CIS1400 – Programming Logic and Technique

Topic 2 → Simple Data Types and the Sequential Control Structure

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

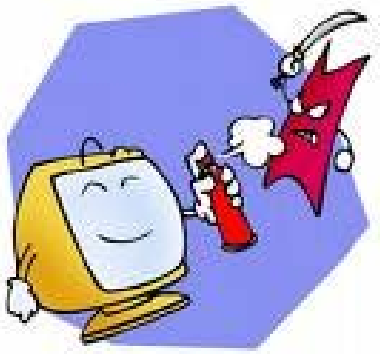
2.7 Documenting a Program

2.8 Designing Your First 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**.

Syntax errors caught during compile. Logic errors use test cases to verify expected output with actual output.

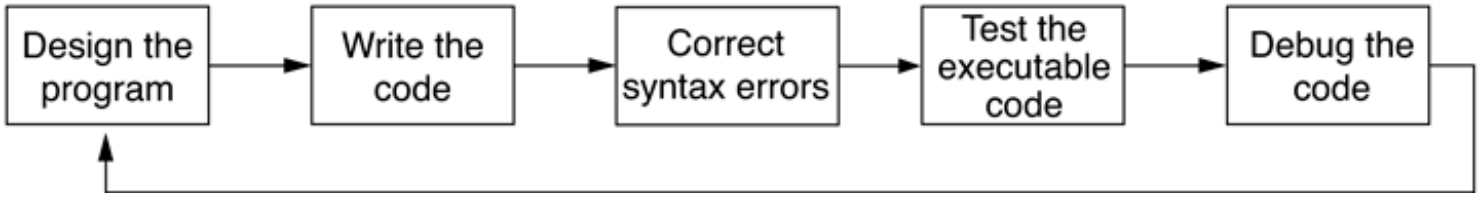


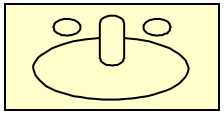
2.1 Designing a Program

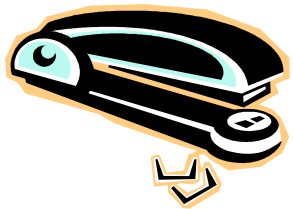
The purpose of Programming Logic and Design is to focus on Flowcharts and Pseudocode.

The design is the foundation of a good program.

Figure 2-1 The program development cycle







"Good specifications will always improve programmer productivity far better than any programming tool or technique."
– Milt Bryce

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.

Calculate and Display Gross Pay for Hourly Employee

1. Get the number of hours worked.
2. Get the hourly pay rate.
3. Multiple the number of hours worked by the hourly pay rate.
4. Display the result of the calculation that was performed in Step 3.

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

See Appendix C in textbook for Pseudocode Reference..

"Computers are good at following instructions, but not at reading your mind."
– Donald Knuth

2.1 Designing a Program

Flowcharts

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

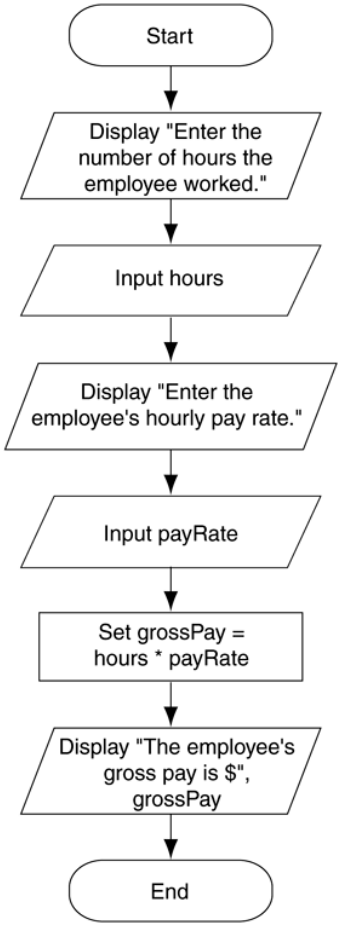
Terminator used for start and stop

Parallelogram used for input and output

Rectangle used for processes

See Appendix B in textbook for Standard Flowchart Symbols.

Figure 2.2 Flowchart for the pay calculating program



2.2 Output, Input, and Variables

Output – data that has been processed into usable form (information)

Input – data that a program receives

Variables – storage locations in memory for data

Computer programs typically follow 3 (possibly 4) steps

1. Input is received
2. Some process is performed on the input
3. Output is produced
4. Data/Information kept for later use on storage media

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

Flowchart for this program is in Figure 2-6.

```
1  Display "Kate Austen"
2  Display "123 Dharma Lane"
3  Display "Asheville, NC 28899"
```

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
```

Flowchart for this program is in Figure 2-7.

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!"
```

Note display of multiple items with single statement.

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

Strongly typed programming languages usually require data type declaration before variable use.

- Functions only applied to intended data types.

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 2-13

```
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
```

Can combine declaration and initialization:
Declare Real test1 = 88.0

Program Output

```
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 2-15 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

2.7 Documenting a Program

Block comments take up several lines and are used for lengthy explanations

Line comments occupy a single line and explain short section of the program

- *Does not need to occupy an entire line*
- *Anything after // to end of line is ignored by compiler/interpreter*

"Commenting your code is like cleaning your bathroom — you never want to do it, but it really does create a more pleasant experience for you and your guests."
– Ryan Campbell

2.8 Designing Your First Program

Calculate the Batting Average For Any Player

Batting Average = Hits ÷ Times at Bat

What is required for each phase of the program?

1. What must be read as input?
   - The number of hits
   - The number of times at bat
2. What will be done with the input?
   - Calculate the batting average
   - Divide the number of hits by the number of times at bat
3. What will be done with the output?
   - Display the player's batting average

2.8 Designing Your First Program

Program 2-15

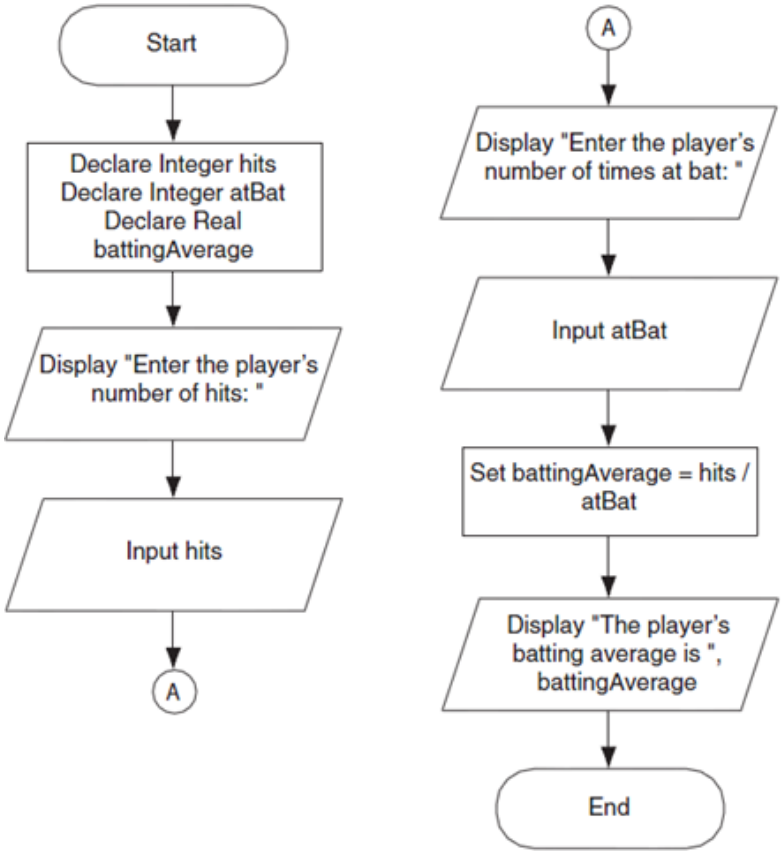
```
1  // Declare the necessary variables.
2  Declare Integer hits
3  Declare Integer atBat
4  Declare Real battingAverage
5
6  // Get the number of hits.
7  Display "Enter the player's number of hits."
8  Input hits
9
10 // Get the number of times at bat.
11 Display "Enter the player's number of times at bat."
12 Input atBat
13
14 // Calculate the batting average.
15 Set battingAverage = hits / atBat
16
17 // Display the batting average.
18 Display "The player's batting average is ", battingAverage
```

Program Output (with Input Shown in Bold)

```
Enter the player's number of hits.
150 [Enter]
Enter the player's number of times at bat.
500 [Enter]
The player's batting average is 0.3
```

2.8 Designing Your First Program

Figure 2-17 Flowchart for Program 2-15



2.8 Designing Your First Program

Summary

- Input
  - Determine data needed for input
  - Choose variables to store the input
- Process
  - Determine calculations to be performed
  - Choose variables to store the calculations
- Output
  - Determine what output the program will display
  - Usually the results of the program's calculations