Programming and Languages

Chapter 14
Telling the Computer What to Do

Copyright © Prentice Hall 2000

Programs

A program is a set of step-by-step instructions that directs the computer to do the tasks you want it to do and produce the results you want.

Copyright © Prentice Hall 2000

Programming Languages

A programming language is a set of rules that provides a way of telling a computer what operations to perform.

Copyright © Prentice Hall 2000

Programmers

A programmer’s job is to convert problem solutions into instructions for the computer. If the project is large, they may also need to coordinate the needs of users, managers, and systems managers.

Copyright © Prentice Hall 2000

The Programming Process

The steps involved in developing a program include:
- Define the problem
- Plan the solution
- Code the program
- Test the program
- Document the program

Copyright © Prentice Hall 2000

Define the Problem

The task of defining the problem consists of identifying what it is you know and what it is you want to obtain.

Copyright © Prentice Hall 2000
Plan the Solution

Planning the solution may involve drawing a flowchart or writing pseudocode, or both.

Flowcharts

A flowchart is a pictorial representation of a step-by-step solution to a problem.

Flowchart Basics

A flowchart consists of arrows to represent direction the program takes and boxes and symbols to represent actions.

Flowchart Symbols

- Process
- Start/Stop
- Connector
- Input/Output
- Decision
- Flow direction

Pseudocode

Pseudocode is an English-like nonstandard language. It allows programmers to focus on the program logic without being concerned about the particulars of a formal programming language.

Code the Program

Coding a program means to translate the logic from a flowchart of pseudocode into a programming language. The rules of computer languages are called syntax. These rules must be followed precisely.
Test the Program

After coding, tests must be performed to make sure the steps are correct and render the desired results. Testing includes:
- Desk-checking
- Translating
- Debugging

Desk-checking

This form of testing involves mentally checking the logic of the program to ensure that it is error-free and workable.

Translating

Programs are commonly translated by a compiler which checks for syntax errors and converts the program into a form the computer understands.

Debugging

Debugging means to detect, locate, and correct mistakes in the program. To find errors, you must test every part of the program under various conditions.

Document the Program

A programmer must document the various stages the program has gone through as well as other specific facts about the program.

Levels of Programming Languages

There are several levels (or “generations”) of programming languages:
- Machine
- Assembly
- High-level
- Very high-level
- Natural
First Generation: Machine Language

This is the lowest level of programming language because it represents data and program instructions as 0s and 1s. All programming languages are eventually converted into machine language.

Second Generation: Assembly Language

Assembly languages replace 0s and 1s with mnemonic codes. Since machine language is the only language the computer can execute, assembly is eventually translated into machine language to execute the program.

Third Generation: High-Level Language

High-level languages use English-like words that are much easier for humans to understand. A translator is needed to convert the high-level language into machine language that computers understand.

Fourth Generation: Very High-Level Language

This generation of language is often known as 4GLs. 4GLs are a shorthand programming language that is about 10 times more productive than third generation languages.

Fifth Generation: Natural Language

This generation of programming languages more resembles “natural” spoken English. The user communicates with the computer by speaking.

Major Programming Languages

There are several languages with which to write your program:

- FORTRAN
- COBOL
- BASIC
- Pascal
- C
- Java
- Object-Oriented
FORTRAN

FORTRAN is a scientifically oriented language designed to execute complex formulas as in economic analysis and engineering.

COBOL

COBOL was designed for business needs such as processing large files and performing business calculations.

BASIC

This programming language is designed to be easy to learn. BASIC can be used for business applications and for use with personal computers.

Pascal

Pascal was developed as a teaching language for computer students.

C

C was originally developed to write system's software but is now considered a general-purpose language.

Java

Java is a network-friendly programming language that permits a piece of software to run on many different platforms.
Object-Oriented

Object-oriented programming is based on objects and their attributes.

Object Classes

An object is a self-contained unit that contains both data and related facts and functions. Objects are arranged hierarchically in classes and subclasses based on their dominant characteristics.

Attributes and Methods

Facts related to an object are called attributes. Methods are the instructions that tell the data what to do.

Object Inheritance

An object in a subclass automatically possesses all the characteristics of the class from which it is derived.

Object-Oriented Languages

There is an object-oriented version of C, called C++. Versions of C++ are available for large and personal computers. Smalltalk supports a visual system of programming. Text is entered with a keyboard but all other interaction takes place with mouse and icons.

Advantages of Object-Oriented Programming

In a programming environment, a programmer does not have to repeat the instructions for “inherited” characteristics. This means a savings in time and money.
Sample Programs

The following five slides demonstrate the syntax and structure of high-level programming languages. Only segments of an entire program are shown. Can you “guess” what each line in the program will do?

Fortran

\begin{verbatim}
WRITE (6,10)
SUM=0
COUNTER = 0
WRITE (6,60)
READ (5,40) NUMBER
IF (NUMBER .EQ. 999) GOTO 2
SUM = SUM + NUMBER
GOTO 1
\end{verbatim}

COBOL

\begin{verbatim}
300-INITIALIZATION-ROUTINE.
DISPLAY "PLEASE ENTER A NUMBER".
ACCEPT NUMBER-ITEM.
400-ENTER-AND-ADD.
ADD NUMBER-ITEM TO SUM-ITEM.
ADD 1 TO COUNTER.
DISPLAY "PLEASE ENTER NEXT NUMBER"
\end{verbatim}

BASIC

\begin{verbatim}
SUM = 0
COUNTER = 0
PRINT "PLEASE ENTER A NUMBER"
INPUT NUMBER
DO WHILE NUMBER <> 999
    SUM = SUM + NUMBER
    COUNTER = COUNTER + 1
    PRINT "Please enter the next number"
\end{verbatim}

Pascal

\begin{verbatim}
BEGIN
  sum :=0;
  counter :=0;
  WRITELN ("PLEASE ENTER A NUMBER");
  READLN (number);
  WHILE number <> 999 DO
    Begin (*while loop*)
      sum := sum + number;
    counter ++;
    WRITELN ("Please enter the next number");
\end{verbatim}

C++

\begin{verbatim}
cout << "PLEASE ENTER A NUMBER";
cin>> number;
while (number !=999)
{
    sum := sum + number;
    counter ++;
    cout <<"Please enter the next number":
\end{verbatim}
Learning OOP

- Read books
- Use the tutorials
- Study sample code
- Write OO code
- Start small
- Use the help file
- Use your time productively

Conclusion

Programming languages have evolved along with the hardware they instruct.
In addition, there is a variety of programming languages designed to fit the problem the programmer is trying to solve.