COURSE TECHNOLOGY

SHELLY CASHMAN SERIES®

Introductory

MICROSOFT

Visual Basic 6

Introductory Concepts and Techniques

Shelly Cashman Repede Mick

SHELLY CASHMAN SERIES®

MICROSOFT

Visual Basic 6

Introductory Concepts and Techniques

Gary B. Shelly
Thomas J. Cashman
John F. Repede
Michael L. Mick

Contributing Author
Deborah L. Fansler

COURSE TECHNOLOGY
ONE MAIN STREET
CAMBRIDGE MA 02142

an International Thomson Publishing company I(T)P

SHELLY CASHMAN SERIES®

CAMBRIDGE • ALBANY • BONN • CINCINNATI • LONDON • MADRID • MELBOURNE
MEXICO CITY • NEW YORK • PARIS • SAN FRANCISCO • TOKYO • TORONTO • WASHINGTON

Printed in the United States of America

For more information, contact:

Course Technology
One Main Street
Cambridge, Massachusetts 02142, USA

International Thomson Editores
Saneca, 53
Colonia Polanco
11560 Mexico D.F. Mexico

ITP Europe
Berkshire House
168-173 High Holborn
London, WC1V 7AA, United Kingdom

ITP GmbH
Konigswinterer Strasse 418
53227 Bonn, Germany

ITP Australia
102 Dodds Street
South Melbourne
Victoria 3205 Australia

ITP Asia
60 Albert Street, #15-01
Albert Complex
Singapore 189969

ITP Nelson Canada
1120 Birchmount Road
Scarborough, Ontario
Canada M1K 5G4

ITP Japan
Hirakawa-cho Kyowa Building, 3F
2-2-1 Hirakawa-cho, Chiyoda-ku
Tokyo 102, Japan

ISBN 0-7895-4653-1

PHOTO CREDITS: *Project 1, pages VB 1.2-3* Wedding photo, wedding rings, Courtesy of PhotoDisc, Inc.; baby Courtesy of Digital Stock; doll on tricycle, robot toy, Courtesy of KPT Metatools; *Project 2, pages VB 2.2-3* Painted mountains, Courtesy of Digital Stock; computer, slides, Courtesy of KPT Metatools; male model, girl with hat, Courtesy of Fargo; happy couple, Courtesy of PhotoDisc, Inc.; *Project 3, pages VB 3.2-3* Chalkboard, circuit board, microscope, rat, Courtesy of PhotoDisc, Inc.

3 4 5 6 7 8 9 10 BC 03 02 01 00 99

Preface

The Shelly Cashman Series® offers the finest textbooks in computer education. In our Microsoft Visual Basic 6 textbooks, you will find an educationally sound and easy-to-follow pedagogy that combines a step-by-step approach with corresponding screens. An Introduction to Visual Basic Programming section at the beginning of the book emphasizes good programming practices and gives students the foundation to produce well-written Windows applications. The Other Ways and More About features offer in-depth knowledge of Visual Basic 6. The project openers provide a fascinating perspective on the subject covered in the project. The Shelly Cashman Series Microsoft Visual Basic 6 books will make your programming class exciting and dynamic and one that your students will remember as one of their better educational experiences.

Objectives of This Textbook

Microsoft Visual Basic 6: Introductory Concepts and Techniques is intended for a one-credit course that includes a survey of Visual Basic programming. No experience with a computer is assumed, and no mathematics beyond the high school freshman level is required. The objectives of this book are:

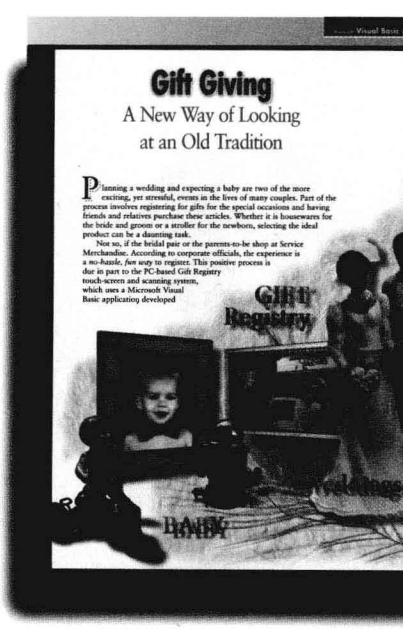
- To teach the basic concepts and methods of object-oriented programming
- To teach the fundamentals of the Microsoft Visual Basic 6 programming system
- To acquaint students with the three-step process of building Windows applications using Visual Basic 6
- To use practical problems to illustrate application-building techniques
- To take advantage of the many new capabilities of building applications in a graphical environment
- To encourage independent study and help those who are working alone in a distance education environment

When students complete the course using this textbook, they will have a basic knowledge and understanding of Visual Basic 6.

Obtaining a Copy of Microsoft Visual Basic 6 Working Model

Microsoft Visual Basic 6 Working Model can be bundled with this textbook for a minimal charge, so your students will have their own copy of Microsoft Visual Basic 6. Bundling the textbook and software is ideal if your school does not have Microsoft Visual Basic 6 or if you have students working at home on their own personal computers.

All the projects and exercises in this book can be completed using the Working Model. The only significant limitation of the Working Model is that it does not provide the MAKE.OLE DLLs command and the MAKE.EXE command.



Other Ways

1. On Project menu click Components
2. Press CTRL+T

The Shelly Cashman Approach

Features of the Shelly Cashman Series Microsoft Visual Basic 6 books include:

- **Project Orientation:** Each project in the book builds a complete application using the three-step process: creating the interface, setting properties, and writing code.
- **Screen-by-Screen, Step-by-Step Instructions:** Each of the tasks required to complete a project is identified throughout the development of the project. Steps to accomplish the task are specified, accompanied by screens.
- **Thoroughly Tested Projects:** Every screen in the book is correct because it is produced by the author only after performing a step, resulting in unprecedented quality.
- **Other Ways Boxes for Reference:** Visual Basic 6 provides a variety of ways to carry out a given task. The Other Ways boxes displayed at the end of many of the step-by-step sequences specify the other ways to do that task. Thus, the steps and Other Ways box make a comprehensive reference unit.
- **More About Feature:** These marginal annotations provide background information that complement the topics covered, adding interest and depth.

More About

Writing Code

Code in a Visual Basic application is divided into blocks called procedures. You write code for the Visual Basic application in a separate window, called the Code window. Using the Code window, you can view and edit any of the code quickly. You can choose to display all code procedures in the same Code window, or display a single procedure at a time.

Organization of This Textbook

Microsoft Visual Basic 6: Introductory Concepts and Techniques provides detailed instruction on how to use Visual Basic 6. The material is divided into an introductory section and three projects as follows:

Introduction to Visual Basic Programming This section provides an overview of application development, user interface design, program development methodology, structured programming, object-oriented programming, and the Visual Basic development system.

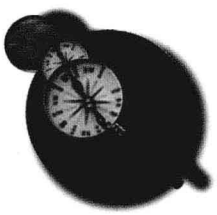
Project 1 - **Building an Application** Project 1 introduces students to the major elements of Visual Basic. Students develop Calculating Sales Commission, a sales commission conversion application. The process of building the application consists of three steps: creating the interface, setting properties, and writing code. Topics include starting Visual Basic; designing a form and adding labels, text boxes, and command buttons; changing the properties of controls; specifying an event procedure; using a function and a method in code; running and saving applications; documenting applications; starting a new project and opening an existing project; and accessing information about Visual Basic using Help.

Project 2 - **Working with Intrinsic Controls and ActiveX Controls** Project 2 presents additional properties of the controls used in Project 1, as well as several new intrinsic controls. ActiveX controls also are explained and used in the project to build the Theater Box Office application. Topics include copying controls; copying code between event procedures; using variables and constants in code statements; and using code statements to concatenate string data.

Project 3 - **Multiple Forms, Dialogs, Debugging, and EXEs** Project 3 extends the basics of building applications. The SavU Loan Analyzer application in this project consists of multiple forms and dialog boxes. Topics include additional properties of the controls presented in Projects 1 and 2; WindowState and modality; adding an icon to a form; using Image, Line, and ScrollBar controls; debugging applications using the features of Visual Basic's Debug window, and creating EXE files.

End-of-Project Student Activities

A notable strength of the Shelly Cashman Series Visual Basic 6 books is the extensive student activities at the end of each project. Well-structured student activities can make the difference between students merely participating in a class and students retaining the information they learn. The activities in the Shelly Cashman Series Visual Basic 6 books include:



- **What You Should Know** A listing of the tasks completed within a project together with the pages where the step-by-step, screen-by-screen explanations appear. This section provides a perfect study review for students.
- **Test Your Knowledge** Four pencil-and-paper activities designed to determine the students' understanding of the material in the project. Included are true/false questions, multiple-choice questions, and two short-answer activities.
- **Use Help** Any user of Visual Basic 6 must know how to use Help. Therefore, this book contains two Help exercises per project. These exercises alone distinguish the Shelly Cashman Series from any other set of Visual Basic 6 instructional materials.
- **Apply Your Knowledge** This exercise requires students to open and manipulate a file on the Data Disk that accompanies the Visual Basic 6 books.
- **In the Lab** Three in-depth assignments per project require students to apply the knowledge gained in the project to solve problems on a computer.
- **Cases and Places** Seven unique case studies require students to apply their knowledge to real-world situations.

Shelly Cashman Series Teaching Tools

A comprehensive set of Teaching Tools accompanies this book in the form of a CD-ROM. The CD-ROM includes an electronic Instructor's Manual and teaching and testing aids. The CD-ROM (ISBN 0-7895-4655-8) is available through your Course Technology representative or by calling one of the following telephone numbers: Colleges and Universities, 1-800-648-7450; High Schools, 1-800-824-5179; and Career Colleges, 1-800-477-3692. The contents of the CD-ROM follow.

- **Instructor's Manual** The Instructor's Manual is made up of Microsoft Word files. The files include lecture notes, solutions to laboratory assignments, and a large test bank. The files allow you to modify the lecture notes or generate quizzes and exams from the test bank using your own word processor. Where appropriate, solutions to laboratory assignments are embedded as icons in the files.
- **Figures in the Book** Illustrations for every figure in the textbook are available. Use this ancillary to create a slide show from the illustrations for lecture or to print transparencies for use in lecture with an overhead projector.
- **Course Test Manager** Course Test Manager is a powerful testing and assessment package that enables instructors to create and print tests from the large test bank. Instructors with access to a networked computer lab (LAN) can administer, grade, and track tests online. Students also can take online practice tests, which generate customized study guides that indicate where in the textbook students can find more information for each question.

- **Lecture Success System** Lecture Success System files are for use with the application software, a personal computer, and projection device to explain and illustrate the step-by-step, screen-by-screen development of a project in the textbook without entering large amounts of data.
- **Instructor's Lab Solutions** Solutions and required files for all the In the Lab assignments at the end of each project are available.
- **Student Files** All the files that are required by the student to complete the Apply Your Knowledge exercises are included.
- **Interactive Labs** Eighteen hands-on interactive labs that take students from ten to fifteen minutes each to step through help solidify and reinforce mouse and keyboard usage and computer concepts. Student assessment is available.

Student Data Disk

A few of the exercises in this textbook, especially the Apply Your Knowledge exercises, require that students begin by opening a file on the Student Data Disk. Students can obtain a copy of the Data Disk by following the instructions on the inside back cover of this textbook.

Acknowledgments

The Shelly Cashman Series would not be the leading computer education series without the contributions of outstanding publishing professionals. First, and foremost, among them is Becky Herrington, director of production and designer. She is the heart and soul of the Shelly Cashman Series, and it is only through her leadership, dedication, and tireless efforts that superior products are made possible. Becky created and produced the award-winning Windows 95 series of books.

Under Becky's direction, the following individuals made significant contributions to these books: Doug Cowley, production manager; Ginny Harvey, series specialist and developmental editor; Ken Russo, graphic designer and Web developer; Mike Bodnar, Stephanie Nance, Mark Norton, and Dave Bonnewitz, graphic artists; Jeanne Black, Quark expert; Marlo Mitchem, production/administrative assistant; Nancy Lamm, copyeditor/proofreader; Cristina Haley, indexer; Sarah Evertson of Image Quest, photo researcher; and Susan Sebok, contributing writer.

Special thanks go to Jim Quasney, our dedicated series editor; Lisa Strite, senior editor; Lora Wade, associate product manager; Tonia Grafakos and Meagan Walsh editorial assistants; and Kathryn Cronin, product marketing manager. Special mention must go to Becky Herrington for the outstanding book design; Mike Bodnar for the logo designs; and Stephanie Nance for the cover design and illustrations.

Gary B. Shelly
Thomas J. Cashman
John F. Repede
Michael L. Mick

MICROSOFT

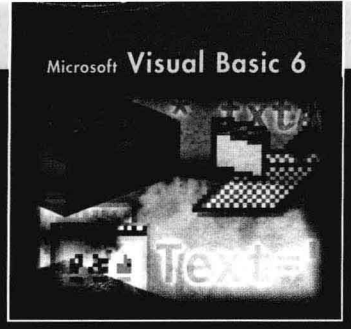
Visual Basic 6

Introductory Concepts and Techniques

C O N T E N T S

Microsoft

Visual Basic 6



Microsoft Visual Basic 6

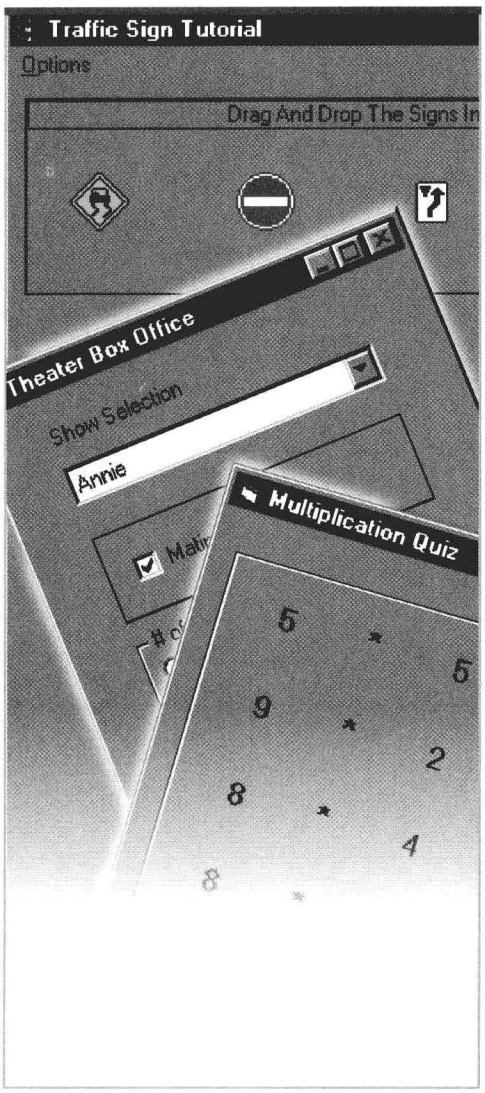
Introduction to Visual Basic Programming

OBJECTIVES

You will have mastered the material in this project when you can:

- Describe programs, programming, applications, and application development
- List six principles of user interface design
- Describe each of the steps in the program development life cycle
- Define structured programming
- Read and understand a flowchart
- Read and understand a HIPO chart
- Explain sequence, selection, and repetition control structures
- Describe object-oriented programming
- Define the terms objects, properties, methods, and events
- Read and understand a generalization hierarchy
- Read and understand an object structure diagram
- Read and understand an event diagram
- Define and explain encapsulation, inheritance, and polymorphism
- Describe rapid application development (RAD) and prototyping
- Describe VBA, VBScript, and the Visual Basic language

Introduction to Visual Basic Programming



Introduction

The **Visual Basic Programming System** encompasses a set of tools and technologies that are being used by more than three million developers worldwide to create computer software components and applications. At the release of an earlier version of Visual Basic, Bill Gates, chairman and CEO of Microsoft Corporation, said, "It has been a long time in coming, but the industrial revolution of software is finally upon us." Although not all people would agree with Mr. Gates's characterization of Visual Basic, most would agree that it is an extremely versatile, powerful, and yes — complex development system (the Professional Edition of version 6 contains more than 5,000 pages of user documentation). The popularity and complexity of Visual Basic is evidenced by the several Internet discussion groups dedicated to Visual Basic as well as the many World Wide Web sites and the wealth of books written about Visual Basic.

Schools across the country are teaching Visual Basic in a variety of different settings. These range from using Visual Basic as a general introduction to computer programming to two- and three-semester course sequences with prerequisite courses in subjects such as logic, software design, and structured programming. The editions of this book are designed to provide an introductory to intermediate working knowledge of Visual Basic and the software development concepts and technologies upon which it is built.

You may come to this course already having studied subjects such as systems analysis and design and structured programming, or this may be your first exposure to how a computer is programmed. The purpose of this introductory section is to provide the *big picture* of software design and development. You easily can find entire books and courses on each of the topics covered in this introduction, and your instructor may decide to spend more or less time on these topics depending on your background and the nature of the course you are taking.

Programming a Computer

Before a computer can start to produce a desired result, it must have a step-by-step description of the task to be accomplished. The step-by-step description is a series of precise instructions called a **program**. When these instructions are placed into the computer's memory, they are called a **stored program**. **Memory** stores the data and instructions that tell the computer what to do with the data.

Once the program is stored, the first instruction is located and sent to the control unit (**fetched**) where it is translated into a form the computer can understand (**decoded**), and then the instruction is carried out (**executed**). The result of the instruction is placed in memory (**stored**). Then, the next instruction is fetched, decoded, and executed. This process, called a **machine cycle**, continues under the direction of the operating system, instruction by instruction, until the program is completed or until the computer is instructed to halt.

In the Windows operating system, you can have multiple windows (programs) open on the desktop at the same time. Unless the computer has more than one CPU (each with its own memory), called **parallel processing**, it still executes only one instruction at a time. It carries these instructions out so fast, however, that it can move back and forth among programs carrying out some instructions from one program, and then some instructions from another program. To the user, it appears as though these programs are running simultaneously. This moving back and forth between programs by the processor is called **multitasking**. Even in a multitasking system, a processor can execute only one instruction at a time.

For the computer to perform another job, a new program is read into memory. By reading stored programs into memory, the computer can be used easily to process a large number of different jobs. The instructions within these jobs can be written, or **coded**, by a computer programmer in a variety of programming languages. Today, more than 2,000 programming languages exist. The process of writing the sets of instructions that make up these jobs is called **computer programming**.

Most computer users do not write their own programs. Programs required for common business and personal applications such as word processing or spreadsheets can be purchased from software vendors or stores that sell computer products. These purchased programs often are referred to as **application software packages**, or simply applications. **Applications** are programs that tell a computer how to accept instructions from the end user and how to produce information in response to those instructions. The process of using a programming language or development environment to build software applications is called **application development**.

User Interface Design

The way that a program accepts instructions from the user and presents results is called the **user interface**. Today, most applications have a graphical user interface (GUI). A **graphical user interface (GUI)** provides visual clues such as small pictures, or **icons**, to help the end user give instructions to the computer. Microsoft Windows is the most widely used graphical user interface for personal computers.

After working only a short time with different applications within the Windows operating system, you will begin to notice similarities in the user interfaces — even among applications created by companies other than Microsoft. This is not a coincidence and this similarity greatly reduces the time required to learn a new application.

Examine the packaging for software applications the next time you are in a store that sells software. Many applications have a Windows logo with the words, Designed for Microsoft Windows. These applications must follow an extensive set of requirements in order to display the Windows logo on the package. These include requirements ranging from supporting the use of long file names to adherence to very detailed Windows interface guidelines. Whether you want to display the Windows logo on your software or not, some basic principles need to be followed in designing a graphical user interface.

The application always should be under the user's control. The application should present choices that allow the user to initiate and control the application's events. One example would be a user's capability not only to initiate a print function, but if necessary, to cancel the print function before it is completed.

User capabilities and preferences vary greatly and change over time. The user should be able to customize the application to meet his or her preferences. For example, the **ToolTips** (pop-up help bubbles) that are valuable to a new user of an application may be annoying to a proficient user. A color scheme that one person finds attractive may be distracting to another. The application should allow, but not require, customizing. An initial default interface should be provided for those who do not want to specify customizing options.

Form should follow function. The interface should be designed to provide direct ways to accomplish tasks and not to be glamorous or trendy. Avoid the temptation to try to find some use within the application for your favorite visual elements. For example, a Drop-down ComboBox control should be used when the control you want to give the user is best accomplished with a combo box and not because you feel that combo boxes make your application more professional looking or fun to use. Some of the best interface elements are ones the user does not notice because their use is so intuitive, they do not have to consciously think, Now how do I do this? or What does this thing do?

Use concepts and metaphors that users are familiar with to make the interface parallel real-world experience. A familiar GUI metaphor for lifting and moving a real-world object from one location to another is to use a mouse to complete a drag-and-drop operation. Dragging and dropping an image of a file folder onto an image of a recycle bin is closer to real-world experiences than typing the characters `del filename.doc` next to the `c:\windows>` characters.

Applications should follow basic graphic design principles and should be consistent visually and functionally. For example, each time a dialog box is used to provide a message to the user, it should have a similar shape, location, color, font size, and style. Functional consistency means the user initiates the same set of events in the same way throughout the use of the application. If the same set of print options is offered to the user when printing report A or report B, then report A's options should not be presented as a menu and report B's options presented as a set of check boxes.

The user always should receive immediate feedback after initiating an event. You probably at one time or another have experienced the frustration of clicking a button or icon and then wondering whether anything was happening. When you click an icon and the mouse pointer changes to an hourglass or animated image, you understand that processing has been initiated in response to your click. If you have ever heard the Windows Chimes, Chord, Exclamation, or other sounds, you have experienced **auditory feedback**.

The application should attempt to prevent users from making mistakes as much as possible, rather than allowing mistakes and then pointing them out to the user. Nevertheless, users will make mistakes, and the application should be capable of responding to their mistakes. The user never should be told that he or she made a mistake. Instead, he or she should be told politely why a function cannot be carried out or prompted to reenter data. These messages often are conveyed in dialog boxes, called **error dialogs**, but they should never use language that implies the user is at fault. Many individuals learn how to use an application by trial and error and so the error dialog is a way to provide positive rather than negative feedback to the user.

The Program Development Life Cycle

When programmers build software applications, they just do not sit down and start writing code. Instead, they follow an organized plan, or **methodology**, that breaks the process into a series of tasks. There are many application development methodologies just as there are many programming languages. These different methodologies, however, tend to be variations of what is called the program development life cycle (PDLC).

The **program development life cycle** (**PDLC**) is an outline of each of the steps used to build software applications. Similarly to the way the system development life cycle (SDLC) guides the systems analyst through development of an information system, the program development life cycle is a tool used to guide computer programmers through the development of an application. The program development life cycle consists of six steps, summarized in Table 1.

Table 1

STEP	*PROCEDURE*	*DESCRIPTION*
1	Analyze the problem	Precisely define the problem to be solved, and write *program specifications* — descriptions of the program's inputs, processing, outputs, and user interface.
2	Design the program	Develop a detailed logic plan using a tool such as *pseudocode, flowcharts, object structure diagrams,* or *event diagrams* to group the program's activities into modules; devise a method of solution or algorithm for each module; and test the solution algorithms.
3	Code the program	Translate the design into an application using a programming language or application development tool by creating the user interface and writing code; include *internal documentation* — comments and remarks within the code that explain the purpose of code statements.
4	Test and debug the program	Test the program, finding and correcting errors (debugging) until it is error free and contains enough safeguards to ensure the desired results.
5	Formalize the solution	Review and, if necessary, revise internal documentation; formalize and complete end-user (external) documentation.
6	Maintain the program	Provide education and support to end users; correct any unanticipated errors that emerge and identify user-requested modifications (enhancements). Once errors or enhancements are identified, the program development life cycle begins again at Step 1.

Structured Programming

Structured programming, or **structured design,** is a methodology used to facilitate translating the problem analyzed in Step 1 of the PDLC into the specific program instructions in Step 3 (coding). Each module identified in the design step of the PDLC may be broken into a number of smaller and more precise instruction sets. This process can continue for several levels of even smaller, more precise instruction sets. The lowest-level instructions often are called **procedures.** A **hierarchy chart,** also called a **top-down chart** or **hierarchical input process output (HIPO) chart,** is a common way to represent this subdivision of activities visually. A hierarchy chart is shown in Figure 1.

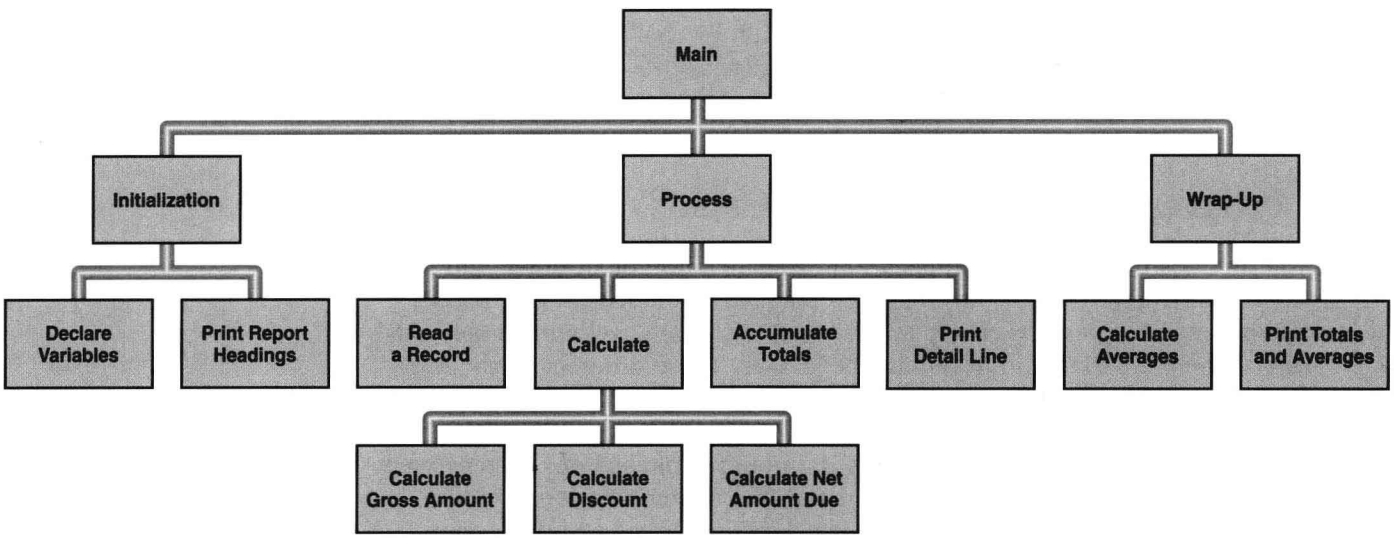


FIGURE 1

A **flowchart** is a design tool used to represent the logic in a solution algorithm graphically. Table 2 shows a standard set of symbols used to represent various operations in a program's logic.