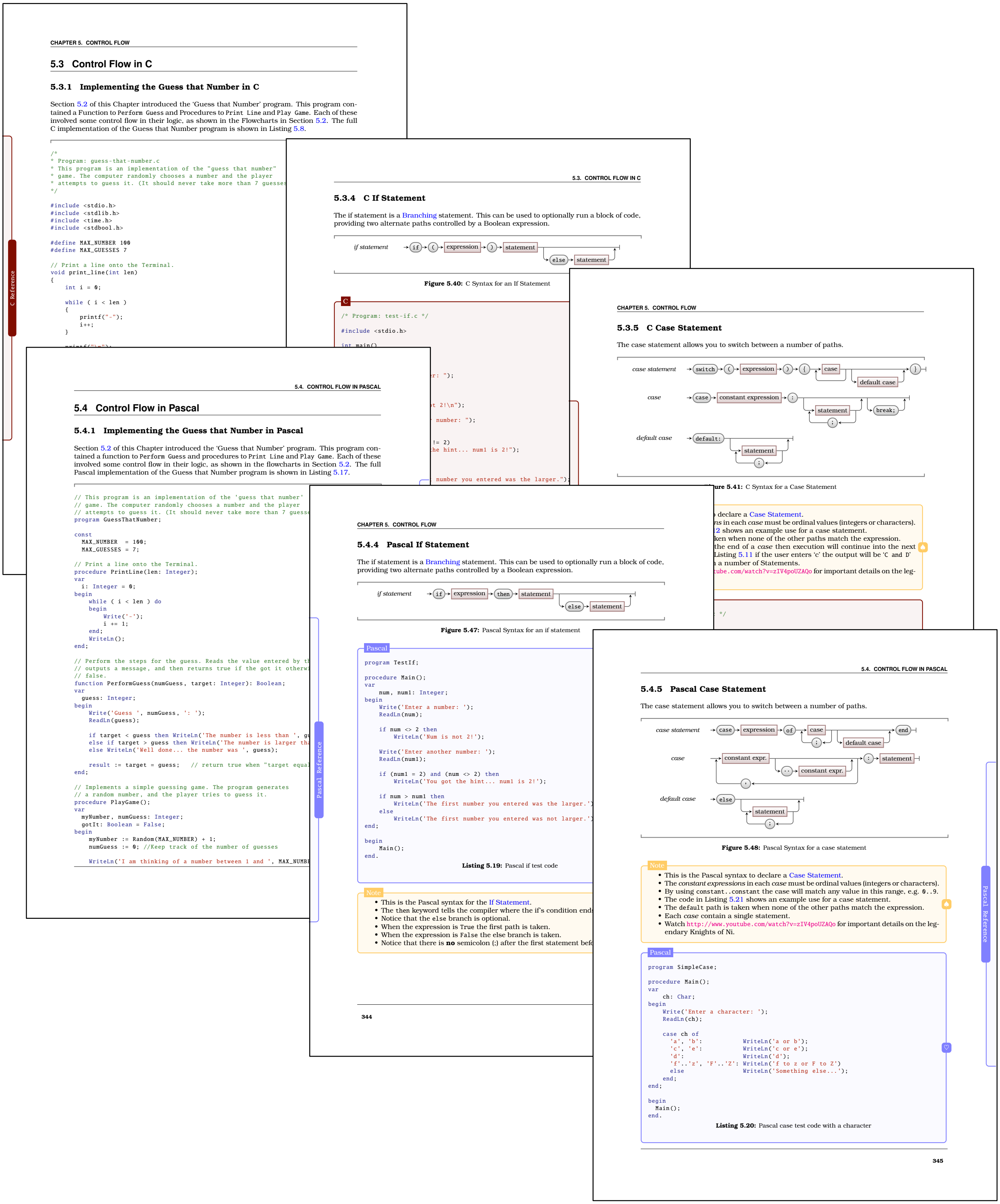

5.3 Control Flow in C

5.3.1 Implementing the Guess that Number in C

Section 5.2 of this Chapter introduced the 'Guess that Number' program. This program contained a Function to Perform Guess and Procedures to Print Line and Play Game. Each of these involved some control flow in their logic, as shown in the Flowcharts in Section 5.2. The full C implementation of the Guess that Number program is shown in Listing 5.8.

```
/*
* Program: guess-that-number.c
* This program is an implementation of the "guess that number"
* game. The computer randomly chooses a number and the player
* attempts to guess it. (It should never take more than 7 guesses
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

#define MAX_NUMBER 100
#define MAX_GUESSES 7

// Print a line onto the Terminal.
void print_line(int len)
{
    int i = 0;

    while ( i < len )
    {
        printf("-");
        i++;
    }
```

C Reference

5.3.4 C If Statement

The if statement is a Branching statement. This can be used to optionally run a block of code, providing two alternate paths controlled by a Boolean expression.

Figure 5.40: C Syntax for an If Statement

C

```
/* Program: test-if.c */

#include <stdio.h>

int main()
```

```
r: ");
t 2!\n");
 number: ");
!= 2)
he hint... num1 is 2!");
 number you entered was the larger.");
```

5.3.5 C Case Statement

The case statement allows you to switch between a number of paths.

ure 5.41: C Syntax for a Case Statement

o declare a Case Statement.
ns in each *case* must be ordinal values (integers or characters).
2 shows an example use for a case statement.
ken when none of the other paths match the expression.
the end of a *case* then execution will continue into the next
Listing 5.11 if the user enters 'c' the output will be `C and D`
a number of Statements.
ube.com/watch?v=zIV4poUZAQo for important details on the leg-

```
*/
```

5.4 Control Flow in Pascal

5.4.1 Implementing the Guess that Number in Pascal

Section 5.2 of this Chapter introduced the 'Guess that Number' program. This program contained a function to Perform Guess and procedures to Print Line and Play Game. Each of these involved some control flow in their logic, as shown in the flowcharts in Section 5.2. The full Pascal implementation of the Guess that Number program is shown in Listing 5.17.

```
// This program is an implementation of the 'guess that number'
// game. The computer randomly chooses a number and the player
// attempts to guess it. (It should never take more than 7 guesse
program GuessThatNumber;

const
  MAX_NUMBER  = 100;
  MAX_GUESSES = 7;

// Print a line onto the Terminal.
procedure PrintLine(len: Integer);
var
  i: Integer = 0;
begin
    while ( i < len ) do
    begin
        Write('-');
        i += 1;
    end;
    WriteLn();
end;

// Perform the steps for the guess. Reads the value entered by th
// outputs a message, and then returns true if the got it otherwi
// false.
function PerformGuess(numGuess, target: Integer): Boolean;
var
  guess: Integer;
begin
    Write('Guess ', numGuess, ': ');
    ReadLn(guess);

    if target < guess then WriteLn('The number is less than ', gu
    else if target > guess then WriteLn('The number is larger tha
    else WriteLn('Well done... the number was ', guess);

    result := target = guess;   // return true when "target equal
end;

// Implements a simple guessing game. The program generates
// a random number, and the player tries to guess it.
procedure PlayGame();
var
  myNumber, numGuess: Integer;
  gotIt: Boolean = False;
begin
    myNumber := Random(MAX_NUMBER) + 1;
    numGuess := 0; //Keep track of the number of guesses

    WriteLn('I am thinking of a number between 1 and ', MAX_NUMBE
```

Pascal Reference

5.4.4 Pascal If Statement

The if statement is a Branching statement. This can be used to optionally run a block of code, providing two alternate paths controlled by a Boolean expression.

Figure 5.47: Pascal Syntax for an if statement

Pascal

```
program TestIf;

procedure Main();
var
    num, num1: Integer;
begin
    Write('Enter a number: ');
    ReadLn(num);

    if num <> 2 then
        WriteLn('Num is not 2!');

    Write('Enter another number: ');
    ReadLn(num1);

    if (num1 = 2) and (num <> 2) then
        WriteLn('You got the hint... num1 is 2!');

    if num > num1 then
        WriteLn('The first number you entered was the larger.'
    else
        WriteLn('The first number you entered was not larger.'
end;

begin
    Main();
end.
```

Listing 5.19: Pascal if test code

Note

- This is the Pascal syntax for the If Statement.
- The then keyword tells the compiler where the if's condition ends
- Notice that the else branch is optional.
- When the expression is True the first path is taken.
- When the expression is False the else branch is taken.
- Notice that there is **no** semicolon (;) after the first statement befo

5.4.5 Pascal Case Statement

The case statement allows you to switch between a number of paths.

Figure 5.48: Pascal Syntax for a case statement

Note

- This is the Pascal syntax to declare a Case Statement.
- The *constant expressions* in each *case* must be ordinal values (integers or characters).
- By using `constant..constant` the case will match any value in this range, e.g. `0..9`.
- The code in Listing 5.21 shows an example use for a case statement.
- The `default` path is taken when none of the other paths match the expression.
- Each *case* contain a single statement.
- Watch http://www.youtube.com/watch?v=zIV4poUZAQo for important details on the legendary Knights of Ni.

Pascal

```
program SimpleCase;

procedure Main();
var
    ch: Char;
begin
    Write('Enter a character: ');
    ReadLn(ch);

    case ch of
      'a', 'b':           WriteLn('a or b');
      'c', 'e':           WriteLn('c or e');
      'd':                WriteLn('d');
      'f'..'z', 'F'..'Z': WriteLn('f to z or F to Z')
      else                WriteLn('Something else...');
    end;
end;

begin
  Main();
end.
```

Listing 5.20: Pascal case test code with a character

Pascal Reference