Introduction to Computers, the Internet and the Web

The chief merit of language is clearness.

—Galen

Our life is frittered away by detail. …Simplify, simplify.

—Henry David Thoreau

He had a wonderful talent for packing thought close, and rendering it portable.

—Thomas B. Macaulay

Man is still the most extraordinary computer of all.

—John F. Kennedy

Things are always at their best in their beginning.

—Blaise Pascal.

OBJECTIVES

In this chapter you will learn:

- Basic computer concepts.
- The different types of programming languages.
- The history of the C programming language.
- The purpose of the C Standard Library.
- The elements of a typical C program development environment.
- Why it is appropriate to learn C in a first programming course.
- How C provides a foundation for further study of programming languages in general and of C++, Java and C# in particular.
- The history of the Internet and the World Wide Web.

Outline

Outline

1.1 Introduction

- **We will learn**
  - **The C programming language**
  - **Structured programming and proper programming techniques**
- **This book also covers C++**
  - **Chapters 18-27 introduce the C++ programming language**
- **This course is appropriate for**
  - **Technically oriented people with little or no programming experience**
  - **Experienced programmers who want a deep and rigorous treatment of the language**

1.2 What is a Computer?

- **Computer**
  - Device capable of performing computations and making logical decisions
  - Computers process data under the control of sets of instructions called computer programs
- **Hardware**
  - Various devices comprising a computer
  - Keyboard, screen, mouse, disks, memory, CD-ROM, and processing units
- **Software**
  - Programs that run on a computer

1.3 Computer Organization

- **Six logical units in every computer:**
  1. **Input unit**
     - **Obtains information from input devices (keyboard, mouse)**
  2. **Output unit**
     - **Outputs information (to screen, to printer, to control other devices)**
  3. **Memory unit**
     - **Rapid access, low capacity, stores input information**
  4. **Arithmetic and logic unit (ALU)**
     - **Performs arithmetic calculations and logic decisions**
  5. **Central processing unit (CPU)**
     - **Supervises and coordinates the other sections of the computer**
  6. **Secondary storage unit**
     - **Cheap, long-term, high-capacity storage**
     - **Stores inactive programs**

1.4 Early Operating Systems

- **Batch processing**
  - **Do only one job or task at a time**
- **Operating systems**
  - **Manage transitions between jobs**
  - **Increased throughput**
    - **Amount of work computers process**
- **Multitasking**
  - **Computer resources are shared by many jobs or tasks**
- **Timesharing**
  - **Computer runs a small portion of one user's job then moves on to service the next user**

1.5 Personal Computing, Distributed Computing, and Client/Server Computing

- **Personal computers**
  - **Economical enough for individual**
- **Distributed computing**
  - **Computing distributed over networks**
- **Client/server computing**
  - **Sharing of information across computer networks between file servers and clients (personal computers)**

1.6 Machine Languages, Assembly Languages, and High-level Languages

Three types of programming languages

1. **Machine languages**
   - **Strings of numbers giving machine specific instructions**
   - **Example:**

```
+1300042774
+1400593419
+1200274027
```

2. **Assembly languages**
   - **English-like abbreviations representing elementary computer operations (translated via assemblers)**
   - **Example:**

```
LOAD    BASEPAY
ADD     OVERPAY
STORE   GROSSPAY
```

1.6 Machine Languages, Assembly Languages, and High-level Languages

Three types of programming languages (continued)

3. **High-level languages**
   - **Codes similar to everyday English**
   - **Use mathematical notations (translated via compilers)**
   - **Example:**

```
grossPay = basePay + overTimePay
```

1.7 Fortran, COBOL, Pascal and Ada

- **Fortran**
  - **developed by IBM Corporation in the 1950s**
  - **used for scientific and engineering applications that require complex mathematical computations**
- **COBOL**
  - **developed in 1959 by computer manufacturers, the government and industrial computer users**
  - **used for commercial applications that require precise and efficient manipulation of large amounts of data**

1.7 Fortran, COBOL, Pascal and Ada

- **Pascal**
  - **Developed by Professor Niklaus Wirth in 1971**
  - **Designed for teaching structured programming**
- **Ada**
  - **Developed under the sponsorship of the U.S. Department of Defense (DOD) during the 1970s and early 1980s**
  - **Able to perform multitasking**

1.8 History of C

- **C**
  - **Evolved by Ritchie from two previous programming languages, BCPL and B**
  - **Used to develop UNIX**
  - **Used to write modern operating systems**
  - **Hardware independent (portable)**
  - **By late 1970's C had evolved to "Traditional C"**
- **Standardization**
  - **Many slight variations of C existed, and were incompatible**
  - **Committee formed to create a "unambiguous, machine-independent" definition**
  - **Standard created in 1989, updated in 1999**

Portability Tip 1.1

Because C is a hardware-independent, widely available language, applications written in C can run with little or no modifications on a wide range of different computer systems.

1.9 C Standard Library

- C programs consist of pieces/modules called functions
  - A programmer can create his own functions
    - Advantage: the programmer knows exactly how it works
    - Disadvantage: time consuming
  - Programmers will often use the C library functions
    - Use these as building blocks
  - Avoid re-inventing the wheel
    - If a pre-made function exists, generally best to use it rather than write your own
    - Library functions carefully written, efficient, and portable

Performance Tip 1.1

Using Standard C library functions instead of writing your own comparable versions can improve program performance, because these functions are carefully written to perform efficiently.

Portability Tip 1.2

Using Standard C library functions instead of writing your own comparable versions can improve program portability, because these functions are used in virtually all Standard C implementations.

1.10 C++

- C++
  - Superset of C developed by Bjarne Stroustrup at Bell Labs
  - "Spruces up" C, and provides object-oriented capabilities
  - Dominant language in industry and academia
- Learning C++
  - Because C++ includes C, some feel it is best to master C, then learn C++
  - Starting in Chapter 18, we begin our introduction to C++

1.11 Java

- **Java is used to**
  - **Create Web pages with dynamic and interactive content**
  - **Develop large-scale enterprise applications**
  - **Enhance the functionality of Web servers**
  - **Provide applications for consumer devices (such as cell phones, pagers and personal digital assistants)**
- **Java How to Program**
  - **Closely followed the development of Java by Sun**
  - **Teaches first-year programming students the essentials of graphics, images, animation, audio, video, database, networking, multithreading and collaborative computing**

1.12 BASIC, Visual Basic, Visual C++, Visual C# and .NET

- **BASIC**
  - **Developed in the mid-1960s by Professors John Kemeny and Thomas Kurtz of Dartmouth College as a language for writing simple programs**
- **Visual Basic**
  - **Introduced by Microsoft in 1991 to simplify the process of making Windows applications**
- **Visual Basic, Visual C++, and Visual C#**
  - **Designed for Microsoft's .NET programming platform**

1.13 Key Software Trend: Object Technology

- **Objects**
  - **Reusable software components that model items in the real world**
  - **Meaningful software units**
    - **Date objects, time objects, paycheck objects, invoice objects, audio objects, video objects, file objects, record objects, etc.**
    - **Any noun can be represented as an object**
  - **Very reusable**
  - **More understandable, better organized, and easier to maintain than procedural programming**
  - **Favor modularity**

1.14 Typical C Program Development Environment

- Phases of C++ Programs:
  - *Edit*
  - *Preprocess*
  - *Compile*
  - *Link*
  - *Load*
  - *Execute*

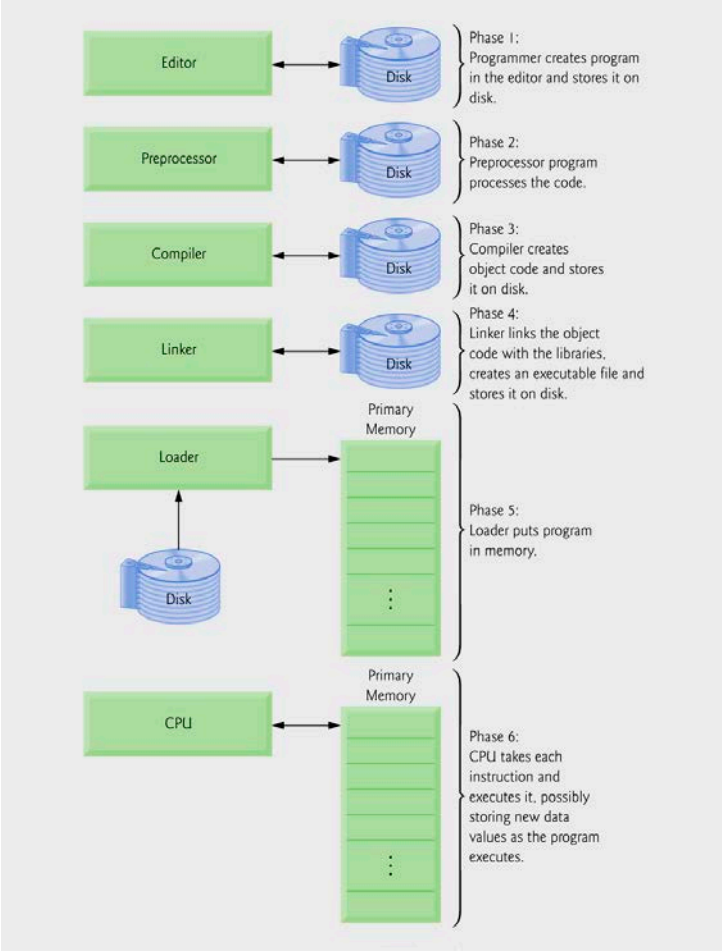

Fig. 1.1 | Typical C development environment.

Common Programming Error 1.1

Errors like division-by-zero occur as a program runs, so these errors are called runtime errors or execution-time errors. Divide-by-zero is generally a fatal error, i.e., an error that causes the program to terminate immediately without successfully performing its job. Nonfatal errors allow programs to run to completion, often producing incorrect results. [Note: *On some systems, divide-by-zero is not a fatal error. Please see your system documentation*.]

1.15 Hardware Trends

- **Every year or two the following approximately double:**
  - **Amount of memory in which to execute programs**
  - **Amount of secondary storage (such as disk storage)**
    - **Used to hold programs and data over the longer term**
  - **Processor speeds**
    - **The speeds at which computers execute their programs**

1.16 History of the Internet

- **The Internet enables**
  - **Quick and easy communication via e-mail**
  - **International networking of computers**
- **Packet switching**
  - **The transfer of digital data via small packets**
  - **Allows multiple users to send and receive data simultaneously**
- **No centralized control**
  - **If one part of the Internet fails, other parts can still operate**
- **TCP/IP**
- **Bandwidth**
  - **Information carrying capacity of communications lines**

1.17 History of the World Wide Web

- **World Wide Web**
  - **Locate and view multimedia-based documents on almost any subject**
  - **Makes information instantly and conveniently accessible worldwide**
  - **Possible for individuals and small businesses to get worldwide exposure**
  - **Changing the way business is done**

1.18 General Notes About C and This Book

- **Program clarity**
  - **Programs that are convoluted are difficult to read, understand, and modify**
- **C is a portable language**
  - **Programs can run on many different computers**
  - **However, portability is an elusive goal**
- **We will do a careful walkthrough of C**
  - **Some details and subtleties are not covered**
  - **If you need additional technical details**
    - **Read the C standard document**
    - **Read the book by Kernigan and Ritchie**

Good Programming Practice 1.1

Write your C programs in a simple and straightforward manner. This is sometimes referred to as KIS (“keep it simple”). Do not “stretch” the language by trying bizarre usages.

Portability Tip 1.3

Although it is possible to write portable programs, there are many problems between different C compilers and different computers that make portability difficult to achieve. Simply writing programs in C does not guarantee portability. The progammer will often need to deal directly with complex computer variations.

Software Engineering Observation 1.1

Read the manuals for the version of C you are using. Reference these manuals frequently to be sure you are aware of the rich collection of C features and that you are using these features correctly.

Software Engineering Observation 1.2

Your computer and compiler are good teachers. If you are not sure how a feature of C works, write a sample program with that feature, compile and run the program and see what happens.