Chapter 2:
Input, Processing, and Output

Starting Out with Programming Logic & Design

Second Edition

by Tony Gaddis

Addison-Wesley
is an imprint of
PEARSON

Chapter Topics

2.1 Designing a Program

2.2 Output, Input, and Variables

2.3 Variable Assignment and Calculations

2.4 Variable Declarations and Data Types

2.5 Named Constants

2.6 Hand Tracing a Program

2.7 Documenting a Program

2.1 Designing a Program

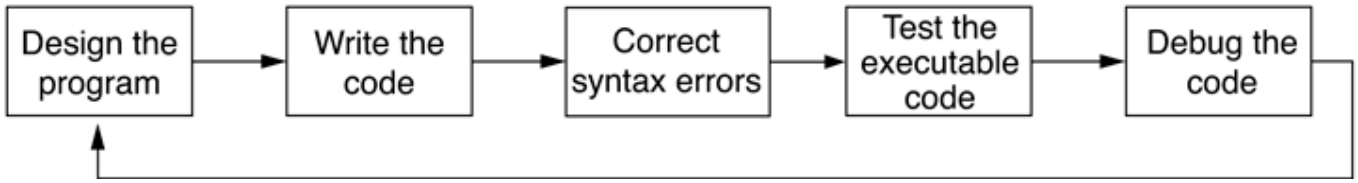
1. The first step in programming is designing – **flowcharts** and **pseudocode** help with this process.
2. Next, the code is written.
3. All code must be cleared of all **syntax errors**.
4. After the executable is created, it can be checked for **logic errors.**
5. If logic errors exist, the program must be **debugged**.

2.1 Designing a Program

The purpose of Programming Logic and Design is to focus on Flowcharts and Pseudocode.

The design is the foundation of a good program.

Figure 2-1 The program development cycle



2.1 Designing a Program

Two steps in designing a program

1. Understand the tasks that the program is to perform.
   - Learning what the customer wants.
2. Determine the steps that must be taken to perform the task.
   - Create an algorithm, or step-by-step directions to solve the problem.
   - Use flowcharts and/or pseudocode to solve.

2.1 Designing a Program

Pseudocode

- Fake code used as a model for programs
- No syntax rules
- Well written pseudocode can be easily translated to actual code

Display "Enter the number of hours"
Input hours
Display "Enter the hourly pay rate"
Input payRate
*Set grossPay = hours * payRate*
Display "The gross pay is $", grossPay

2.1 Designing a Program

Flowcharts

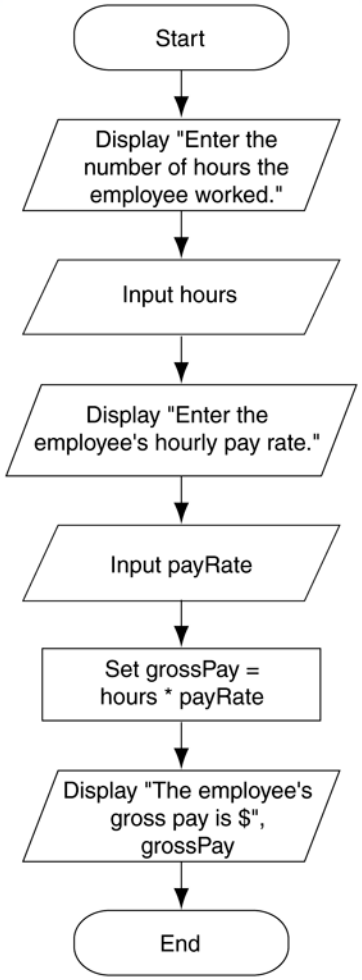
- A diagram that graphically depicts the steps that take place in a program

Terminator used for start and stop

Parallelogram used for input and output

Rectangle used for processes

Figure 2.2 Flowchart for the pay calculating program



2.2 Output, Input, and Variables

Output – data that is generated and displayed

Input – data that a program receives

Variables – storage locations in memory for data

Computer programs typically follow 3 steps

1. Input is received
2. Some process is performed on the input
3. Output is produced

2.2 Output, Input, and Variables

Display is the keyword to show output to the screen

Sequence – lines execute in the order they appear

String Literals – a sequence of characters

Figure 2-4 The statements execute in order

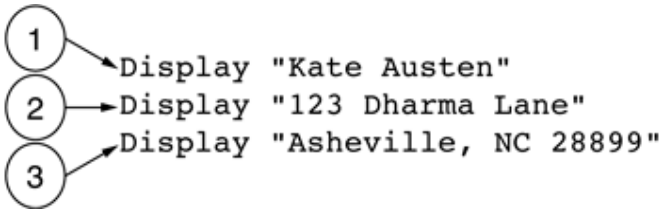
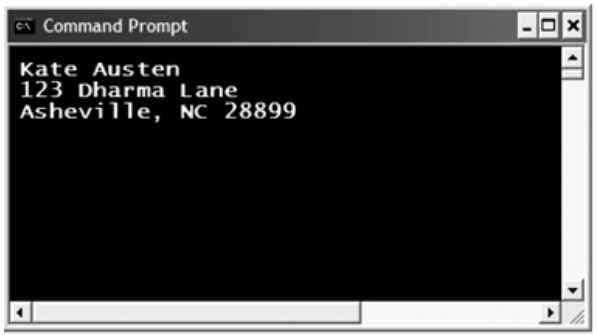


Figure 2-5 Output of Program 2-1



2.2 Output, Input, and Variables

Input is the keyword to take values from the user of the program

It is usually stored in *variables*

Program 2-2

```
1  Display "What is your age?"
2  Input age
3  Display "Here is the value that you entered:"
4  Display age
```

Program Output (with Input Shown in Bold)

```
What is your age?
24 [Enter]
Here is the value that you entered:
24
```

2.2 Output, Input, and Variables

Programmers can define variable names following certain rules

- Must be one word, no spaces
- Generally, punctuation characters are avoided
- Generally, the first character cannot be a number
- Name a variable something that indicates what may be stored in it

camelCase is popular naming convention

2.3 Variable Assignment & Calculations

Variable assignment does not always have to come from user input, it can also be set through an assignment statement

Set price = 20

Program 2-6

```
1  Set dollars = 2.75
2  Display "I have ", dollars, " in my account."
3  Set dollars = 99.95
4  Display "But now I have ", dollars, " in my account!"
```

Program Output

```
I have 2.75 in my account.
But now I have 99.95 in my account!
```

2.3 Variable Assignment & Calculations

Calculations are performed using math operators

The expression is normally stored in variables

Set sale = price – discount

Table 2-1 Common math operators

Symbol	Operator	Description
+	Addition	Adds two numbers
–	Subtraction	Subtracts one number from another
*	Multiplication	Multiplies one number by another
/	Division	Divides one number by another and gives the quotient
MOD	Modulus	Divides one number by another and gives the remainder
^	Exponent	Raised a number to a power

2.4 Variable Declarations & Data Types

A *variable declaration* includes a variable's name and a variable's data type

Data Type – defines the type of data you intend to store in a variable

- Integer – stores only whole numbers
- Real – stores whole or decimal numbers
- String – any series of characters

- *Declare Real grossPay*

2.4 Variable Declarations & Data Types

For safety and to avoid logic errors, variables should be *initialized* to 0 or some other value

Program 2-12

```
 1  Declare Real test1
 2  Declare Real test2
 3  Declare Real test3
 4  Declare Real average
 5
 6  Set test1 = 88.0
 7  Set test2 = 92.5
 8  Set test3 = 97.0
 9  Set average = (test1 + test2 + test3) / 3
10  Display "Your average test score is ", average
```

Program Output (with Input Shown in Bold)

```
Your average test score is 92.5
```

2.5 Named Constants

A *named constant* is a name that represents a value that cannot be changed

- Makes programs more self explanatory
- If a change to the value occurs, it only has to be modified in one place

Constant Real INTEREST_RATE = 0.069

2.6 Hand Tracing a Program

Hand tracing is a simple debugging process for locating hard to find errors in a program

Involves creating a chart with a column for each variable, and a row for each line of code

Figure 2-14 Program with the hand trace chart completed

```
1  Declare Real test1
2  Declare Real test2
3  Declare Real test3
4  Declare Real average
5
6  Set test1 = 88.0
7  Set test2 = 92.5
8  Set average = (test1 + test2 + test3) / 3
9  Display "Your average test score is ", average
```

	test1	test2	test3	average
1	?	?	?	?
2	?	?	?	?
3	?	?	?	?
4	?	?	?	?
5	?	?	?	?
6	88	?	?	?
7	88	92.5	?	?
8	88	92.2	?	undefined
9	88	92.5	?	undefined

2.7 Documenting a Program

External documentation describes aspects of the program for the user, sometimes written by a technical writer

Internal documentation explains how parts of the program works for the programmer, also known as *comments*

// comments are often distinguished within

// the program with line comments