OBJECTIVES

At the end of this chapter, you should be able to:

- identify and distinguish between high-level and low-level programming languages
- explain the difference between programming languages from first to fifth generation
- list the steps to create a program using the correct terminology
- list techniques to test and debug a program
- explain the purpose of documentation.

Programming languages

Computer programmers write programs. But a set of instructions that we would understand is very different from instructions that a computer would understand. To write a program, you have to use a specific programming language. Programming languages range from the very basic binary system (Chapter 4) to complex languages with rules and syntax.

Low-level and high-level languages

Programming languages are divided into two categories: low-level and high-level. Low-level programming languages have a very basic form and are often quite difficult for programmers to use. Different brands of CPU understand different kinds of low-level language, so this type of language is specific to the machine. High-level languages are made for programmers to read and use easily. Computers do not understand them, so compilers and translators (Chapter 5) are used to translate high-level code into machine code. High-level languages can be used on any machine.

Generations of programming languages

Programming languages have evolved over the years and have reached their fifth generation.

First generation

First-generation languages are low-level languages – machine language or machine code. They are programming languages in their purest form. Machine language can be understood by a computer, so it does not need to be translated. Binary is used to program in machine language. Machine language executes very quickly, but it is difficult and time-consuming to write programs in machine language. Here is an example of code written in binary:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Add</td>
</tr>
<tr>
<td>SUB</td>
<td>Subtract</td>
</tr>
<tr>
<td>STO</td>
<td>Store</td>
</tr>
<tr>
<td>NOP</td>
<td>No operation; stop</td>
</tr>
</tbody>
</table>

Recall how Chapter 4 used 1s and 0s to solve simple arithmetic problems. Now imagine trying to use this system to write complex programs. It would be difficult to write them and it would be easy to make mistakes. That is why later generations of programming languages evolved – to make it easier for programmers to write sophisticated software.

Second generation

Second-generation languages are low-level languages and are often called assembly language. They use short pieces of code instead of binary (Table 9.1). Each code looks and sounds like the value or action it represents, but programmers have to learn and remember them. It is slightly easier to write programs in assembly language than in machine code. An assembler is used to translate assembly language into machine code.

This command instructs the computer to add the value 20 to accumulator B:

ADD B, 20
Third generation

Third-generation languages are high-level languages. Instead of using machine-specific code, they use statements in English that are easy for programmers to use. Here are some common third-generation languages:

- Pascal
- COBOL (Common Business-Oriented Language)
- FORTRAN (Formula Translation)
- BASIC (Beginner's All-Purpose Symbolic Instruction Code)
- C

And here is a Pascal program that adds two values:

```pascal
PROGRAM Addition;
VAR
  numA, numB, numC : integer;
BEGIN
  read(numA);
  read(numB);
  numC := numA + numB;
  write(numC);
END.
```

The commands are English words that are fairly simple to write and understand. A program written in a third-generation language is easy to follow, because it is structured with statements and keywords. The BEGIN and END keywords tell you where the processing begins and ends.

Fourth generation

Fourth-generation languages are very similar to third-generation languages, except they are even easier to use. They use English-like statements and offer extra help through prompts, instructions and wizards. Wizards are program assistants that guide the user. This means that programs can be written very quickly and easily and may not require the expertise of a programmer. Fourth-generation languages are often used in business because it is fairly quick and easy to train people who are not computer experts on how to use them. Here are some fourth-generation languages:

- Structured Query Language (SQL)
- Visual Basic
- C++
- Oracle Report

SQL is used for managing databases. The following SQL commands tell the computer to search a collection of data called `customer_information` for data called `cust_name` and select the data if purchases are over 100: