INTRODUCTION

Programming can be fun. If you've ever used a computer and wondered how it worked, or better yet, wanted to make your computer do something better, or different—something that it couldn't do before—then you already have the mind of a programmer.

The only requirement to be a programmer is the desire to do it. You don't need a college degree, an aptitude for mathematics, or special schooling. All you need is a computer and the curiosity to learn programming.

People may want to learn how to program their computer for any number of reasons. Some people want to write computer programs for a career. Others want to create a new product that improves upon an existing program or even forges a new market altogether. Still others want to learn programming for the sheer mental exhilaration of doing so, in much the same way that people enjoy crossword puzzles and brain teasers. Whatever your reasons for learning to program, you'll find that plenty of people just like you have learned to program a computer. That means you can do so too.

A Programming Primer

A computer program is like a recipe that gives the computer step-by-step instructions to follow. Just as you can write a recipe in English, Spanish, Chinese, Arabic, or Swahili, so too can you write a computer program using any variety of programming languages you choose. Contrary to popular belief, there is no inherently "best" programming language just as there is no single "best" human language.

Like human languages, the best programming language depends entirely on circumstances. Just as you would probably want to learn Japanese if you wanted to talk to people in Tokyo, so might you want to choose one programming language over another based on what you want to do. Some programming languages are designed to be easy to learn, others are designed for calculating mathematical formulas, and still others are designed for manipulating text. If you write the same program in two different languages, you'll find that one language will be easier to use than another, although no language will be best for writing all types of programs every time.

When learning to program a computer, you actually need to learn three related, but different, tasks:

How to solve a particular problem. If you want to write a program to predict the winners of a horse race, you need to know what data you need and what steps to follow to predict winning race horses. If you want to write a video game, you need to know the goal of the game, the rules to restrict the player's actions, and what obstacles will get in the player's way. If you don't know what problem you want to solve and how to solve it, the computer will never know how to solve it either.

How to use a particular programming language. Once you know what problem you want to solve, you need to choose a specific programming language to use. That means learning what commands are available in your chosen programming language and how to type those commands to tell the computer what to do step by step. In this book, you'll be learning the Visual Basic programming language.

How to use a specific program to help you write programs in your chosen programming language. To type program commands, you need to use an editor, which acts like a simple word processor that lets you write, edit, and save your programming instructions in a file. When you're done using the editor to type your program commands into a file, you need another program called a compiler, which turns your programming commands into a special file that can run by itself on other computers. To write programs in the Visual Basic language, you need to use Microsoft Visual Basic 2005 Express (hereafter referred to as Visual Basic Express), which is known as an *Integrated Development Environment (IDE)* since it includes both an editor and a compiler.

But before you start learning anything about Visual Basic Express or the Visual Basic programming language, take some time to learn the general principles behind all computer programming. Once you understand how computer programming works, you can better understand how to write a program using any programming language, including Visual Basic.

The Principles of Writing a Computer Program

A computer program is nothing more than a long list of instructions that tells the computer what to do, what to display on the screen, and what to do with data it receives from the user. The more complicated your program, the more instructions you'll need

to write in order to make your program work. A simple program, one that displays your name on the screen, might only require a handful of instructions, while a more complicated program, such as Microsoft Windows, could take several million instructions.

The key to programming is to write the fewest number of instructions that do the maximum amount of work as possible. The more instructions you write, the longer it will take and the more likely you may make a mistake. So rather than attempt to write one huge, monolithic program at one sitting, programmers divide a large program into several smaller ones. Once they write and test each small program, they put them together, like building blocks, to create a much larger program.

When you break a large program into several smaller ones, you can either store them in a single file on your hard disk, or store them in separate files. The advantage of storing everything in a single file is that you only have to look for one file when you want to edit or modify a program. The disadvantage of storing everything in a single file is that the longer your program gets, the harder it can be to find anything, much like trying to read a novel printed on a single scroll of paper.

For that reason, most programmers not only divide a large program into several smaller ones, but they also store the smaller programs in separate files. Separate files provide several advantages over storing everything in a single file:

- Smaller files are easier to edit, read, and ultimately understand.
- Separate files let different programmers work on the same program.
- Separate files isolate you from other parts of your program, which can prevent you from messing up other parts of your program by mistake.

KEY POINTS TO REMEMBER

No matter what computer, operating system, or programming language you use, the general principles of programming remain the same:

- Break a large program into smaller parts
- Make each part of a program as short and simple as possible (to make it easier to write, understand, and modify later)
- Keep each part of your program separate and isolated as much as possible (to keep one part of your program from accidentally messing up another part of your program)

The Principles of Programming Languages

The only language that computers understand is something called *machine code*, which consists of nothing more than instructions written as a series of zeroes and ones, such as:

1110 0110 1100 0001

1010 0000 1001 0111

0101 1110 0011 0110

One problem with machine code is that it's extremely difficult to understand just what each command really tells the computer to do. Another problem is that one misplaced 0 or 1 can give the computer a completely different command than you intended, and trying to find your mistake buried in a page full of zeroes and ones can be nearly impossible.

For that reason, few people program in machine code. To simplify programming, computer scientists created another programming language called *assembly language*. Assembly language replaces multiple lines of cryptic machine code with a single line of assembly language commands that can look very different, but be as equally confusing:

```
title   Hello World
; This program displays "Hello, World!"

dosseg
.model small
.stack 100h

.data
say_hello db 'Hello, World!',0dh,0ah,'$'

.code
main  proc
      mov    ax,@data
      mov    ds,ax

      mov    ah,9
      mov    dx,offset say_hello
      int    21h

      mov    ax,4C00h
      int    21h
main  endp
end   main
```

Unfortunately, computers don't understand assembly language either; they only understand machine code. To convert assembly language programs into machine code, computer scientists have created special programs called *assemblers*, which convert assembly language into machine code. Basically, every programming language is designed to make it easy for humans to write programs without going through the tedium and nuisance of learning machine code.

While somewhat simpler to understand then machine code, assembly language is still too complicated for most people to understand. So to make programming easier, computer scientists have invented computer languages that are much easier for people to read, write, and understand. As a result, programs that accomplish the same task, but written in different languages, can look drastically different:

```
; LISP
(DEFUN HELLO-WORLD ()
                (PRINT (LIST 'HELLO 'WORLD)))
```

The above example uses a programming language called LISP, which is designed for experimenting in giving computers artificial intelligence. The example below shows an equivalent program written in the popular C++ programming language.

```
#include <iostream>
int main()
{
    std::cout << "Hello, world!\n";
}
```