```
IF valid = True THEN
    SEND 'Valid form' TO DISPLAY
ELSE
    SEND 'The form you entered does not exist.' TO DISPLAY
END IF
```

SKILLS CRITICAL THINKING, PROBLEM SOLVING

ACTIVITY 22
IMPLEMENTING A LOOK-UP CHECK

Implement the look-up check algorithm in the high-level programming language you are studying.

LENGTH CHECK

It is sometimes necessary to check that the length of a value entered falls within a specified range. For example, all UK postcodes are between six and eight characters long, so validation could be used to check that the length of a postcode entered is within this range. Needless to say, that doesn't necessarily mean it is correct.

```
RECEIVE postCode FROM (STRING) KEYBOARD
SET length TO LENGTH(postCode)
IF length >= 6 AND length <= 8 THEN
    SEND 'Valid' TO DISPLAY
ELSE
    SEND 'Invalid' TO DISPLAY
END IF
```

SKILLS CRITICAL THINKING

ACTIVITY 23
IMPLEMENTING A LENGTH CHECK

Implement the length check algorithm in the high-level programming language you are studying.

TESTING VALIDATION RULES

It is important to test your validation rules to ensure they work as expected. You should use:

Normal data – This is data that is within the limits of what should be accepted by the program. For example, a password with seven characters fulfils the validation rule that states that passwords must be between six and eight characters in length.

Boundary data – This is data that is at the outer limits of what should be accepted by the program. For example, if a validation rule specifies that the range of acceptable values is >= 75 AND <= 100, then a value of 100 is at the upper limit and a value of 75 at the lower.

GENERAL VOCABULARY

erroneous not correct; based on incorrect information

Erroneous data – This is data that should not be accepted by the program. For example, a value of 0 should not be accepted by either of the validation rules given above.

WORKING WITH TEXT FILES

SUBJECT VOCABULARY

text file a sequence of lines, each of which consists of a sequence of characters

The programs you have created so far haven't required a huge amount of data entry, but imagine typing a set of test results for everyone in your computer science class. It would take a considerable amount of time to do and – even worse – when the program terminates, all of the data will be lost. Should you need to use it again you'd have to re-enter it. This is where storing data in an external file comes in really useful. If the data you enter is stored in an external **text file** you can access it as often as you like without having to do any further keying in.

Commas are used to separate individual values on a line and a special character is used to denote the end of a line.

Text files provide permanent storage for data. This means that the data can be reused without having to be retyped. Data can be read from, written to and appended to a file.

SUBJECT VOCABULARY

file handle a label that is assigned to a resource needed by the program. It can only access the file through the computer's operating system

overwritten if a file exists on the computer and a new file is created with the same name, the new file is kept and the old file is written over and lost

PYTHON

To access a text file, you must first give it a **file handle**, e.g. `myFile`.

Files can be opened to read from them, to write to them or to append data to them (Table 2.11).

`myFile = open('names.txt', 'r')`	Open a file named `names.txt` so that the data can be read.
`myFile = open('names.txt', 'w')`	Open a file named `names.txt` to be written to and create a new file if it does not exist. Unfortunately, if the file did exist it would **overwrite** all of its data.
`myFile = open('names.txt', 'a')`	Open a file named `names.txt` to add data to it.

▲ **Table 2.11** Using text files

After a file has been opened it must be closed. The data is not written to a file until this command is executed.

The following program would create a text file called 'names' and add two items of data.

```
myFile = open('names.txt', 'w')
myFile.write('First item')
myFile.write('Second item')
myFile.close()
```

The file it creates would be: `‘First itemSecond item’`.

There is nothing to separate the two items of data. Therefore, it is useful to add a character such as a ‘,’ or a ‘;’ between them.

```
myFile = open(‘names.txt’, ‘w’)
myFile.write(‘First item’ + ‘,’)
myFile.write(‘Second item’ + ‘,’)
myFile.close()
```

The file created would be: `‘First item,Second item’`.

It is important to separate the data items so that they can be read back into a data structure.

The following program would write the contents of an array into a text file called `cars.txt` with a comma between them.

```
Alist = [‘BMW’, ‘Toyota’, ‘Audi’, ‘Renault’, ‘Rover’]

myFile = open(‘cars.txt’, ‘w’)

for index in range(len(Alist)):
    myFile.write((Alist[index]) + ‘,’)

myFile.close()
```

The file can be read back into an array in the following way.

```
Blist = []
myFile = open(‘cars.txt’, ‘r’)

Blist = myFile.read().split(‘,’)

MyFile.close()
```

`Blist = myFile.read().split(‘,’)` reads the data and splits it into separate items where there is a ‘,’.

JAVA

Java programs that read and write to files often start with:

```
import system.io.*
```

This allows your Java code to access files such as `File` and `FileReader`, FileWriter from the `system.io` namespace.

```
import java.io.*;
class Main {
  public static void main(String[] args) {
    try {
      FileWriter fw = new FileWriter(“names.txt”);
      fw.write(“First item”);
      fw.write(“Second item”);
      fw.close();
    } catch (IOException e) {
      System.out.println(“Could not write to file”);
    }
  }
}
```

This example will write two strings to the file `names.txt`, which would then contain: `First itemSecond item`.

It is useful to separate each data value in a file with a comma (,) or new line (`\n`) so that they can then be read back into an array or opened as a comma separated value (CSV) file to be edited in a spreadsheet program:

```
import java.io.*;
class Main {
  public static void main(String[] args) {
    try {
      FileWriter fw = new FileWriter(“names.txt”);
      fw.write(“First item,”);
      fw.write(“Second item\n”);
      fw.close();
    } catch (IOException e) {
      System.out.println(“Could not write to file”);
    }
  }
}
```

Note that the two examples above contain `try…catch` blocks of code. These are necessary in Java because file input or output can often cause runtime errors and Java needs to know what to do instead of crashing the program.

The following program will write the contents of an array of car brands into a file called `cars.txt` with each value separated by a comma:

```
import java.io.*;
class Main {
  public static void main(String[] args) {
    String[] cars = {“BMW”, “Toyota”, “Audi”, “Renault”,
“Rover”};
    try {
      FileWriter fw = new FileWriter(“cars.txt”);
      for(int index = 0; index < cars.length; index++) {
        fw.write(cars[index] + “,”);
      }
      fw.close();
    } catch (IOException e) {
      System.out.println(“Could not write to file”);
    }
  }
}
```

The file can be read back into an array in the following way:

```
import java.io.*;
import java.util.*;
class Main {
  public static void main(String[] args) {
    String[] cars;
    try {
      File file = new File("cars.txt");
      Scanner scanner = new Scanner(file);
      String line = scanner.nextLine();

      // split each line into an array of strings
      cars = line.split(",");

      // display each car brand
      for(int index = 0; index < cars.length; index++) {
        System.out.println(cars[index]);
      }

      scanner.close();
    } catch (IOException e) {
      System.out.println("Could not read from file");
    }
  }
}
```

C#

In C#, you're likely to use the File class for reading and writing files. To use the File class you will need to add the line `Using System.IO;` at the top of your C# program, usually under the existing `Using System;` line. The File class provides over 50 different methods for working with files, so this section only provides an introduction to some of the basic methods.

WRITING TO A FILE

C#

The following program creates a file called `"names.txt"` and adds two items of data.

```
StreamWriter sw = new StreamWriter("names.txt");
sw.Write("First Item");
sw.Write("Second Item");
sw.Close();
```

The file it creates would contain: `First itemSecond item`. There is nothing to separate the two items of data. Therefore, it is useful to add a character such as a comma or semicolon between them. The file created would then contain: `First item,Second item`. It is important to separate the data items so that they can be read back into a data structure.

The following program would write the contents of an array into a text file called `cars.csv` with a comma between each item.

```
string[] makes = {“BMW”, “Toyota”, “Audi”, “Renault”, “Rover”};
StreamWriter sw = new StreamWriter(“cars.csv”);

foreach (string currentItem in makes)
{
    sw.Write(currentItem + “,”);
}

sw.Close();
```

The file can be read back into an array using:

```
string[] makes = File.ReadAllText(“z:cars.txt”).Split(‘,’);
```

ACTIVITY 24

A student has coded a computer game which stores the five highest scores in an array. She now wants to save those scores to a file and load them back in when the game is run again.

Write a program that will save the array of scores to a suitable text file and then load them back in again. Run your program to check that it is working as intended.

PYTHON

A two-dimensional array can be stored in a text file in a similar way.

A teacher stores the names of students and their test scores in a two-dimensional array.

```
Slist = [[‘Faruq’, 60], [‘Jalila’, 90]]
```

The names and scores can be saved in a text file:

```
Slist = [[‘Faruq’, 60], [‘Jalila’, 90]]
myFile = open(‘results.txt’, ‘w’)

for x in range(len(Slist)):
    myFile.write(Slist[x][0] + ‘,’)
    new = str(Slist[x][1])
    myFile.write(new + ‘,’)

myFile.close()
```

It is easier to copy the items from a text file into a two-dimensional array in two stages.

First, copy them into a one-dimensional array:

```
Blist = []
myFile = open(“results.txt”, “r”)
Blist = myFile.read().split(‘,’)
myFile.close()
```

Second, move them into a two-dimensional array:

```
newList = []
for index in range(0, len(Blist) - 1, 2):
    newLlist.append([Blist[index], Blist[index + 1]])
```

A step of 2 is used as each record in the two-dimensional array contains 2 items from the one-dimensional array.

JAVA

The following code writes the student test scores to a file:

```
import java.io.*;
import java.util.*;
class Main {
  public static void main(String[] args) {
    // store student scores in a 2d array
    String[][] studentScores = {
      {“Faruq”, “60”},
      {“Jalila”, “90”}
    };

    // write scores to a file
    try {
      FileWriter fw = new FileWriter(“results.txt”);
      for(int row = 0; row < studentScores.length; row++) {

        // write student’s name
        fw.write(studentScores[row][0] + “,”);

        // write student’s score
        fw.write(studentScores[row][1] + “\n”);
      }
      fw.close();
    } catch (IOException e) {
      System.out.println(“Could not write to file”);
    }
  }
}
```

The following code reads the test scores back into a two-dimensional array:

```
import java.io.*;
import java.util.*;
class Main {
  public static void main(String[] args) {
    // list of arrays of strings to store results
    ArrayList<String[]> results = new ArrayList<String[]>();
    // read scores from a file
    try {
      File file = new File("results.txt");
      Scanner scanner = new Scanner(file);

      // read each line from the file
      while(scanner.hasNextLine()) {
        String line = scanner.nextLine();

        // split line into name and score and add it to
           the list
        String[] singleStudent = line.split(",");
        results.add(singleStudent);
      }
      scanner.close();
    } catch (IOException e) {
      System.out.println("Could not read from file");
    }

    // display scores for each student
    for(int i = 0; i < results.size(); i++) {
      System.out.println(results.get(i)[0] + ": " +
results.get(i)[1]);
    }
  }
}
```

In the above example, a list is used instead of an array because a list is a dynamic data structure. This allows us to add new items as we read them from the file, which we couldn't do with an array.

C#

```
string[,] grades = new string[,] { { "Faruq", "60" }, {
"Jalila", "90" } };

StreamWriter sw = new StreamWriter("grades.csv");

for (int i = 0; i < grades.GetLength(0); i++)
{
   sw.WriteLine(grades[i, 0] + "," + grades[i,1]);
}

sw.Close();

}
```

Reading back in to two-dimenional array:

```
// read all of file into array
string[] allLines = System.IO.File.ReadAllLines("grades.
csv");
int length = allLines.Length;

string[,] grades;
grades = new string[length, 2];

// for each line in array containing all lines from the
file, split using delimiter and assign to 2d array
for (int i = 0; i < length; i++)
{
   string[] temp = allLines[i].Split(',');
   grades[i, 0] = temp[0];
   grades[i, 1] = temp[1];
}
```

ACTIVITY 25

In Activity 24, only the high scores were saved. Change your program so that high scores and the names of the players who attained them are stored.

CHECKPOINT

Strengthen

SKILLS REASONING

S1 What might happen if a program doesn't include validation on user input?

SKILLS CRITICAL THINKING

S2 Can you think of a data structure that is suitable for storing a list of values used in a look-up check?

SKILLS PROBLEM SOLVING

S3 Show how your name, date of birth and favourite colour would be stored in a text file.

SKILLS REASONING

S4 Why is it beneficial to write data to a text file?

Challenge

SKILLS PROBLEM SOLVING

C1 Develop a program that:

- writes a set of employee records consisting of employee number, name and department to a text file
- reads in the stored records from the text file
- allows the user to search for an employee's details.

How confident do you feel about your answers to these questions? If you're not sure you answered them well, try redoing the activities in this section.

SUMMARY

- Validation techniques should be used to ensure that data entered by a user or from a file is valid. They can't guarantee that the data is correct, only that it is reasonable.
- A range check is used to ensure that data is within a specified range.
- A length check is used to ensure that data has a length within a specified range.
- A presence check is used to ensure the user has entered some data.
- When users are required to choose from a list of options, their input should be validated to ensure that their choice is valid.
- Large sets of data are normally stored in text files. The advantage of writing data to a file is that the data is not lost when the program is terminated. It can be read in from the file whenever it is needed.

10 SUBPROGRAMS

When programs are being coded, they should be as structured as possible so that people reading them can quickly understand their logic. One way of doing this is to use self-contained modules, which can be reused where possible. This ensures that programs are shorter, as commands do not have to be repeated several times. These modules can then be called when they are needed.

LEARNING OBJECTIVES

- Describe what a subprogram is
- Explain the benefits of using subprograms
- Explain the difference between global and local variables
- Explain the concept of passing data into and out of subprograms
- Create subprograms that use parameters
- Write code that uses user-defined and pre-existing subprograms

A subprogram is a self-contained module of code that performs a specific task. For example, a high-level programming language could provide a predefined subprogram that calculates the average of a set of numbers. A programmer can use this subprogram in any program they write without needing to know how it works. In other words, subprograms support the process of abstraction.

FUNCTIONS

EXAMPLE FUNCTIONS

A function returns a value to the main program that called it.

```
FUNCTION dice ( )                # This is the start of the function.
    simDie = RANDOM(6)           # This statement uses the
                                 predefined function RANDOM
                                 to generate a random number
                                 between 1 and 6 which is stored
                                 in the variable simDie.
RETURN simDie                    # This returns the value of the
                                 variable simDie to the main
                                 program.
END FUNCTION                     # This ends the definition of the
                                 function.
```

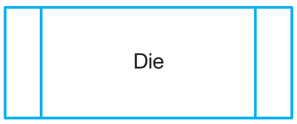
This function is called by the main program like this.

```
dieThrow = dice()
```

The value returned by the function is stored in the variable `dieThrow`.

DID YOU KNOW?

In a flowchart, a subprogram is represented by this symbol.



▲ **Figure 2.3** The symbol for a subprogram in a flowchart

In high-level programming languages, this would be written as:

PYTHON

```
def dice()
    simDie = random.randint(1, 7)
    return simDie
```

It would be called from the main program by:

```
dieThrow = dice()
```

JAVA

```
import java.util.*;
class Main {
  // define the function
  public static int dice() {
    Random r = new Random();
    return 1 + r.nextInt(6);
  }

  public static void main(String[] args) {
    // call the function
    int dieThrow = dice();
    System.out.println(dieThrow);
  }
}
```

C#

As C# is an **OOP language** it has methods rather than functions. There are a number of ways of implementing methods, but it could be written like this:

```
public static int dice()
{
  Random random = new Random();
  return random.Next(1, 7);
}
```

It would be called from the main program by: `int value = dice();.`

SUBJECT VOCABULARY

OOP language an object-oriented programming language. Instead of data structures and separate program structures, both data and program elements are combined into one structure called an object

LOCAL AND GLOBAL VARIABLES

SUBJECT VOCABULARY

local variable a variable that is accessed only from within the subprogram in which it is created

global variable a variable that can be accessed from anywhere in the program, including inside subprograms

Notice that there are two variables that store the random number generated by the function. In the function itself, the variable `simDie` is used. This variable only exists within the function and is referred to as a **local variable**.

In the main program the value returned by the function is stored in the variable `dieThrow`. It can be used anywhere within the main program and is therefore referred to as a **global variable**.

PROCEDURES

Unlike a function, a procedure does not return a value to the main program.

EXAMPLE PROCEDURE

In the dice example, a procedure would be written in pseudocode as:

```
PROCEDURE averageScore(score1, score2, score3)
BEGIN PROCEDURE
    SET total TO score1 + score2 + score3
    SET average TO total / 3
    SEND average TO DISPLAY
END PROCEDURE
```

Python
In Python, it would be exactly the same without the return command.

```
PROCEDURE dice()
    simDie = RANDOM(6)
    SEND simDIE TO DISPLAY
END PROCEDURE
```

Java
Because a procedure doesn't return a value, Java uses the void keyword to indicate that a subprogram doesn't return a value.

```
import java.util.*;
class Main {
  // define the procedure
  public static void dice() {
    Random r = new Random();
    System.out.println(1 + r.nextInt(6));
  }

  public static void main(String[] args) {
    // call the procedure
    dice();
  }
}
```

C#
In C#, the equivalent to a procedure is a method that doesn't return a value. For example:

```
public static void dice()
{
  Random random = new Random();
  int value = random.Next(1, 7);
  Console.WriteLine(value);
}
```

ARGUMENTS AND PARAMETERS

SUBJECT VOCABULARY

parameter the names of the variables that are used in the subroutine to store the data passed from the main program as arguments

Data for the functions and procedures to work on can be passed from the main program as arguments. The function accepts them as **parameters**.

PYTHON

```
# Function rectangle
def rectangle(length, width):
    area = length * width
    return area

#Main program
rectangleLength = int(input('Please enter the length of
the rectangle'))
rectangleWidth = int(input('Please enter the width of the
rectangle'))
rectangleArea  = rectangle(rectangleLength, rectangleWidth)
print(rectangleArea)
```

JAVA

```
import java.util.*;
class Main {
  // define the rectangle function
  public static int rectangle(int length, int width) {
    int area = length * width;
    return area;
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Please enter the length of the
rectangle: ");
    int rectangleLength = scanner.nextInt();

   System.out.print("Please enter the width of the rectangle: ");
    int rectangleWidth = scanner.nextInt();
    scanner.close();

    int rectangleArea = rectangle(rectangleLength,
rectangleWidth);
    System.out.println(rectangleArea);
  }
}
```

C#

```
public static void Main()
{
  string lengthString, widthString;
  int length, width, rectangleArea;
```

```
    Console.WriteLine("Please enter the length of the rectangle");
    lengthString = Console.ReadLine();
    length = int.Parse(lengthString);

    Console.WriteLine("Please enter the width of the rectangle");
    widthString = Console.ReadLine();
    width = int.Parse(widthString);

    rectangleArea = rectangle(length, width);
    Console.WriteLine(rectangleArea);
}

public static int rectangle(int length, int width)
{
    int area;
    area = length * width;
    return area;
}
```

In this example, two data items are passed to the function – `rectangleLength` and `rectangleWidth`. These are called arguments.

The function receives them as parameters called length and width when the function is declared.

In the function, a variable is declared – area. This is called a local variable and its **scope** is within the function. If you tried to use it in the main program you would get an error message.

SUBJECT VOCABULARY

scope the region of code within which a variable is visible

Lots of arguments can be passed to the function and many values can be returned.

In the following example, two values are requested and returned.

PYTHON

```
# Function rectangle
def rectangle(length, width):
    area = length * width
    circumference = (2 * length) + (2 * width)
    return area, circumference

#Main program
rectangleLength = int(input('Please enter the length of 
the rectangle'))
rectangleWidth = int(input('Please enter the width of the 
rectangle'))
rectangleArea, rectangleCircumference  = rectangle
(rectangleLength, rectangleWidth)
print(rectangleArea)
print(rectangleCircumference)
```

JAVA
In Java, functions can only return one value so if you need to return more than one value, you need to put them in a list or array.

```
import java.util.*;
class Main {
  // returns an array of the area and perimeter of a rectangle
  public static int[] rectangle(int length, int width) {
    int area = length * width;
    int perimeter = 2 * (length + width);
    return new int[]{area, perimeter};
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Please enter the length of the 
rectangle: ");
    int rectangleLength = scanner.nextInt();

    System.out.print("Please enter the width of the 
rectangle: ");
    int rectangleWidth = scanner.nextInt();
    scanner.close();

    int[] results =  rectangle(rectangleLength, 
rectangleWidth);
    int rectangleArea =results[0];
    int rectanglePerimeter = results[1];
    System.out.println("Area: " + rectangleArea);
    System.out.println("Perimeter: " + rectanglePerimeter);
  }
}
```

C#
Methods in C# can't return multiple values unless you use `By Reference`, which is beyond the scope of this book.

ACTIVITY 26

From the above example, list:

a the global variables
b the local variables
c the arguments
d the parameters.

SUBPROGRAMS AND MENUS

Functions and procedures are useful when using menus in a program. When a user selects a menu option, they can be sent to a particular function or procedure.

WORKED EXAMPLE

For a system login and password control, the main program could have a menu system like this:

1 Register as a new user
2 Login
3 Change your password
4 Exit.

Now that we are using structured programming using procedures, it is relatively easy to direct a user to the part of the program they need.

The user enters a number between 1 and 4 and is directed to the correct section:

Python

```
print (“1. Register as a new user”)
print (“2. Login.”)
print (“3. Change your password.”)
print (“4. Exit.”)

choice = int(input(‘Please select a menu option.’))
if choice == 1:
    newUser()
elif choice == 2:
    login()
elif choice == 3:
    changePassword()
elif choice == 4:
    exit()
else:
    print(‘Incorrect option. Try again.’)
```

Java

```
import java.util.*;
class Main {
  // procedures can be implemented later
  // just shown here to illustrate the structure
  public static void newUser() {}
  public static void login() {}
  public static void changePassword() {}
  public static void exit() {}
  public static void main(String[] args) {
    // display main menu
    System.out.println(“1. Register as a new user”);
    System.out.println("2. Login");
    System.out.println("3. Change your password");
    System.out.println("4. Exit");
    // get user input
    Scanner scanner = new Scanner(System.in);
    int choice = scanner.nextInt();

    switch(choice) {
      // Register as a new user
      case 1:
        newUser();
      break;

      // Login
      case 2:
        login();
      break;

      // change password
      case 3:
        changePassword();
      break;

      // exit
      case 4:
        exit();
      break;

      // anything else
      default:
        System.out.println("Incorrect option, try 
again");
      break;
    }
    scanner.close();
  }
}
```

C#

```
string choiceString;
int choice;

Console.WriteLine("1. Register as a new user");
Console.WriteLine("2. Login");
Console.WriteLine("3. Change your password");
Console.WriteLine("4. Exit");
Console.WriteLine("Please select a menu option");
choiceString = Console.ReadLine();
choice = int.Parse(choiceString);
if (choice == 1)
{
  newUser();
}
else if (choice == 2)
{
  login();
}
else if (choice == 3)
{
  changePassword();
}
else if (choice == 4)
{
  exit();
}
else
{
  Console.WriteLine("Incorrect option. Try again");
}
```

ACTIVITY 27

a Describe the purpose of the program in the worked example above and explain how it functions.

b If the user inputs an incorrect option, they receive an error message and then the program terminates. Edit the program so that the program will run until a suitable option is input.

THE BENEFITS OF USING SUBPROGRAMS

Repeated sections of code need only be written once and called when necessary. This shortens the development time of a program and means that the finished program will occupy less memory space when it is run.

Subprograms also improve the structure of the code, making it easier to read through and follow what is happening. It's less complicated to check your code and debug your program if you use subprograms because each subprogram can be coded, inspected and tested independently. If changes have to be made at a later date it is easier to change a small module than having to work through the whole program.

In large development teams different members can be working independently on different subprograms. They can use and develop standard libraries of subroutines that can be reused in other programs.

BUILT-IN FUNCTIONS

SUBJECT VOCABULARY

built-in functions functions that are provided in most high-level programming languages to perform common tasks

In addition to user-written subprograms, most high-level programming languages have a set of **built-in functions** for common tasks. These are designed to save the programmer time, such as functions that print, count the number of characters in a string and generate random numbers.

CHECKPOINT

Strengthen

SKILLS CRITICAL THINKING

S1 What are the benefits of using subprograms?

SKILLS REASONING

S2 What is meant by the scope of a variable? Use your own examples.

SKILLS REASONING

S3 What happens when a global variable and a local variable share the same name?

SKILLS CRITICAL THINKING

S4 Provide an example of a common built-in function.

Challenge

SKILLS PROBLEM SOLVING

C1 Create and implement a calculator program that:

- allows the user to enter a set of numbers
- uses separate functions to calculate the mean, mode and median
- allows the user to select which function they want
- uses appropriate validation.

How confident do you feel about your answers to these questions? If you're not sure you answered them well, try redoing the activities in this section.

SUMMARY

- A subprogram is a section of code within a larger piece of code that performs a specific task. It can be used at any point in the program.
- A function is a subprogram that returns a value to the main program.
- A procedure is a subprogram that does not return a value to the main program.
- Parameters are values that are passed to a subprogram when it is called.
- Local variables can only be accessed from within the subprogram in which they are created.
- Global variables can be accessed anywhere in the program, including inside subprograms.
- Built-in functions are functions provided in a high-level programming language to perform common tasks.