Includes Ready-to-Run Software!

Computer Literacy T.M.

SURVIVAL KIT

For the APPLE II, IIe Family of Computers

Arthur Luehrmann and Herbert Peckham

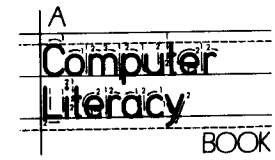
Computer Literacy™

SURVIVAL KIT

For the Apple II, IIe
Family of Computers

Arthur Luehrmann

Herbert Peckham



McGraw-Hill Book Company

New York St. Louis San Francisco Auckland Bogotá Guatemala Hamburg
Johannesburg Lisbon London Madrid Mexico Montreal New Delhi Panama
Paris San Juan São Paulo Singapore Sydney Tokyo Toronto

Editor: Jeff McCartney
Production Management: Suzanne LanFranchi
Photo Research: C. Buff Rosenthal, Alan Forman
Cover Design: Bob Mitchell

This book was set in 11 point Century Schoolbook by York Graphic Services, Inc.

ISBN 0-07-049206-9

Library of Congress Cataloging in Publication Data

Luehrmann, Arthur.
Computer literacy survival kit for the Apple II family of computers.

Includes index.
1. Apple II (Computer) I. Peckham, Herbert D. II. Title.
QA76.8.A662L84 1984 001.64 83-24900
ISBN 0-07-049206-9

1 2 3 4 5 6 7 8 9 10 DOCDOC 92 91 90 89 88 87 86 85 84

Acknowledgments

These materials result from the contributions of many people. An expert advisory board reviewed every word in our initial drafts. Some 600 people at 16 national test sites used the manuscript for four months. An evaluation team from Stanford University gathered data that led to major improvements. As the book took final form, we were further aided by an expert in computer awareness topics, and many other consultants. Without the help of these dedicated teams, we would have been unable to meet our goals for this important course of study.

ADVISORY BOARD We especially thank our three advisors, each of whom is a master computer educator with many years of experience. **Bobby Goodson, Helen Joseph,** and **Flora Russ** met with us in numerous all-day sessions and shaped our initial drafts into practical, teachable materials, ready for field testing.

COMPUTER-AWARENESS CONSULTANT We were also fortunate in having the help of **Dr. Ronald E. Anderson** of the University of Minnesota. In addition to being an old friend from the early days of educational computing, he is a national expert on the personal, societal, and ethical issues that should be studied in any computer literacy course.

TEST SITE STAFF The people who helped us at the field test sites are too numerous to name individually. We thank them all for giving the authors and evaluation team practical information on use and effectiveness, leading to major changes.

EVALUATION TEAM The national field test was planned and evaluated by **Meg Korpi, John Agnew, Steve Leitz, Stuart MacMillan,** and **Kathleen Gilbert-Macmillan.** They gathered and analyzed information, conducted visits to the local sites and held telephone interviews with remote locations. Their work was supervised by **Dr. Dennis Phillips,** Stanford University.

OTHERS We have particularly benefited from the work of two psychologists who have studied learning problems of people in computer courses: **Dr. Elliot Soloway,** Yale University, and **Dr. Richard Mayer,** University of California, Santa Barbara. To Professor Mayer we owe acknowledgment of his findings that people need to use concrete models to understand how computers work.

Automatic indentation and renumbering of BASIC statements encourages good writing habits. We are grateful to **Peter Rowe** for creating the WRITING TOOLS programs on the *Computer Literacy* diskette.

This entire project has needed good planning, coordination among teams, communication, and careful monitoring. For these management skills, we owe much to **Martha Ramirez,** President, Computer Literacy. She also contributed many helpful suggestions regarding content and form.

In the end, a book is words, photographs, ink, and paper. The unerring eye of **Antonio Padial** helped set our crooked words straight and make our rough meaning plain. To editors **Deborah Posner** and **Martha O'Neill,** and production director **Suzanne LanFranchi,** we owe the form and substance of this book. Finally, the entire project would not have seen the light of day without the foresight and continual support of **Robert Bowen,** Vice President, McGraw-Hill Book Company.

Arthur Luehrmann
Herbert Peckham

计算机基础知识 苹果机 ... DISK 0000
系列计算机的继续使用的应用程序

Computer Literacy™

SURVIVAL KIT

For the Apple II, IIe Family of Computers

Other McGraw-Hill titles by the authors

Luehrmann & Peckham APPLE PASCAL: A HANDS-ON APPROACH

Luehrmann & Peckham COMPUTER LITERACY FOR THE TRS-80 MODEL III AND MODEL 4 VERSION

Peckham BASIC: A HANDS-ON METHOD

Peckham with Ellis & Lodi HANDS-ON BASIC FOR THE APPLE II

Peckham with Ellis & Lodi HANDS-ON BASIC FOR THE ATARI 400/800/1200XL

Peckham HANDS-ON BASIC FOR THE IBM PERSONAL COMPUTER

Peckham with Ellis & Lodi HANDS-ON BASIC FOR THE TRS-80 COLOR COMPUTER

Peckham HANDS-ON BASIC WITH THE PET

Peckham PROGRAMMING BASIC WITH THE TI HOME COMPUTER

Other BYTE/McGraw-Hill computing titles

Buffington YOUR FIRST PERSONAL COMPUTER: HOW TO BUY AND USE IT

Hohenstein ALL ABOUT HAND-HELD COMPUTERS

Mullish & Kruger: APPLESOFT BASIC: FROM THE GROUND UP

Thomas: LEARN BASIC: A GUIDE TO PROGRAMMING THE TEXAS INSTRUMENTS COMPACT COMPUTER 40

Zboray & Sachs: PROGRAMS FOR PROFIT: HOW TO REALLY MAKE MONEY WITH A PERSONAL COMPUTER

About the Authors

Arthur Luehrmann and Herbert Peckham are pioneers in the field of computer education. They have worked extensively with educators at all levels to develop methods for teaching about computers. They are the authors of many books and articles on programming, the computer curriculum, and computers. Each was trained as a physicist and has taught physics. Together with Martha Ramírez, they formed the partnership Computer Literacy to develop educational materials for a national computer curriculum.

Arthur Luehrmann did his undergraduate and graduate work at the University of Chicago. While teaching at Dartmouth College in 1972, Dr. Luehrmann coined the phrase "computer literacy," and he has worked to make the concept of computer literacy an integral part of American education. Formerly, he was associate director of the Lawrence Hall of Science at the University of California, Berkeley. He now devotes full time to writing and speaking about computer literacy.

Herbert Peckham was graduated from the United States Military Academy. He received a Master's degree in physics from the Naval Postgraduate School, and did additional graduate work in physics at University of California, Berkeley. He has taught physics, computer science, and mathematics at Gavilan College. He is a widely published author of educational materials for use with computers.

Introduction

This book guides you as you explore the computer, discover its workings, and apply what you learn. If you are eager to get right into the computer activities that follow, there will be no harm in skipping most of the Introduction for now. **However, be sure to read the section "How to Use This Book" on page xi.**

The activities and reading you will do here are equivalent to a half-year introductory course in computing. However, you will quickly see that using a computer does not require special math skills. You don't have to be a genius. In fact, millions of junior-high students right now are learning in school the same things you will learn here. Anyone with curiosity and time can become computer literate.

What Is "Computer Literacy"?

The word "literacy" carries a great deal of weight. We all believe that literacy is the most important goal of education: Every citizen should be able to read and write. Democratic government, career opportunities, and personal welfare all depend on these skills. As a society, we also agree that "math literacy" is essential for all citizens. But what about "computer literacy"? Is there anything so important about computers that *every* citizen should learn it?

The "Age of Information" Many social critics have characterized our time as a transition from the "Industrial Age" to the "Age of Information." At the beginning of this century, most people in developed nations worked at blue-collar jobs in factories. By the end of this century, four-fifths of us will be information workers, not manual workers. Put simply, our jobs will be to receive information from others, to process information in various ways, and to pass information along to others.

The information machine Just as the steam engine magnified the manual worker's abilities by an enormous factor, so the computer can magnify an information worker's abilities. The computer is designed to do only three things: receive, process, and give back information—*the very things that all information workers do.* Therefore, a person's skills in using computer systems will largely determine his or her value in the work force in coming decades. Computer literacy is likely to have as big an impact on career opportunities in the future as ordinary literacy did in the past, when farm mechanization drove millions of workers into the cities in search of factory work.

Thinking skills There is more to computer literacy than access to jobs and careers, however. Like ordinary literacy and math literacy, computer literacy is a thinking skill. *Computer literacy means being able to tell a computer to do what you want it to.* To tell a computer to do anything, you must understand exactly what you want done. If the computer does not do what you wanted, you must think again about the instructions you gave. You may need to carry out experiments and revise your ideas. Notice that these are exactly the same kinds of things you do when writing a report or solving a math problem. Reading, writing, doing math, and working with a computer all require thought.

Why Programming?

To tell the computer to do what you want it to, you must have a language to do the "telling." Such languages are called "programming languages." Writing instructions for the computer is called "programming." You cannot use a computer without giving it instructions—that is, programming it. *In short, programming skills form the backbone of computer literacy.*

Many languages There are many thousands of languages for telling the computer to do what you want. Some languages can be used only for a single application, such as putting information into a data base and getting it out, writing letters or papers, or drawing graphs. Other languages are more general in their purpose and can be used for millions of different applications. (In fact these general-purpose languages are used to create special applications such as the three just mentioned.)

Special-purpose languages In many ways, using a special-purpose language for a single application is like using an airplane or bus for transportation. If the plane or bus is going where you are going, that is the best and simplest way to travel. Few people choose to drive from New York to San Francisco. Likewise, few people choose to write their own programs to tell the computer to do things many people want done. It is usually better to buy a standard program and learn the special-purpose language for that task.

General-purpose languages Using a general-purpose language is like driving your own car. Flying from New York to San Francisco makes good sense, but how do you get from the airport to your friend's house 30 miles away in Palo Alto? Either you drive there in a rented car, or someone drives to the airport and picks you up. Computer applications are like this too: Special-purpose programs handle common needs, but if you want the computer to do something different, you have to "drive it yourself." That means writing programs in a general-purpose language.

How the computer works There is another reason for learning a general-purpose programming language. It is only when you begin to write simple programs of your own that you will understand how a computer works, what its powers are, and what its limits are. *There is no better way to gain a general appreciation of computers than by learning to write programs for them.*

Why BASIC?

There are about a dozen commonly used general-purpose languages. You may have heard of Pascal, Logo, FORTRAN, or COBOL, for example. This book uses BASIC.

No "right" language Every computer language has loyal followers who say that it is better than the rest. Such partisan debate obscures an important truth: Most computer languages are far more alike than they are different from one another. The main ideas of programming can be learned in any of the common languages. Learning these ideas in one language makes it very easy to learn another language. People who work with computers often know two or three programming languages.

The common language of microcomputers Literacy of any kind takes practice. If you become computer literate in a rarely used language, you will have little opportunity to practice your new skills, and they will be forgotten quickly. This book uses BASIC, despite certain flaws, because BASIC is available on almost every computer you are likely to see. If you know how to express your ideas well in BASIC, you will usually be able to tell any computer to do what you want.

Structured BASIC As you will soon see, there is a lot more to writing programs than knowing the vocabulary, spelling, and grammar rules of a language. Just as in English, writing means using grammar and vocabulary to express an idea or feeling. In this book, you will learn how to plan, organize, and control your writing efforts, using methods called "top-down design" and "structured programming." Consistent use of these methods, in BASIC or in any other language, will make your programs easy to read, easy to get right, and easy to improve.

How to Use This Book

There are nine parts to this book. Each part begins with a one-page overview and ends with a review of the ideas covered. The parts are divided into several chapters, called "sessions". The even-numbered sessions are to be carried out at the computer; they are "hands-on" sessions. The odd-numbered sessions are to be read after one hands-on session is complete and before going on to the next one.

Hands-on sessions Actually, there are two kinds of hands-on sessions. In the first kind, you will *discover* the things a computer can do. This will happen as a result of experiments that you yourself carry out at the keyboard, guided by suggestions in the book. In the second kind, you will *apply* what you have learned by carrying out projects on the computer: drawing pictures, printing words, and creating games, for example.

Reading sessions There are two kinds of activities in the reading sessions also. You will *review* what you have experienced in the previous discovery session at the computer; you will learn new vocabulary terms, definitions, rules, and useful general ideas about computers and computing. After that, you may be asked to *plan* projects to be carried out during the next hands-on session.

Advice for hands-on discovery sessions These sessions cannot work unless you do the things suggested. It is best to set aside about 30 minutes at a time and complete the whole session in one sitting. Do not skip any activity in the session. Pay attention to the questions that appear just after each group of activities. Make mental notes of the answers, or jot them down on paper. If a question leaves you puzzled, look in Appendix 3 for the answer. At the end of most discovery sessions, you will find a few activities under the heading **On Your Own.** They are optional, but it is a good idea to do them. The goal of computer literacy is to be able to do things on one's own. These activities give you a chance to make certain that you can accomplish tasks without being given specific directions.

Advice for hands-on project sessions The hands-on sessions devoted to projects are also open-ended. Take as much time as needed to complete your project. If you are stumped, look in Appendix 2 for a sample program written for that project.

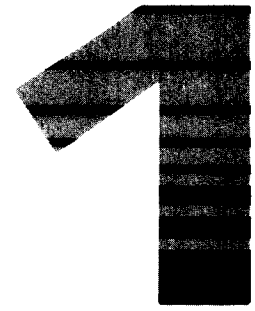
Advice for reading sessions Do *not* try to read the odd-numbered sessions while you are at the computer. It is better to leave the computer and find a comfortable, quiet place for reading and thinking. Allow a little "settling time" between one hands-on session and the next reading session. As you read, look for new vocabulary, rules, and explanations for the things you have seen at the computer. Use the questions at the end of the session as a self-test. (Answers are in Appendix 3.) If a project is suggested, do the planning on paper *before* going back to the computer.

Part reviews When you come to the end of one of the main parts of the book, spend a few minutes reading the review pages. Think about each new computer word and new idea introduced. Make certain that you understand it before going ahead to the next part. The terms will be used often in later parts.

Do experiments! Our final advice is this: When in doubt about how something works on the computer, do an experiment of your own to find

out. Like the world around us, a computer *is* what it *does,* and not necessarily what we or anyone else *says* about it. Nothing you type on the keyboard can harm the computer. So, when you have a question or an idea about the computer, explore it on your own at the keyboard. The things you learn this way will stick with you far better than the things you read in a book or are told.

Contents



Part 1

Taking Control

Computers are everywhere. There is one in every digital watch and every pocket calculator. New cars, television sets, microwave ovens, and typewriters often have computers. Without computers, there would be no video games and no electronic toys. At the office, computers help people type, file information, and send messages back and forth. In the factory, computers guide robot arms that build cars.

Slaves or masters? Computers are making revolutionary changes in the way we live, play, and work. Are these good changes? Some people worry that we may become slaves to this new machine, totally dependent on it and no longer in control of our own lives. Others see a brighter future in which people are masters and machines are helpers. Which future will actually happen?

Computer literacy The answer to that question depends on what you decide to do about the computer. You can decide that computers are too difficult and leave the whole subject in the hands of the "experts." Or you can decide to learn about computers and take control of them yourself. In other words, you can become **computer literate.**

Literacy and freedom It took you many years to master the skills of reading and writing, but these skills gave you *freedom.* You do not have to depend on others to read street signs for you, or to tell you the latest news, or to write your letters to friends or representatives in Congress. You do not have to trust experts: You are *literate.* In the same way, your "math literacy" gave you freedom: You do not have to trust a math expert to tell you whether three oranges for 80 cents is a better buy than one orange for 25 cents. You are in control.

Controlling the computer Computer literacy is like these other skills: It takes time and practice to learn, but it puts you in control of the computer and frees you from having to depend on and trust a computer expert. When you become computer literate, you will know two important things: (1) what things a computer *can* do and (2) how to tell a computer to do the things *you* want it to do.

1 The Computer in Your Life

IN THIS SESSION YOU WILL:

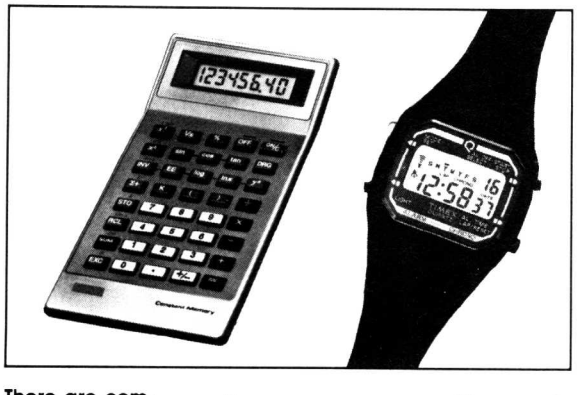
- Learn some ways people use computers.
- Learn how computers are important in everyday life.
- Learn the difference between dedicated computers and general-purpose computers.
- Learn that controlling computers can be useful and fun.

Computers and People

January of 1980 is an important month in the history of computers and people. During that month, more computers were built than children were born.

A computer for everyone? For the first time, it began to seem possible that any person who wanted a computer could own one. In fact, only three years later there were as many computers in the world as people. Where are all those computers?

Computers in calculators Have you used a pocket calculator? There is a computer inside a pocket calculator. It notices each key that you press, knows all the rules of arithmetic, and writes answers on the display. The calculator can do arithmetic because a computer tells it how.



There are computers in this calculator and in this digital watch.

Computers in watches There is also a computer inside each digital watch—the watches without hands. The watch contains a tiny quartz crystal that vibrates many times every second. The computer counts each beat of the crystal, adds up all the beats, and writes the time on the display. The watch shows the time because a computer tells it how.

Computers in typewriters Some typewriters let you type whole letters or compositions into a part called the "memory." Later you can call up what you have written, correct spelling errors, and rewrite as much as you want. After that, you can have the typewriter type a perfect copy automatically. This typewriter remembers because a computer inside tells it how.

Computers in stereo and TV sets To pick a station on older stereo and TV sets, you turn a knob that moves some metal parts inside the

set. To pick a station on newer units, you just touch a small button. A computer notices which button you touch, makes electrical changes inside the unit, and writes the new channel number or station frequency on a small digital display. The computer may also remember your favorite stations and how loud you like the sound.

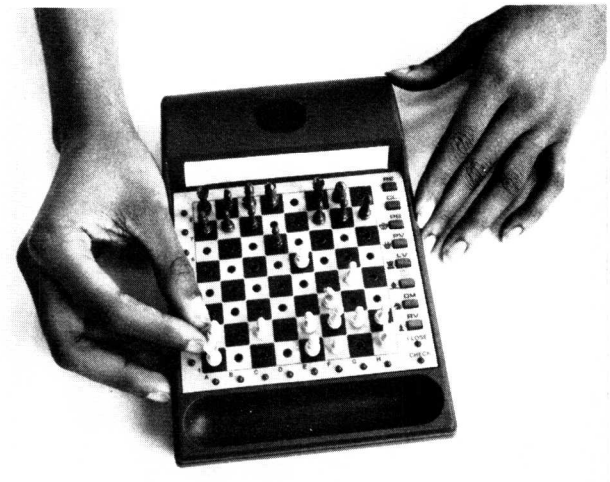
Computers in games and toys Have you used a video game or an electronic toy? An electronic chess game? Usually, a computer tells these games what to do and how to do it.

Same parts in all It might surprise you that there are computers inside all these household appliances. Most people probably think that a pocket calculator and an electronic game must look very different inside. In fact, the parts of one are much like the parts of the other. The main difference is that one computer "knows" the rules of arithmetic and the other computer "knows" the rules of chess.

Who Tells the Computer?

The fact that a computer can store knowledge is very important. This ability makes computers different from every other machine ever invented.

The stored program Before a computer can play chess, it must be told the rules and tactics of the game. This information is contained in a **computer program**, or simply, a **program**. *A program is a list of instructions that the computer carries out one at a time.* With a different program, the same computer that carried out the steps of a chess game can carry out the steps of a multiplication problem.



This chess game has a dedicated computer.

Writing programs The knowledge inside any computer was put there by a person. Before a computer can play chess, a good chess player has to write the rules and tactics into a program and then put the program into the computer. After that, the computer can play chess, but only as well as the person who wrote the program. *Whenever you see a computer in use, remember that it is doing only what some person told it to do.*

Dedicated computers You cannot easily change the program in a chess game, a typewriter, a TV set, a watch, or a calculator. Each program is a permanent part of the appliance. We say that each of these computers is **dedicated** to carrying out the steps of one program. It