles (representing terminal points), parallelograms (representing input/output operations), and a rectangle (representing a process).



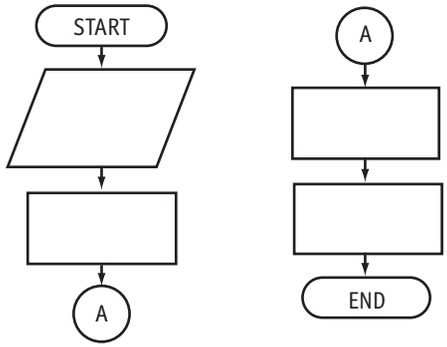
The rounded rectangles, or terminal points, indicate the flowchart's starting and ending points. The parallelograms designate input or output operations. The rectangle depicts a process such as a mathematical computation, or a variable assignment. Notice that the symbols are connected with arrows that indicate the direction of program flow.

Connectors

Sometimes a flowchart is broken into two or more smaller charts. This is usually done when a flowchart does not fit on a single page, or must be divided into sections. A connector symbol, which is a small circle with a letter or number inside it, allows you to connect two flowcharts.

In Figure N-2 the A connector indicates that the second flowchart segment begins where the first flowchart segment ends.

Figure N-2



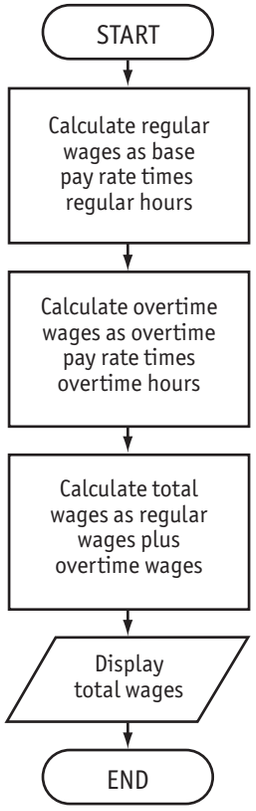
Flowchart Structures

There are four general flowchart structures:

- Sequence
- Decision
- Case
- Repetition

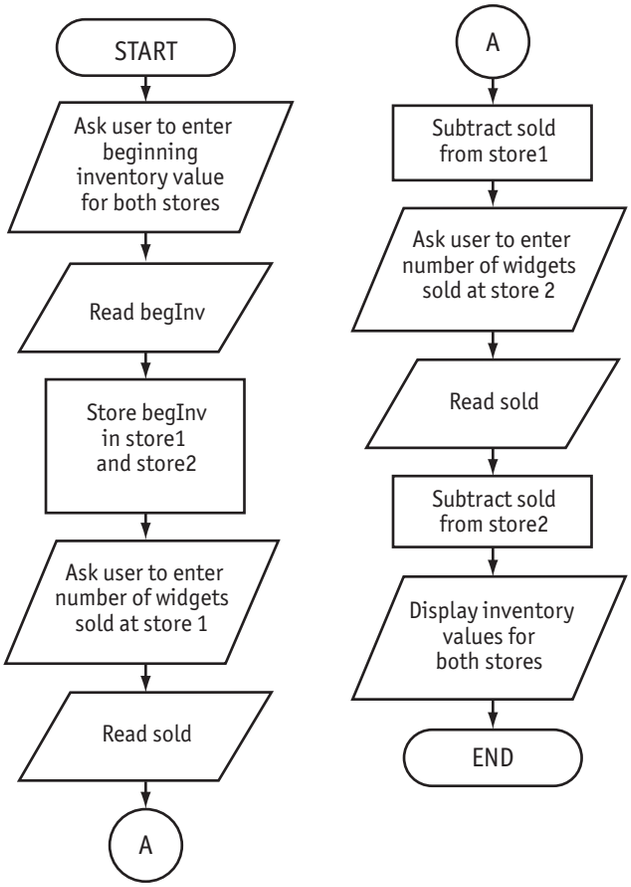
A sequence structure indicates a series of actions or steps, performed in order. The flowchart for the pay-calculating program is an example of a sequence structure. The flowchart shown in Figure N-3 below is also a sequence structure. It depicts the steps performed in Program 2-20, from Chapter 2.

Figure N-3



The flowchart shown in Figure N-4, which is another sequence structure, depicts the steps performed in Program 3-11, the inventory program in Chapter 3. Notice the use of connector symbols to link the two flowchart segments.

Figure N-4



The Decision Structure

In a decision structure the next action to be performed depends on the outcome of a true/false test. A statement that is either true or false is evaluated. In the simplest form of decision structure, if the result of the test is true, a set of instructions is executed and if the result of the test is false, these instructions are skipped. In a flowchart a new symbol, the diamond, is used to represent the true/false statement being tested. Figure N-5 shows the general form of this decision structure.

Figure N-5

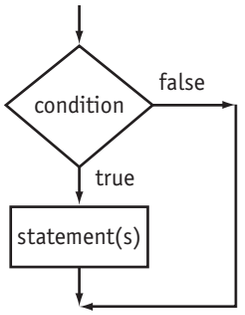
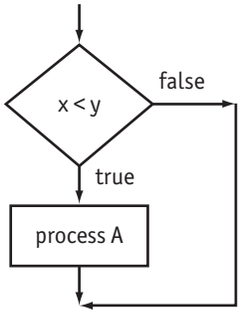


Figure N-6 illustrates the use of this decision structure. In this flowchart the statement, also called a condition, `x < y` is tested. If this condition is true, the statements in process A are executed. If the condition is false, the process A statements are skipped. This decision structure is coded in C++ as an `if` statement.

Figure N-6



A slightly more complex form of the decision structure allows one set of statements to be executed if the test condition is true and another set to be executed if the test condition is false. Figure N-7 illustrates this form of the decision structure.

Figure N-7

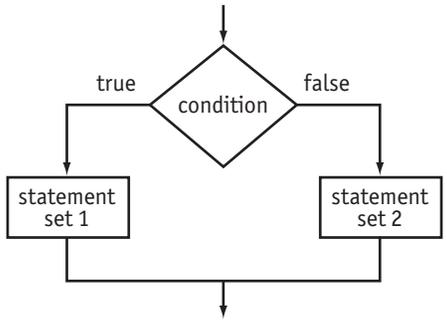


Figure N-8 shows a flowchart segment in which the statement `x < y` is again tested. But this time a different *process* is executed depending on whether `x < y` evaluates to true or false. If it is true, the statements that make up process A are executed. If it is false, the statements that make up process B are executed instead.

Figure N-8

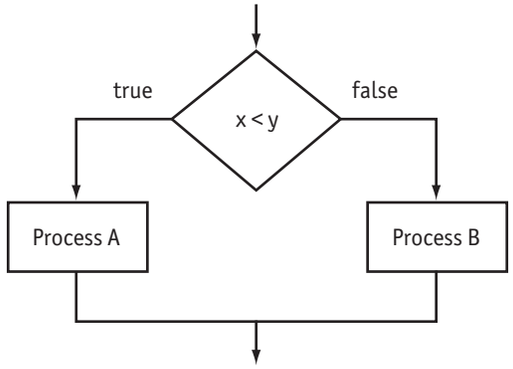
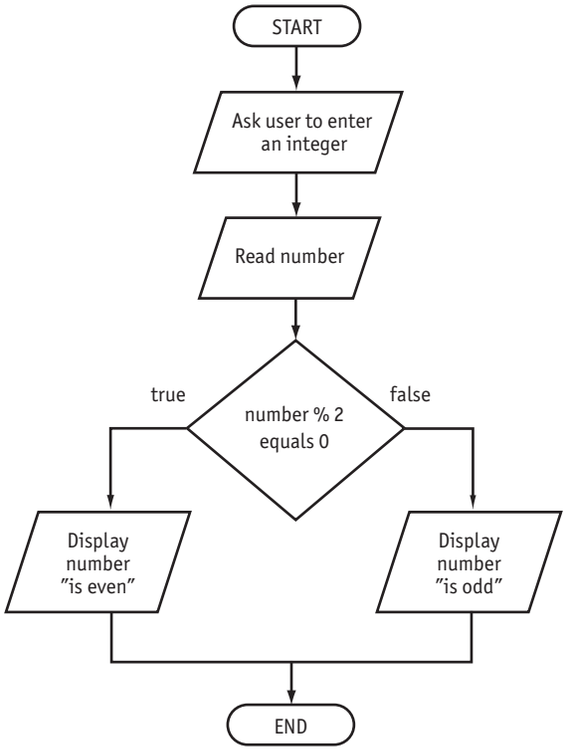


Figure N-9 shows a flowchart that depicts the logic of Program 4-3, from Chapter 4. The decision structure shows that one of two possible messages is displayed on the screen, depending on the value of the expression `number % 2`. Notice in Program 4-3 that C++ uses an `if/else` statement to implement this decision structure.

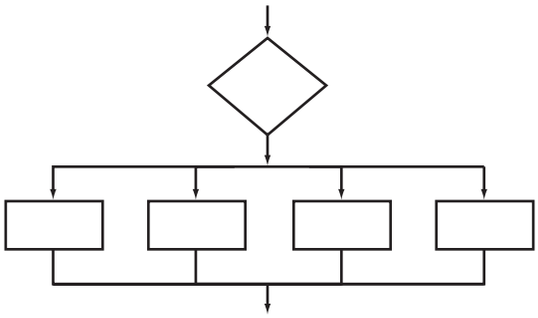
Figure N-9



The Case Structure

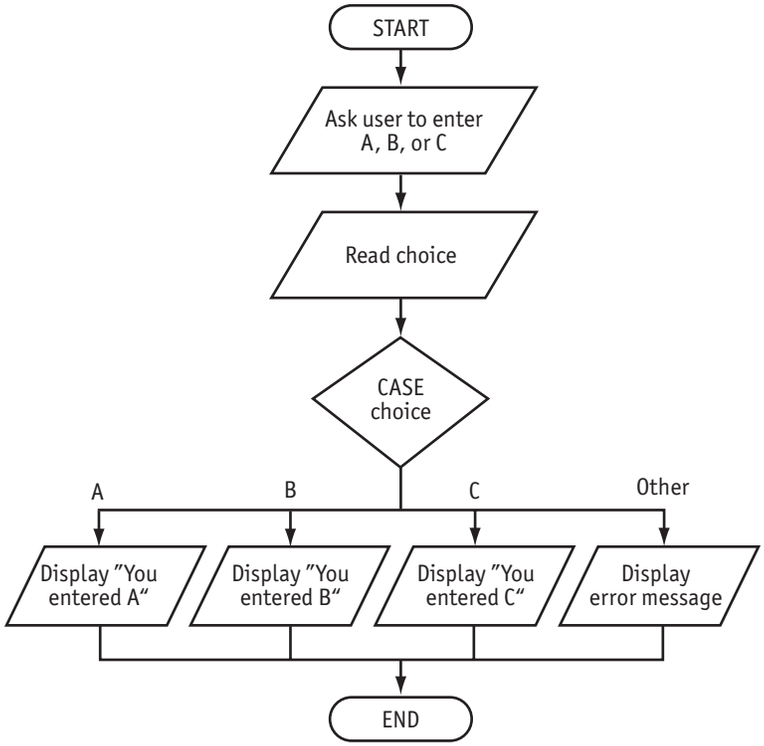
A case structure is a still more complex form of decision structure in which many different possible actions can be taken depending on the value of an integer or character variable or an integer expression. Figure N-10 shows the general form of a case structure.

Figure N-10



The flowchart shown in Figure N-11 depicts the logic of Program 4-23, which is also from Chapter 4. Program 4-23 illustrates how C++ uses a `switch` statement to implement the case structure. In this program, one of four possible paths is followed, depending on the value stored in the variable `choice`.

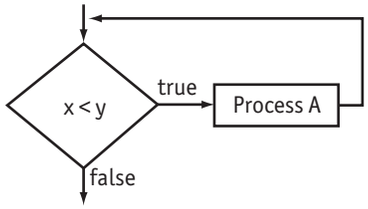
Figure N-11



Repetition Structures

A repetition structure represents a block of program code that repeats. This type of structure is commonly known as a loop. Figure N-12 shows an example of a repetition structure.

Figure N-12



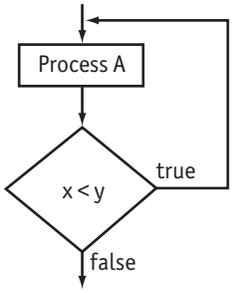
Notice the use of the diamond symbol. A repetition structure tests a statement that indicates whether or not some condition exists. If the statement is true, meaning the condition exists, a set of actions is performed. Then the statement is tested again. If it is still true, the condition still exists and the actions are repeated. This continues until the statement is no longer true.

In the flowchart segment just shown, the statement `x < y` is tested. If it is true, then the statements that make up process A are executed and `x < y` is evaluated again. Process A is repeated as long as x is less than y. When x is no longer less than y, the repetition stops and the structure is exited.

There are two forms of the repetition structure: pre-test and post-test. The pre-test structure tests its condition *before* it performs an action or set of actions. The flowchart segment we have just examined illustrates a pre-test structure. Notice that Process A does not execute at all if the condition x < y is not true to begin with. The pre-test repetition structure is coded in C++ as either a `while` loop or a `for` loop.

A post-test repetition structure tests its condition *after* it performs an action or set of actions. This type of loop always executes its code at least once. Figure N-13 shows a flowchart segment representing a post-test structure.

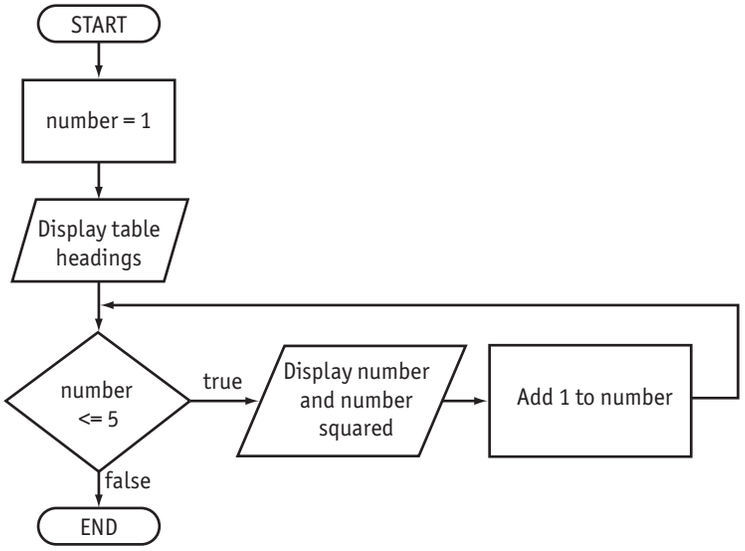
Figure N-13



The post-test repetition structure is coded in C++ as a `do-while` loop.

Figure N-14 shows a flowchart depicting the logic of Program 5-6, which appears in Chapter 5. It uses a pre-test loop.

Figure N-14

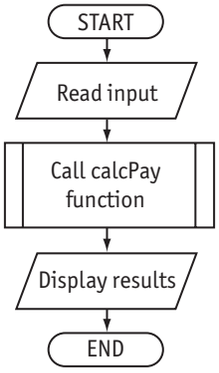


Modules

A program module (such as a function in C++) is represented in a flowchart by this symbol.

Figure N-15 illustrates its use. The code in the module is executed at the point in the program where the symbol is located in the flowchart.

Figure N-15



A separate flowchart can then be constructed for the module. Figure N-16 shows a flowchart for the `main` function of Program 6-14, the health club membership program that appears in Chapter 6. Notice how it "calls" several other modules, causing them to execute their code. Each of these other modules carries out some of the activities of the program.

Figure N-16