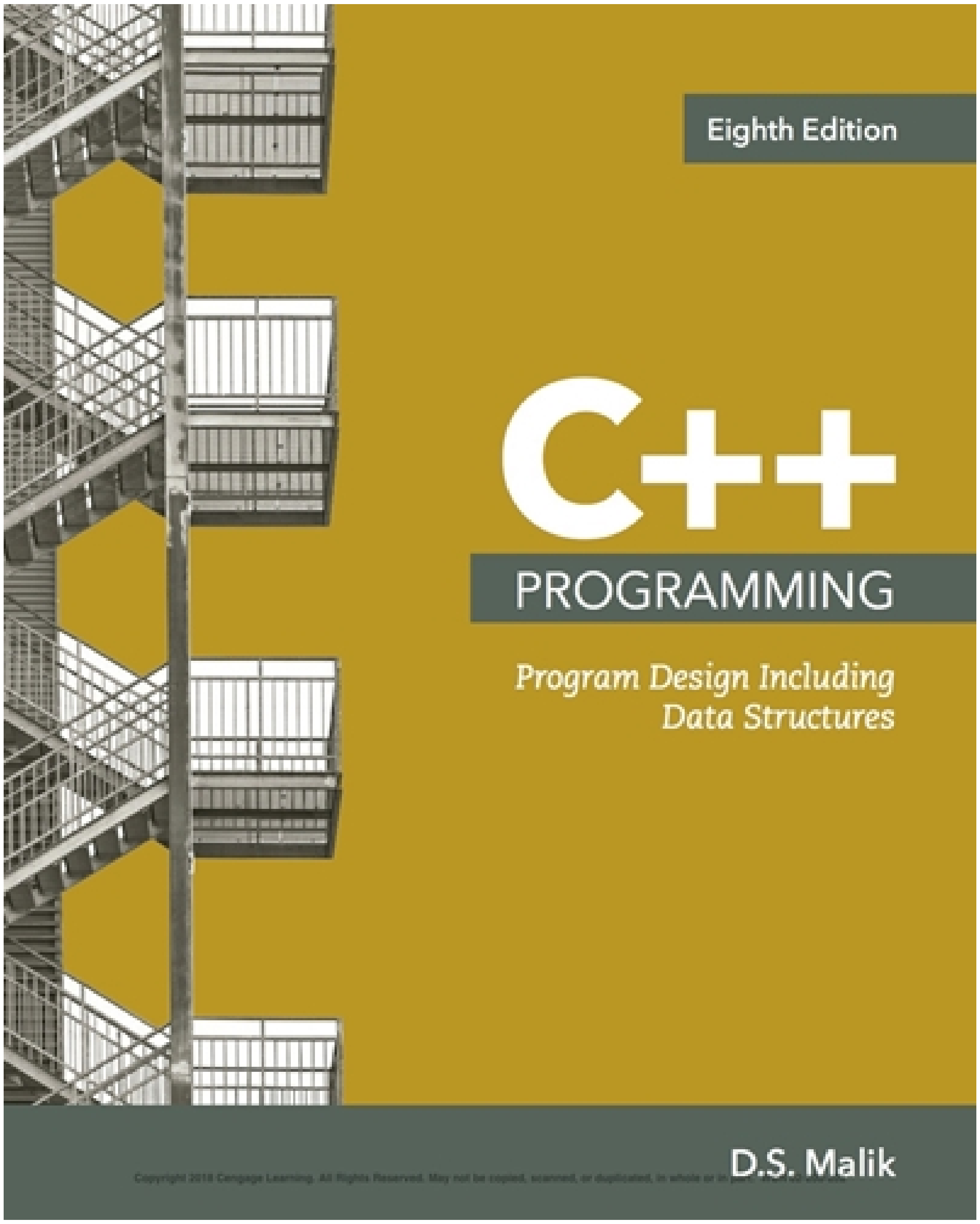
Solutions for C Programming Program Design Including Data Structures 8th Edition by Malik



Solutions

Chapter 1

1. a. true; b. false; c. false; d. true; e. false; f. true; g. true; h. true; i. false; j. false; k. true; l. true; m. true; n. true; o. true; p. false; q. true; r. true; s. true
2. CPU.
3. Base 2 or binary.
4. The equivalent machine language program of a high-level language program.
5. In linking, an object program is combined with other programs in the library, used in the program, to create the executable code.
6. Loader
7. #
8. Preprocessor
9. Programming is a process of problem solving.
10. A step-by-step problem-solving process in which a solution is arrived at in a finite amount of time.
11. (1) Analyze and outline the problem and its solution requirements, and design an algorithm to solve the problem. (2) Implement the algorithm in a programming language, such as C++, and verify that the algorithm works. (3) Maintain the program by using and modifying it if the problem domain changes.
12. (1) Thoroughly understand the problem. (2) Understand the problem requirements. (3) If the problem is complex, divide the problem into subproblems and repeat Steps 1 and 2.
13. To find the weighted average of the four test scores, first you need to know each test score and its weight. Next, you multiply each test score with its weight, and then add these numbers to get the average. Therefore,

```
1. Get testScore1, weightTestScore1
2. Get testScore2, weightTestScore2
3. Get testScore3, weightTestScore3
4. Get testScore4, weightTestScore4
5. weightedAverage = testScore1 * weightTestScore1 +
                     testScore2 * weightTestScore2 +
                     testScore3 * weightTestScore3 +
                     testScore4 * weightTestScore4;
```

14.

```
a. Get quarters
b. Get dimes
c. Get nickels
d. Get pennies
e. changeInPennies = quarters * 25 + dimes * 10 + nickels * 5
                   + pennies
```

15. To find the price per square inch, first we need to find the area of the pizza. Then we divide the price of the pizza by the area of the pizza. Let `radius` denote the radius and `area` denote the area of the circle, and `price` denote the price of pizza. Also, let `pricePerSquareInch` denote the price per square inch.

a. Get `radius`

b. `area = π * radius * radius`

c. Get `price`

d. `pricePerSquareInch = price / area`

16. To determine the least selling price of a car, we need to know the listing price of the car. The algorithm is as follows:

    a. Get `listingPrice`

    b. Calculate the least selling price `leastSellingPrice` using the formula:

    `leastSellingPrice = listingPrice × 0.85) + 500`

17. Suppose that `radius` denotes radius of the sphere, `volume` denotes volume of the sphere, and `surfaceArea` denotes the surface area of the sphere. The following algorithm computes the volume and surface area of the sphere.

Algorithm	**C++ Instruction (Code)**
1. Get the radius.	`cin >> radius;`
2. Calculate the volume.	`volume = (4.0 / 3.0) * 3.1416 * radius * radius * radius;`
3. Calculate the surface area.	`surfaceArea = 4.0 * 3.1416 * radius * radius;`

18. Suppose that `billingAmount` denotes the total billing amount, `movingCost` denotes moving cost, `area` denotes the area of the yard that needs to be moved, `numOfAppl` denotes the number of fertilizing applications, and `numOfTrees` denotes the number of trees to be planted. The following algorithm computes and outputs the billing amount.

    a. Enter the area of the yard.

    b. Get `area`

    c. Enter the number of fertilizing applications.

    d. Get `numOfAppl`

    e. Enter the number of trees to be planted.

    f. Get `numOfTrees`

    g. Calculate the `billingAmount` using the formula:

```
billingAmount = (area / 5000) * 35.00 + numOfAppl * 30.00
           + numOfTrees * 50.00
```

19. Suppose that `billingAmount` denotes the total billing amount, `numOfItemsOrdered` denotes the number of items ordered, `shippingAndHandlingFee` denotes the shipping and handling fee, and `price` denotes the price of an item. The following algorithm computes and outputs the billing amount.

```
a. Enter the number of items bought.
b. Get numOfItemsOrdered
c. billingAmount = 0.0;
d. shippingAndHandlingFee = 0.0;
e. Repeat the following for each item bought.
   i. Enter the price of the item
   ii. Get price
   iii. billingAmount = billingAmount + price;
f. if billingAmount < 200
      shippingAndHandlingFee = 10 * numOfItemsOrdered;
g. billingAmount = billingAmount + shippingAndHandlingFee
i. Print billingAmount
```

20. Suppose `amountWithdrawn` denotes the amount to be withdrawn, `serviceCharge` denotes the service charges, if any, and `accountBalance` denotes the total money in the account.

You can now write the algorithm as follows:

```
a.  Get amountWithdrawn.
b.  if amountWithdrawn > 500
      Print "The maximum amount that can be drawn is $500"
    otherwise (if accountBalance <= 0)
        Print "Account balance is <= 0. You cannot withdraw any money."
    otherwise
    {
        if (amountWithdrawn > accountBalance)
        {
             Print "Insufficient balance. If you withdraw, services charges
                     will be $25.00. Select Yes/No."

             if (Yes)
             {
                 if (amountWithdrawn > 300)
                     serviceCharge = (amountWithdrawn – 300) * 0.04;
                 otherwise
                     serviceCharge = 0;
```

```
                accountBalance = accountBalance - amountWithdrawn
                                 - serviceCharge - 25;
                Print "Collect your money. "
            }
        }
    }
    otherwise
    {
         if (amountWithdrawn > 300)
             serviceCharge = (amountWithdrawn - 300) * 0.04;
        otherwise
             serviceCharge = 0;

        accountBalance = accountBalance - amountWithdrawn
                         - serviceCharge;
        Print "Collect your money."
    }
```

21. Suppose x1 and x2 are the real roots of the quadratic equation.

    a. Get a

    b. Get b

    c. Get c

    d.
```
if (b * b - 4 * a * c < 0)
  Print "The equation has no real roots."
Otherwise
{
    temp = b * b - 4 * a * c;
    x1 = (-b + temp) / (2 * a);
    x2 = (-b - temp) / (2 * a);
}
```

22. Suppose that tuition denotes the tuition per semester, semesterUnits denotes the average semester units, courseUnits denotes the number of course units, semesterWeeks denotes the number of weeks in a semester, classDaysPerWeek denotes the number of days a class meets in a week, classDaysPerSemester denotes the total number of days a class meets in a semester, costPerUnit denotes the cost of one unit of a course, and costPerClass denotes the cost of one class.

    a. Get tuition

    b. Get semesterUnits

    c. Get semesterWeeks

    d. Get courseUnits

    e. Get classDaysPerWeek

    f. Compute costPerUnit using the following formula.

```
costPerUnit = tuition / semesterUnits;
```

    g. Compute classDaysPerSemester using the following formula.

```
classDaysPerSemester = classDaysPerWeek * semesterWeeks;
```

h. Compute `costPerClass` using the following formula.

```
costPerClass = (costPerUnit * courseUnits) / classDaysPerSemester;
```

23. Suppose `averageTestScore` denotes the average test score, `highestScore` denotes the highest test score, `testScore` denotes a test score, `sum` denotes the sum of all the test scores, `count` denotes the number of students in class, and `studentName` denotes the name of a student.

a. First you design an algorithm to find the average test score. To find the average test score, first you need to count the number of students in the class and add the test score of each student. You then divide the sum by count to find the average test score. The algorithm to find the average test score is as follows:

i. Set `sum` and `count` to `0`.

ii. Repeat the following for each student in class.

1. Get `testScore`

2. Increment `count` and update the value of `sum` by adding the current test score to `sum`.

iii. Use the following formula to find the average test score.

```
if (count is 0)
    averageTestScore = 0;
otherwise
    averageTestScore = sum / count;
```

b. The following algorithm determines and prints the names of all the students whose test score is below the average test score.

Repeat the following for each student in class:

i. Get `studentName` and `testScore`

ii.

```
if (testScore is less than averageTestScore)
    print studentName
```

c. The following algorithm determines the highest test score

i. Get first student’s test score and call it `highestTestScore`.

ii. Repeat the following for each of the remaining students in the class

1. Get `testScore`

2.
```
if (testScore is greater than highestTestScore)
    highestTestScore = testScore;
```

d. To print the names of all the students whose test score is the same as the highest test score, compare the test score of each student with the highest test score and if they are equal print the name. The following algorithm accomplishes this.

Repeat the following for each student in the class:

i. Get `studentName` and `testScore`

ii.
```
if (testScore is equal to highestTestScore)
    print studentName
```

You can use the solutions of the subproblems obtained in parts a to d to design the main algorithm as follows:

1. Use the algorithm in part a to find the average test score.
2. Use the algorithm in part b to print the names of all the students whose score is below the average test score.
3. Use the algorithm in part c to find the highest test score.
4. Use the algorithm in part d to print the names of all the students whose test score is the same as the highest test score

Chapter 1

An Overview of Computers and Programming Languages

A Guide to this Instructor's Manual:

We have designed this Instructor's Manual to supplement and enhance your teaching experience through classroom activities and a cohesive chapter summary.

This document is organized chronologically, using the same headings that you see in the textbook. Under the headings, you will find lecture notes that summarize the section, Teacher Tips, Classroom Activities, and Lab Activities. Pay special attention to teaching tips and activities geared towards quizzing your students and enhancing their critical thinking skills.

In addition to this Instructor's Manual, our Instructor's Resources also contain PowerPoint Presentations, Test Banks, and other supplements to aid in your teaching experience.

At a Glance

Instructor's Manual Table of Contents

Lecture Notes

Overview

The aim of this book is to teach your students to design and write programs in C++. However, it is useful for them to understand the basic terminology and different components of a computer before they begin programming. Chapter 1 describes the main components of a computer system, the history and evolution of computer languages, and some fundamental ideas about how to solve problems with computer programming. It also introduces the structured design and object-oriented programming methodologies. Finally, students will look at a C++ program, understand how it is processed, and learn about the ANSI/ISO Standard C++, C++11, and C++14.

Objectives

In this chapter, the student will:

- Learn about different types of computers
- Explore the hardware and software components of a computer system
- Learn about the language of a computer
- Learn about the evolution of programming languages
- Examine high-level programming languages
- Discover what a compiler is and what it does
- Examine a C++ program
- Explore how a C++ program is processed
- Learn what an algorithm is and explore problem-solving techniques
- Become aware of structured design and object-oriented design programming methodologies
- Become aware of Standard C++, ANSI/ISO Standard C++, C++11, and C++14

Teaching Tips

Introduction

1. Discuss the numerous ways in which computers have affected our daily lives, including communications, banking, and course work. Emphasize that these are all made possible by computer programs, which are developed with the help of programming languages.

Teaching Tip	Introduce this chapter with some real-world computer applications. Discuss the relationship between the services that students might typically use when surfing the Internet and the implementation behind them. Use online stores such as https://www.amazon.com/, entertainment databases such as http://www.imdb.com/, or an online investment service such as https://us.etrade.com/b/home as examples.

A Brief Overview of the History of Computers

1. Describe the evolution of computers from the abacus to the present. Explain that at first, computers were large devices only available to a few people, and that by the mid-1990s, computers became affordable and small enough for people from all walks of life.

2. Mention several categories of computers, including mainframes, midsize, and microcomputers.

Teaching Tip	Computer technology is changing rapidly. Ask students to compare their first computer with the one they currently use and discuss the similarities and differences between the two.

Elements of a Computer System

1. Describe the basic commands that a computer performs: input, output, storage, and arithmetic/logical operations.

Hardware

1. Discuss the main hardware components of a computer. These include the central processing unit (CPU), main memory, input/output devices, and secondary storage. Use Figure 1-1 to illustrate how these components work together.

Central Processing Unit and Main Memory

1. Explain that the CPU can be thought of as the brain of the computer and that it is also the most expensive component.

Teaching Tip	The diagram on the following Web page illustrates how CPU components work together: http://computer.howstuffworks.com. It might also be useful to show students an actual motherboard to acquaint them with the innards of a computer.

2. Emphasize that all data must be brought into main memory before it can be processed and that the information in main memory is lost when the computer is turned off.

3. Using Figure 1-1, discuss how main memory is ordered into a sequence of cells.

Secondary Storage

1. Explain why secondary storage is necessary to retain information permanently.

2. Give some examples of secondary storage devices.

Input/Output Devices

1. Explain how input and output devices are needed for a computer to perform useful tasks.

2. Give some examples of input/output devices.

Teaching Tip	Novel input and output devices are appearing regularly. Here is one called "Skinput" that allows you to use the skin on your arm as a keyboard: http://www.chrisharrison.net/index.php/Research/Skinput.

Software

1. Define software as computer programs that are written to perform specific tasks. Give some examples, such as a word processing program.

2. Describe and give examples of the two types of software programs: system and application.

Quick Quiz 1

1. Main memory is an ordered sequence of cells called ____________________.
   Answer: memory cells

2. True or False: Main memory stores numbers and letters.
   Answer: False

3. True or False: Programs do not have to be loaded into main memory before they are executed.
   Answer: False

4. The devices that computers use to display results are called ____________________.
   Answer: output devices

The Language of a Computer

1. Define analog and digital electrical signals and explain the difference between the two. Explain why computers use digital signals.

2. Describe the language of the computer, called machine language. Define the terms binary digit (bit), binary code (binary number), and byte. Use Table 1-1 to illustrate the terms used for various numbers of bytes.

3. Discuss the ASCII encoding scheme commonly used on computers. Explain how a seven-bit ASCII character is converted into an eight-bit representation for computer processing.

Teaching Tip	Show the ASCII character set in Appendix C to your students and talk about its organization. In particular, note the letters, numbers, and new line and tab characters, as these characters will be useful in future programming assignments.

<u>Quick Quiz 2</u>

1. True or False: Analog signals represent information with a sequence of 0s and 1s.
   Answer: False

2. A(n) ______________________ is 1,024 bytes.
   Answer: kilobyte (KB)

3. The basic language of a computer is called ______________________.
   Answer: machine language

4. How do you convert a seven-bit binary representation of an ASCII character to an eight-bit binary representation?
   Answer: Add a 0 to the left of the seven-bit representation of the ASCII encoding of the character

The Evolution of Programming Languages

1. Explain how and why machine languages evolved into assembly language and later into high-level languages such as Basic, FORTRAN, C++, and Java.

2. Give a brief overview of machine language instructions using the example in the textbook.

Teaching Tip	Point out that the process of writing programs in machine language was extremely tedious and error-prone.

3. Give a brief overview of assembly language instructions using the sample code in the text. Introduce the concept of a mnemonic.

4. Explain how an assembler works.

5. Explain the role of a compiler in high-level languages.

Processing a C++ Program

1. This section introduces a simple C++ program to your class. Step through the program in this section, line by line, and explain how C++ processes output.

2. Point out all C++ keywords and functions in this C++ program, such as `#include`, `using namespace`, `return`, and `cout`. In addition, discuss C++ syntax, including the use of braces, parentheses, and the stream insertion operator (<<).

3. Using Figure 1-2, review the steps required to process a program written in C++. Define the terms source code (program), preprocessor, object program, library, linker, and loader as you discuss the process.

Teaching Tip	Students are probably anxious about using a C++ integrated development environment (IDE). Demonstrate how to use the Visual C++ Express or Visual Studio 2015 IDE with the program in the previous section. Explain the development process by pasting the code into the text editor, compiling and building the project, and executing the program.

Quick Quiz 3

1. A C++ source program must be saved in a text file with a(n) ____________________ extension.
   Answer: .cpp

2. What is the role of a compiler?
   Answer: A compiler checks the source program for errors and, if no error is found, translates the program into the equivalent machine language.

3. True or False: A loader is a program that combines the object program with other programs in the library.

Answer: False

Programming with the Problem Analysis-Coding-Execution Cycle

1. Define the term algorithm and describe the three-step problem-solving process in a programming environment.

2. Discuss the components of the problem analysis-coding-execution cycle using Figure 1-3.

Teaching Tip	Emphasize the need for problem analysis before coding with real-world examples, i.e., building an airplane or space shuttle. What might happen if shortcuts are taken?

3. Illustrate how to design an algorithm in C++ using Examples 1-1 through 1-5.

Programming Methodologies

1. This section introduces two popular approaches to programming design: structured design and object-oriented design.

Structured Programming

1. Define structured design as dividing a problem into smaller subproblems. Further clarify this concept by explaining equivalent terms for this approach: top-down design, bottom-up design, stepwise refinement, and modular programming.

Object-Oriented Programming

1. Spend a considerable amount of time introducing object-oriented design. Explain that the first step in the process is to identify components called objects, which form the basis of the problem solution, and determine how the objects will interact with each other. The next step is to specify the data that is relevant to each object as well as the operations to be performed on that data.

2. Use the video object example in the text to illustrate some possible data and operations of objects.

Teaching Tip	Object-oriented design may be initially difficult for your students to grasp. Illustrate the concept of an object further by using a music app such as Google Play. Show the app on a device and search for the songs of a favorite artist. Ask students to think of the song as an object and to describe both its data and its operations.

3. Provide a brief overview of the subject matter that will be covered in subsequent chapters, including C++ data types, functions, control structures, and classes. Finally, emphasize that when programming in C++, OOD works well in conjunction with structured design.

<u>Quick Quiz 4</u>

1. What is another term for structured design?
   Answer: top-down design, stepwise refinement, modular programming

2. In OOD, the first step in the problem-solving process is to identify the components called ____________________.
   Answer: objects

3. True or False: An object consists of data and operations on that data.
   Answer: True

4. True or False: C++ was designed especially to implement OOD.
   Answer: True

ANSI/ISO Standard C++

1. Discuss the history of C++ and how the C++ language evolved in slightly different ways, depending on the compiler. Explain that before ANSI/ISO Standard C++, programmers were not assured that their program would run in a different environment from the one on which it was developed.

2. In 2011, the second standard of C++ was approved. The main objective of this standard, referred to as C++11, is to make the C++ code cleaner and more effective. C++14 was approved as an update in 2014.

Teaching Tip	Discuss why the ANSI/ISO Standard makes for more robust and dependable C++ programs, but that there may still be portability issues with older or noncompliant compilers. If possible, demonstrate this issue with a couple of different compilers.

Class Discussion Topics

1. Ask your students to talk about any previous programming experience they might have had. Discuss the object-oriented approach to programming as it relates to any of the languages brought up in the discussion. Assure students with no background in programming that they will be able to pick up the basics with a little practice.

2. Many companies use internal programs that were implemented before object-oriented programming came into widespread use. What are the issues involved in moving their systems to an object-oriented approach?

Additional Projects

1. Ask students to do some research and create a timetable that describes the evolution of computer languages from machine language to the present day. The table does not have to be comprehensive, but it should include approximately ten major languages. For each language, they should include the creation date and author(s), a brief description, and the state of its use today.

2. Ask your students how they would like their final class grade to be determined in terms of the relative weight given to projects, weekly assignments, midterm, final, etc. Then, ask them to design an algorithm to calculate their final grade using the percentages they have chosen.

Additional Resources

1. History and evolution of computer languages:
   http://www.scriptol.com/programming/history.php

2. A list of applications written in C++:
   http://www.stroustrup.com/applications.html

3. ASCII table and description:
   http://www.asciitable.com/

4. An illustrated history of computers:
   http://www.computersciencelab.com/ComputerHistory/History.htm

5. An overview of the upcoming C++17 standard:
   https://isocpp.org/std/status

Key Terms

- **Address (of a cell)**: **a** unique location in main memory for each cell
- **Algorithm**: a step-by-step problem-solving process in which a solution is arrived at in a finite amount of time
- **American Standard Code for Information Interchange (ASCII)**: the most commonly used encoding scheme used on personal computers; the ASCII data set uses seven bits to represent 128 characters, numbered from 0 to 127
- **Analog signal**: a continuous wave form used to represent such things as sound
- **Application program**: a software program that performs a specific task
- **Assembler**: a program that translates a program written in assembly language into an equivalent program in machine language
- **Binary (base 2)**: the number system that a computer uses
- **Binary code (binary number)**: a sequence of 0s and 1s
- **Binary digit (bit)**: the digit 0 or 1
- **Bit**: a binary digit 0 or 1
- **Build (Rebuild)**: the command that does the linking on Visual C++ and Visual Studio .NET
- **Byte**: a sequence of eight bits
- **Central processing unit (CPU)**: the brain of the computer and the single most expensive piece of hardware in a personal computer
- **Compiler**: a program that translates instructions written in a high-level language into the equivalent machine language
- **Decimal system (base 10)**: the number system we use in daily life
- **Digital signal**: represents information with a sequence of 0s and 1s
- **High-level language**: a programming language that is similar to natural speaking languages
- **Input device**: a device that feeds data and programs into a computer
- **Kilobyte (KB)**: 1024, or 2^{10} bytes
- **Library**: includes prewritten code
- **Linker**: a program that combines the object program with other programs in the library, and is used in the program to create the executable code
- **Loader**: a program that loads an executable program into main memory
- **Machine language**: the language of a computer; a sequence of 0s and 1s
- **Main memory**: memory that is directly connected to the CPU
- **Memory cells**: an ordered sequence of cells in main memory
- **Mnemonic**: an instruction that is in an easy-to-remember form
- **Object program**: the machine language version of the high-level language program
- **Object-oriented design (OOD)**: a programming methodology that identifies components called objects, which form the basis of the solution to a problem

- **Object-oriented programming (OOP) language**: a programming language that implements OOD
- **Operating system**: monitors the overall activity of the computer and provides services
- **Output device**: a device that the computer uses to display results
- **Preprocessor**: a program that processes statements in a C++ program that begin with the symbol #
- **Random access memory**: memory that is directly connected to the CPU
- **Secondary storage**: a device that stores information permanently
- **Source code (source program)**: a program that is written in a high-level language
- **Structured design (top-down design, bottom-up design, stepwise refinement, modular programming)**: the act of dividing a problem into smaller subproblems
- **Structured programming**: the process of implementing a structured design
- **System program**: a program that controls the computer