

4.1.1 Flow of Control

In 1966, Böhm and Jacopini published a paper entitled *Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules* [7]. It wasn't too long before the computer science community realized that their work implied that any algorithm can be expressed using what are now called *structured control* constructs: sequence, selection, and repetition. The essential idea of structured control is that subcollections of steps need to have a single entry and a single exit. This was in sharp contrast with the then current practice of using many *goto* statements (step 4 in Example 4.1). Pseudocode allows *goto*'s, but they are discouraged. The three major structured control categories are described next.

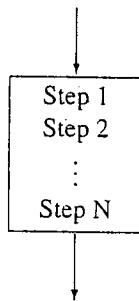


Figure 4.1. A sequential block.

Sequence

The simplest control structure is *sequence*. The convention is to read algorithms from top to bottom, completing each step in order unless directed otherwise. Sequential control is the default. You would most likely perform a purely sequential algorithm without even realizing that you were following this simple convention.

A sequential block of instructions (Figure 4.1) clearly has a single entry (just above the first instruction in the sequence) and a single exit (just after the final instruction in the block).

Selection

Selection control constructs allow the algorithm to take different paths for different initial data. For our purposes, it is useful to consider one-way, two-way, and multiway selection.

One-Way Selection One-way selection allows some steps to be completed conditionally. That is, sometimes the steps will be completed and other times they will be skipped. The pseudocode that achieves this is the *if-then* construct:

```

if condition then
  Step 1
  Step 2
  :
  Step N
    
```

The indented steps (1 – N here) are called the *body* of the if-then construct. The steps in the body are conditionally executed² (depending on the value of *condition*).

Often, the word *then* is omitted in the pseudocode notation. (That convention will be followed in this text.)

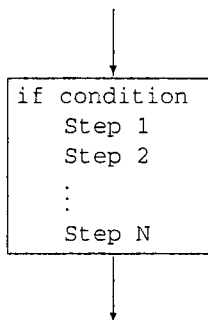


Figure 4.2. An if-then block.

```

if condition
  Step 1
  Step 2
  :
  Step N
    
```

Steps 1–N are completed only if the proposition *condition* evaluates to *true*. The indentation is used to indicate that all N of the steps are dependent on *condition* being true.

The if-then construct (Figure 4.2) is a single-entry, single-exit structure; the exit is the step immediately following the final step in the if-then (independent of whether the statements in the if-then body are actually executed).

Two-Way Selection In a two-way selection, if the proposition *condition* is *true*, the first set of steps are completed; otherwise, the second set of steps are completed. The indentation is again an important part of the notation.

```

if condition then
    Step 1
    Step 2
    :
    Step N
else
    Step N+1
    Step N+2
    :
    Step N+M

```

or

```

if condition
    Step 1
    Step 2
    :
    Step N
else
    Step N+1
    Step N+2
    :
    Step N+M

```

The two-way selection (also called an *if-else* construct) is a single-entry, single-exit control structure (Figure 4.3). It doesn't matter whether the *true* path (then) or the *false* path (else) is chosen; the step after the if-else will always be the one immediately after the structure.

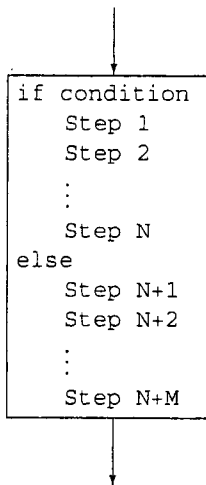


Figure 4.3. An if-else block.

Multiway Selection A multiway selection construct provides more than two mutually exclusive options (Figure 4.4). In a multiway selection, we can pick from among

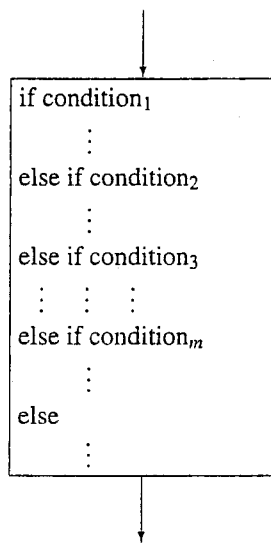


Figure 4.4 A multiway selection block.

many mutually exclusive options. We may also optionally provide a default choice if none of the other options are appropriate. One way to express this in pseudocode is as follows:

```

if condition1
    :
else if condition2
    :
else if condition3
    : : :
else if conditionm
    :
else
    :

```

or

```

if condition1
    :
else if condition2
    :
else if condition3
    : : :
else if conditionm
    :

```

where the final *else* section may be omitted if there is no default collection of steps.

As usual, there is a single entry and a single exit. Once one of the conditions is true, the steps in that section are executed and then the first step after the multiway structure is the next executed.

Repetition

The final control structure category consists of those that repeat a collection of statements. These structures can repeat for either a predetermined number of times or for a variable number of times (determined by a condition).

Fixed Iteration When the number of times a collection of steps needs to be executed is known (or is stored in a variable), the proper repetition structure is a fixed iteration. One of the many possible pseudocode expressions of this structure is a simple *for* loop. The *for* loop uses an index variable to count how many times the loop has been executed.

Each time through the loop, the index variable (*i* in the following pattern) is incremented. The loop is complete when the index variable gets larger than its terminal value (*n* in the pattern). Every time step *m* is executed, control goes back to the top of the *for* loop. If *i* is still less than or equal to *n*, the loop body is executed again. Otherwise, the next statement will be the one immediately after step *m*.

```
for i = 1 to n
  Step 1
  Step 2
  :
  Step m
```

The indentation highlights the body of the loop (the statements that are repeated). Note that the body of the loop is using the sequence structure.

Indefinite Iteration There are two common control structures that allow a loop to be executed until some condition is met. The more common is the *while* loop, which uses a pretest to determine when to quit. The less common *repeat-until* loop uses a post-test to determine when to exit the loop.

while condition	repeat
Step 1	Step 1
Step 2	Step 2
:	:
Step N	Step N
	until condition

The *while* loop tests the condition before entering the body of the loop. If the condition is false, the loop body is not executed and the first statement after step *N* is the next executed.³ Otherwise, the loop body is executed and then control goes back to the top, where the condition is once again checked.

The *repeat-until* loop always executes the loop body at least once. After the loop body has been executed, the condition is checked. If the condition is false, the loop body is repeated again; otherwise the loop is terminated and the first step after the *until* line is the next to be executed.

Note the difference: A *while* loop keeps executing as long as the condition is true; a *repeat-until* loop keeps executing as long as the condition is false.

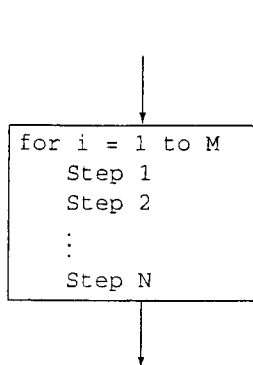


Figure 4.5 A *for* block.

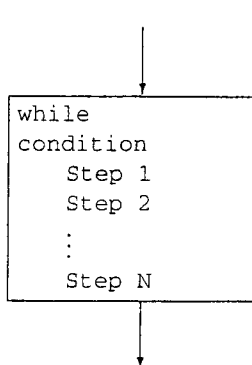


Figure 4.6 A *while* block.

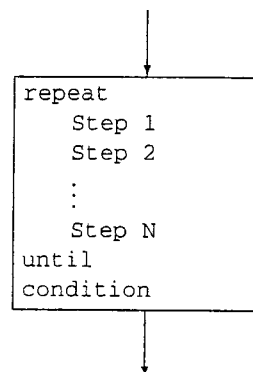


Figure 4.7 A *repeat* block.

Library of Congress Cataloging in Publication Data

Gossett, Eric

Discrete mathematics with proof / Eric Gossett.

p. cm.

Includes bibliographical references and index.

ISBN 0-13-066948-2

1. Mathematics. 2. Computer science--Mathematics. I. Title

QH39.3 .G68 2003 2003

510--dc21

2002032965

CIP