Programming Logic & Design

Chapter 2 – Input , Processing & Output

Queens College

CSD 1133 – CPCM -2023S

Chapter 2 – Input , Processing & Output

Topics:

- Designing a Program
- Input, output and variables
- Variable Assignment and Calculations
- Variable Declarations and Data Types
- Named Constants
- Designing the first computer program

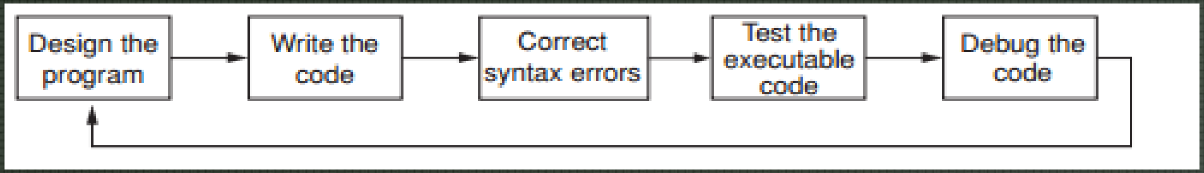
Chapter 2 – Input , Processing & Output

Designing a Program :

As we discussed earlier, programmers typically use high-level languages to write programs. However, all professional programmers will tell you that a program should be carefully designed before the code is actually written. When program -- - mmers begin a new project, they never jump right in and start writing code as the first step. They begin by creating a design of the program.
Each computer language has its own rules, known as syntax, that must be followed when writing a program. A language's syntax rules dictate things such as how key words, operators, and punctuation characters can be used. A syntax error occurs if the programmer violates any of these rules.
If the program contains a syntax error, or even a simple mistake such as a misspelled key word, the compiler or interpreter will display an error message indicating what the error is. Once all of the syntax errors and simple typing mistakes have been corrected, the program can be compiled and translated into a machine language program (or executed by an interpreter, depending on the language being used).

Chapter 2 – Input , Processing & Output

Once the code is in an executable form, it is then tested to determine whether any logic errors exist. A logic error is a mistake that does not prevent the program from running, but causes it to produce incorrect results.
If there are logic errors, the programmer debugs the code. This means that the programmer finds and corrects the code that is causing the error.
This entire process, which is known as the program development cycle, is repeated until no errors can be found in the program. Following figure shows the steps in the process.



This course focuses entirely on the first step of the program development cycle: designing the program. The process of designing a program is arguably the most important part of the cycle.

Chapter 2 – Input , Processing & Output

It is essential that you understand what a program is supposed to do before you can determine the steps that the program will perform. Typically, a professional programmer gains this understanding by working directly with the customer.

The programmer studies the information that was gathered from the customer during the interviews and creates a list of different software requirements. A software requirement is simply a single function that the program must perform in order to satisfy the customer .Once you understand the task that the program will perform, you begin by breaking down the task into a series of steps. This is similar to the way you would break down a task into a series of steps that another person can follow ;

- Example : **Task** → Calling a friend on the telephone
  Steps
- Pick up the phone and listen for a dial tone
- Press each digit of the phone number on the phone
- If busy, hang up phone, wait 5 minutes, jump to step 2
- If no one answers, leave a message then hang up

Chapter 2 – Input , Processing & Output

This is an example of an algorithm, which is a set of well-defined logical steps that must be taken to perform a task.
A programmer breaks down the task that a program must perform in a similar way. An algorithm is created, which lists all of the logical steps that must be taken. Of course, this algorithm isn't ready to be executed on the computer. The steps in this list have to be translated into code. Programmers commonly use two tools to help them accomplish this: **pseudocode and flowcharts**. Let's look at each of these briefly.

Pseudocode:

The word pseudo means fake, so pseudocode is fake code. It is an informal language that has no syntax rules, and is not meant to be compiled or executed. Instead, programmers use pseudocode to create models, or "mock-ups" of programs. Because programmers don't have to worry about syntax errors while writing pseudocode, they can focus all of their attention on the program's design. Once a *satisfactory design has been created with pseudocode, the pseudocode can be translated directly to actual code.*

Chapter 2 – Input , Processing & Output

For example, suppose you have been asked to write a program to calculate and display the gross pay for an hourly paid employee. Here are the steps that you would take:

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

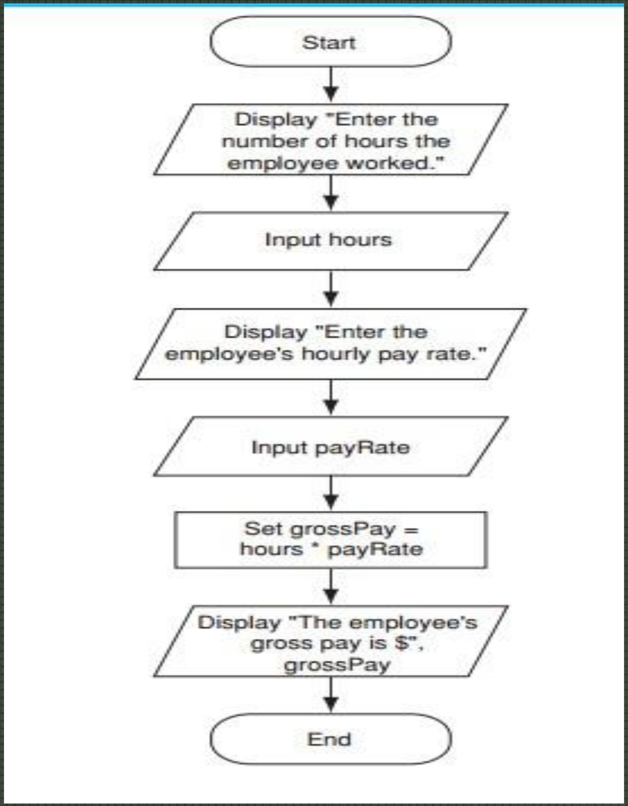
A sample pseudocode for the above pay calculating program is shown below

```
Display "Enter the number of hours the employee worked."
Input hours
Display "Enter the employee's hourly pay rate."
Input payRate
Set grossPay = hours * payRate
Display "The employee's gross pay is $", grossPay
```

Chapter 2 – Input , Processing & Output

Flow Charts:

Flowcharting is another tool that programmers use to design programs. A flowchart is a diagram that graphically depicts the steps that take place in a program. Figure below shows how you might create a flowchart for the pay calculating program we discussed earlier.



Start
Display "Enter the number of hours the employee worked."
Input hours
Display "Enter the employee's hourly pay rate."
Input payRate
Set grossPay = hours * payRate
Display "The employee's gross pay is $", grossPay
End

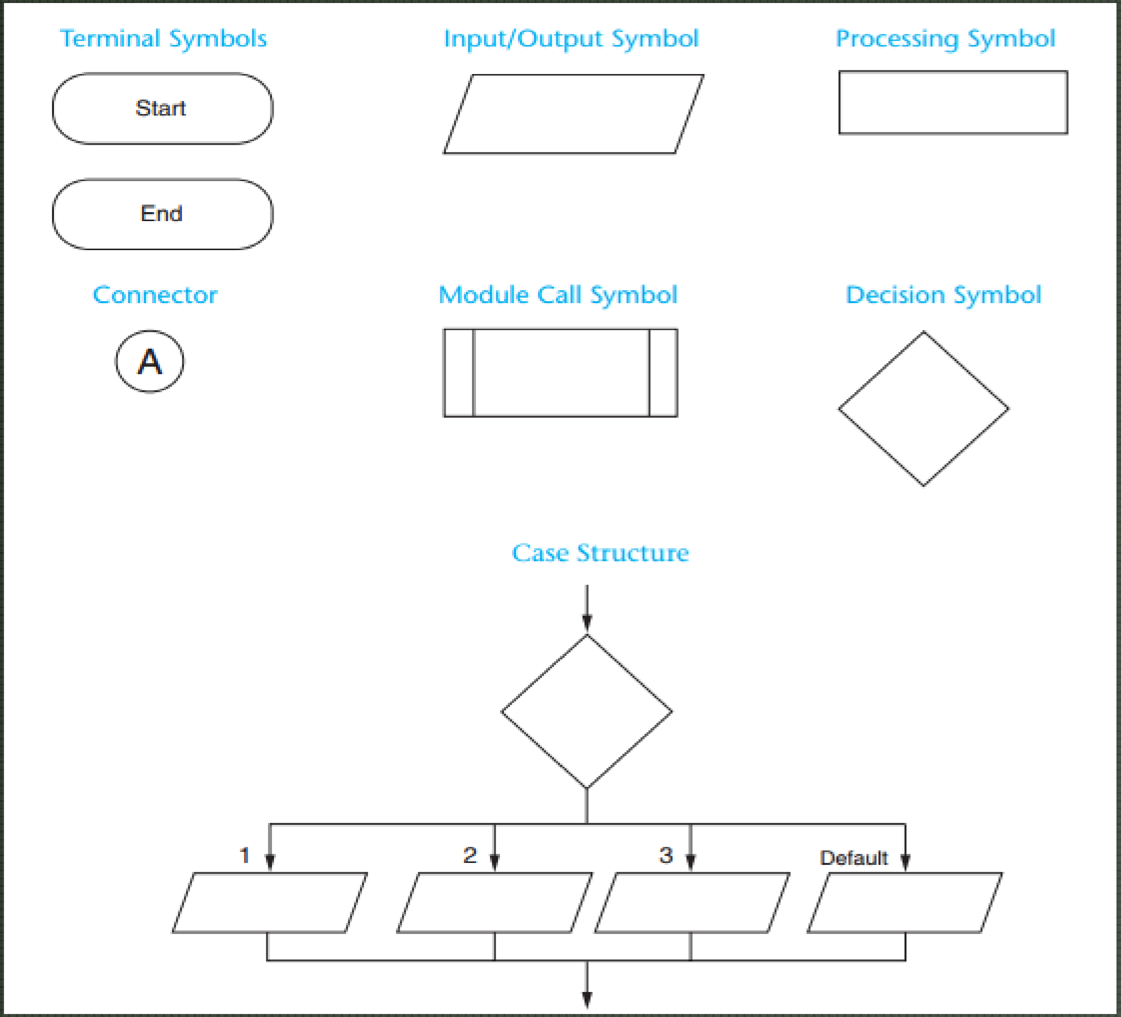
Chapter 2 – Input , Processing & Output

Notice that there are three types of symbols in the flowchart: ovals, parallelograms, and rectangles. The ovals, which appear at the top and bottom of the flowchart, are called terminal symbols. The Start terminal symbol marks the program's starting point and the End terminal symbol marks the program's ending point.
Between the terminal symbols are parallelograms, which are used for both input symbols and output symbols, and rectangles, which are called processing symbols. Each of these symbols represents a step in the program. The symbols are connected by arrows that represent the "flow" of the program. To step through the symbols in the proper order, you begin at the Start terminal and follow the arrows until you reach the End terminal

The disadvantage to drawing flowcharts by hand is that mistakes have to be manually erased, and in many cases, require that the entire page be redrawn. A more efficient and professional way to create flowcharts is to use software. There are several specialized software packages available that allow you to create flowcharts.

Chapter 2 – Input , Processing & Output

Flowchart symbols :



Chapter 2 – Input , Processing & Output

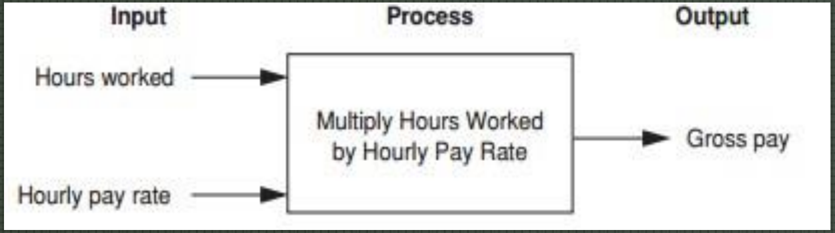
Checkpoint :

What is a software requirement?
What is an algorithm?
What is pseudocode?
What is a flowchart?

Output, Input and Variables :

Computer programs typically perform the following three-step process:
1. Input is received.
2. Some process is performed on the input.
3. Output is produced.

Input is any data that the program receives while it is running. One common form of input is data that is typed on the keyboard. Once input is received, some process, such as a mathematical calculation, is usually performed on it. The results of the process are then sent out of the program as output.



Chapter 2 – Input , Processing & Output

Perhaps the most fundamental thing that you can do in a program is to display a message on the computer screen. As previously mentioned, all high-level languages provide a way to display screen output. We use the word Display to write pseudocode statements for displaying output on the screen. Here is an example:

```
Display "Hello world"
```

Notice that after the word Display, we have written Hello world inside quotation marks. The quotation marks are not to be displayed. They simply mark the beginning and the end of the text that we wish to display

Task 1: How do you write a pseudocode program that displays your name and address on the computer screen.

```
Display “ John Doah”
Display “ 125, Whittal Road”
Display “ Toronto , ON   M1G 2V5”
```

Task 2: Can you draw a Flow chart for the above ?

Chapter 2 – Input , Processing & Output

Variables and Input

Programs use variables to store data in memory. A variable is a storage location in memory that is represented by a name. For example, a program that calculates the sales tax on a purchase might use a variable named tax to hold that value in memory

Next , we will discuss a basic input operation: reading data that has been typed on the keyboard. When a program reads data from the keyboard, usually it stores that data in a variable so it can be used later by the program. In pseudocode we will read data from the keyboard with the Input statement. As an example, look at the following statement:

```
Input hours
```

The word Input is an instruction to read a piece of data from the keyboard. The word hours is the name of the variable in which that the data will be stored.

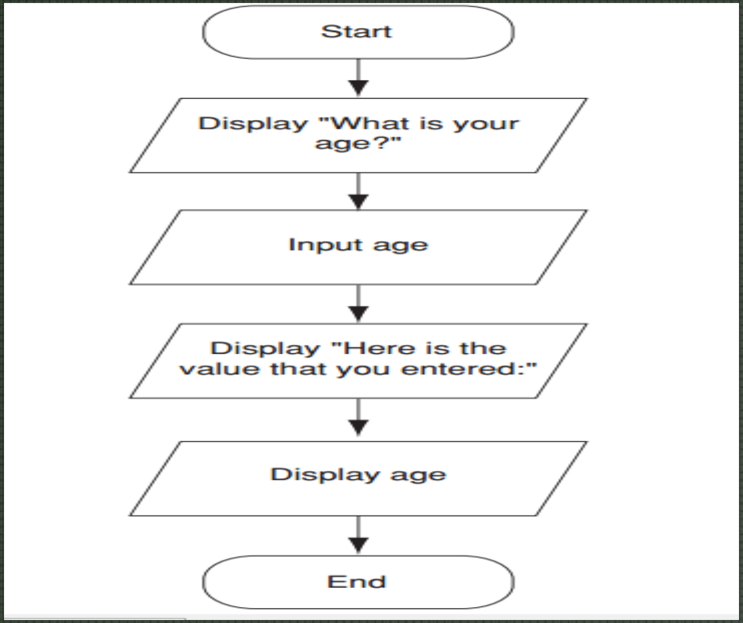
Pseudocode :

```
Display " What is your first name ?"
Input name
```

What Is the output of Display "name" ?

Chapter 2 – Input , Processing & Output

Flowchart


Start
Display "What is your age?"
Input age
Display "Here is the value that you entered:"
Display age
End

Variable Names:

All high-level programming languages allow you to make up your own names for the variables that you use in a program. Every language has its own set of rules that you must abide by when creating variable names.

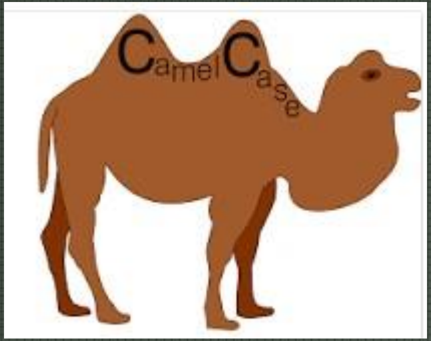
Although the rules for naming variables differ slightly from one language to another, there are some common restrictions:

Chapter 2 – Input , Processing & Output

● Variable names must be one word. They cannot contain spaces.
●In most languages, punctuation characters cannot be used in variable names. It is usually a good idea to use only alphabetic letters and numbers in variable names.
● In most languages, the first character of a variable name cannot be a number.
In addition to following the programming language rules, you should always choose names for your variables that give an indication of what they are used for

For example, a variable that holds the temperature might be named *temperature*
Since we cannot use space for variable names we can use underscore character to represent a space. For example, gross_pay is more readable than grosspay.
Another way to address this problem is to use the camelCase naming convention.
camelCase names are written in the following manner:
• You begin writing the variable name with lowercase letters.
• The first character of the second and subsequent words is written in uppercase.

grossPay
payRate
hotDogsSoldToday



Chapter 2 – Input , Processing & Output

When you are displaying multiple items on the screen, you want to separate those items with spaces between them. Most programming languages do not automatically print spaces between multiple items that are displayed on the screen.

```
Display "January ", "February ", "March"
```

Prompting the User

Getting keyboard input from the user is normally a two-step process:

1. Display a prompt on the screen.
2. Read a value from the keyboard.

A prompt is a message that tells (or asks) the user to enter a specific value. For example, the pseudocode shown below gets the user to enter his or her age with the following statements:

```
Display "What is your age?"
Input age
```

Checkpoint :

What is a variable?
Summarize three common rules for naming variables.
What two steps usually take place when a program prompts the user for input?

Chapter 2 – Input , Processing & Output

Variable Assignment and Calculations

In the previous section, we discussed how the Input statement gets a value typed on the keyboard and stores it in a variable. You can also write statements that store specific values in variables. The following is an example, in pseudocode:

```
Set price = 20
```

This is called an assignment statement. An assignment statement sets a variable to a specified value. In this case, the variable price is set to the value 20.

Variables are called "variable" because they can hold different values while a program is running. Once you set a variable to a value, that value will remain in the variable until you store a different value in the variable.

```
Set amount = 8000
Display " I have  ", amount ," in my account"
Set amount = 5000
Display " Now I have only  ", amount ," in my account"
```

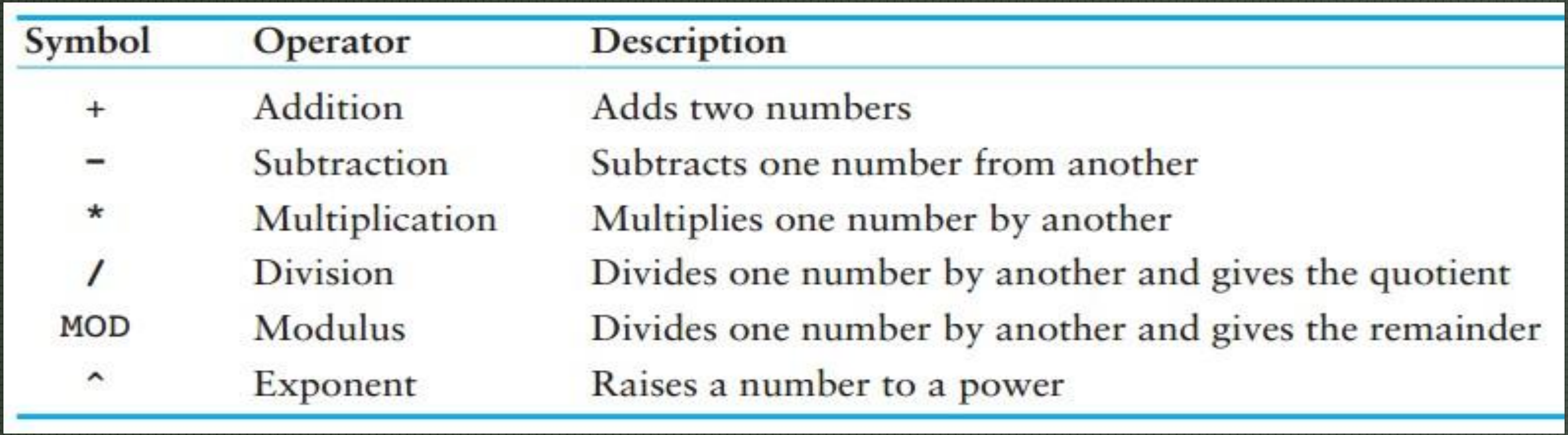
Explain why following statement is incorrect ?

```
Set 5000 = amount
```

Chapter 2 – Input , Processing & Output

Performing Calculations

Most real-world algorithms require calculations to be performed. A programmer's tools for performing calculations are math operators. Programming languages commonly provide the operators shown in following Table



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	Raises a number to a power

When we use a math expression to calculate a value, normally we want to save that value in memory so we can use it again in the program. We do this with an assignment statement.

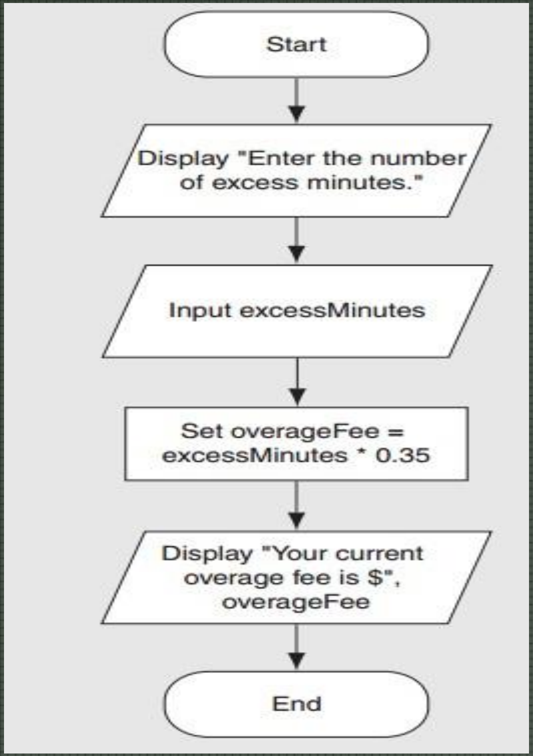
```
Set Salary = 3000
Set Tax = 420
Set   net_Salary   =   Salary   –   Tax
Display " Net salary is", net_Salary
```

Chapter 2 – Input , Processing & Output

Problem :

Suppose your cell phone calling plan allows you to use 700 minutes per month. If you use more than this limit in a month, you are charged an overage fee of 35 cents for each excess minute. Design a program that will simplify the task. You would like to be able to enter the number of excess minutes, and have the program perform the calculation for you.

Flowcharts


Start
Display "Enter the number of excess minutes."
Input excessMinutes
Set overageFee = excessMinutes * 0.35
Display "Your current overage fee is $", overageFee
End

Pseducode

```
Display " Enter the number of excess minutes"
Input excessMinutes
Set overageFee = excessMinutes * 0.35
Display " Your current overage fee is $ ", overageFee
```

Chapter 2 – Input , Processing & Output

Programming Task :
Suppose a retail business is planning to have a storewide sale where the prices of all items will be 20 percent off. Design a program to calculate the sale price of an item after the discount is subtracted. (Design the Pseducode and Flowchart)

The Order of Operations :
Consider the following statement:
Set outcome = 12 + 6 / 3
The outcome variable could be assigned either 6 or 14, depending on when the division takes place
In most programming languages, the order of operations can be summarized as follows:
1. Perform any operations that are enclosed in parentheses.
2.Perform any multiplications, divisions, or modulus operations as they appear from left to right.
3. Perform any additions or subtractions as they appear from left to right

12+6/3 → 14 (12+6)/3 →6

Chapter 2 – Input , Processing & Output

Programming Task :
Suppose you want to deposit a certain amount of money into a savings account, and then leave it alone to draw interest for the next 10 years. At the end of 10 years you would like to have $10,000 in the account. How much do you need to deposit today to make that happen? You can use the following formula to find out:

$$P = \frac{F}{(1+r)^n}$$

The terms in the formula are as follows:
● P is the present value, or the amount that you need to deposit today.
● F is the future value that you want in the account. (In this case, F is $10,000.)
● r is the annual interest rate.
● n is the number of years that you plan to let the money sit in the account.
It would be nice to write a computer program to perform the calculation
(Design the Pseducode and Flowchart)

The ^ symbol is commonly used as the exponent operator, and its purpose it to raise a number to a power.

Chapter 2 – Input , Processing & Output

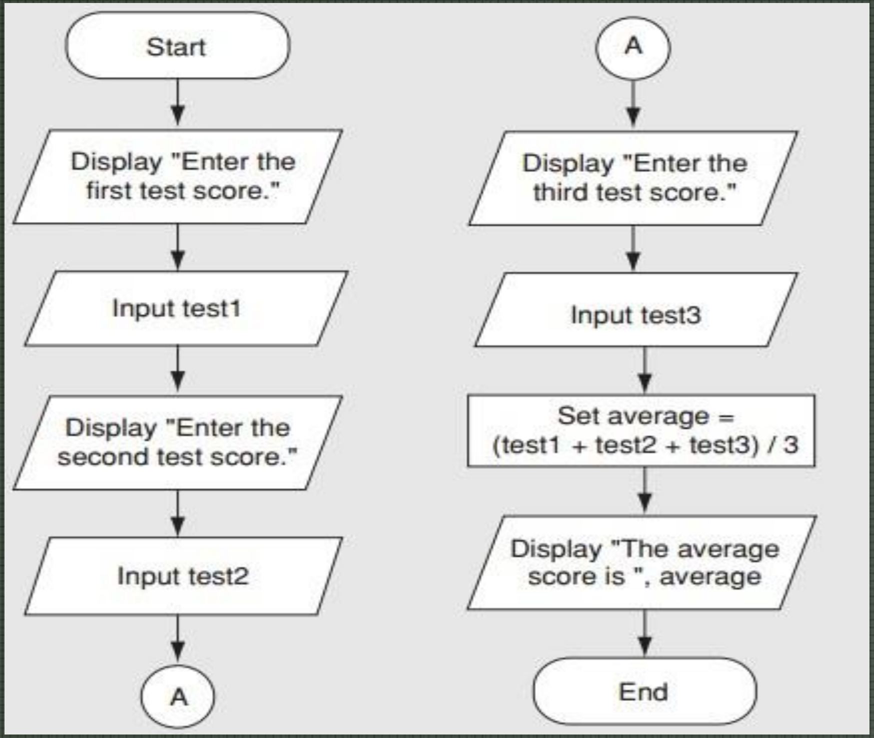
Problem :
Suppose you have taken 3 tests in your computer programming class, and you want to write a program that will display the average of the test scores.

Solution:

Pseducode

```
Display “Enter the first test score.”
Input test1
Display “Enter the second test score.”
Input test2
Display “Enter the third test score.”
Input test3
Set average = (test1+test2+test3) / 3
Display “ The average score is “, average
```

Flowchart


Start
Display "Enter the first test score."
Input test1
Display "Enter the second test score."
Input test2
A
A
Display "Enter the third test score."
Input test3
Set average = (test1 + test2 + test3) / 3
Display "The average score is ", average
End

Chapter 2 – Input , Processing & Output

Variable Declaration and Data Types:
Most programming languages require that you declare all of the variables that you intend to use in a program. A variable declaration is a statement that typically specifies two things about a variable:
● The variable's name
● The variable's data type
A variable's data type is simply the type of data that the variable will hold. Once you declare a variable, it can be used to store values of only the specified data type. In most languages, an error occurs if you try to store values of other types in the variable.
In this course we will use only three data types when we declare variables: Integer, Real, and String. Here is a summary of each:
●A variable of the Integer data type can hold whole numbers. For example, an Integer variable can hold values such as 42, 0, and –99. An Integer variable cannot hold numbers with a fractional part, such as 22.1 or –4.9.
●A variable of the Real data type can hold either whole numbers or numbers with a fractional part. For example, a Real variable can hold values such as 3.5, –87.95, and 3.0.

Chapter 2 – Input , Processing & Output

● A variable of the String data type can hold any string of characters, such as someone's name, address, password, and so on.

we will begin variable declarations with the word Declare, followed by
a data type, followed by the variable's name.

Here is an example:
Declare Integer length
A variable declaration statement typically causes the variable to be created in memory. For this reason, you have to write a variable's declaration statement before any other statements in the program that use the variable.

When you declare a variable, you can optionally assign a value to it in the declaration statement. This is known as initialization. For example, the following statement declares a variable named price and assigns the value 49.95 to it:
Declare Real price = 49.95
An *uninitialized* variable is a variable that has been declared, but has not been initialized or assigned a value. Uninitialized variables are a common cause of logic errors in programs.
Declare Real dollars
Display "I have ", dollars, " in my account.

Chapter 2 – Input , Processing & Output

when dividing an integer by another integer. In many programming languages, when an integer is divided by an integer the result will also be an integer. This behavior is known as integer division.

Set number = 3 / 2

This statement will store 1 in the number variable, not 1.5.

(If you are using a language that behaves this way and you want to make sure that a division operation yields a real number, at least one of the operands must be a real number or a Real variable)

Checkpoint:

What two items do you usually specify with a variable declaration?
What is variable initialization?
Do uninitialized variables pose any danger in a program?

Named constant

is a name that represents a value that cannot be changed during the program's execution. The following is an example of how we will declare named constants in our pseudocode:

Constant Real INTEREST_RATE = 0.069

Chapter 2 – Input , Processing & Output

Designing our first Program:

In this section we will solve a simple problem, go through the process of analyzing the program’s requirements, and design the algorithm in pseudocode and a flowchart.

Here is the programming problem:
In baseball, batting average is commonly used to measure a player’s batting ability. You use the following formula to calculate a player’s batting average:

Batting Average = Hits ÷ Times at Bat

In the formula, Hits is the number of successful hits made by the player, and Times at Bat is the number of times the player was at bat
Program’s actions can typically be divided into the following three phases:
1. Input is received.

If we look at the batting average formula, we see that two values are needed to perform the calculation:
• The number of hits
• The number of times at bat
Because these values are unknown, the program will have to prompt the user to enter them.

Chapter 2 – Input , Processing & Output

2. Some process (such as a calculation) is performed on the input.

The batting average program will divide the number of hits by the number of times at bat. The result of the calculation is the player's batting average

3. Output is produced.

The output of the batting average program will be the result of the calculation, which is stored in a variable named battingAverage.

Pseudocode :

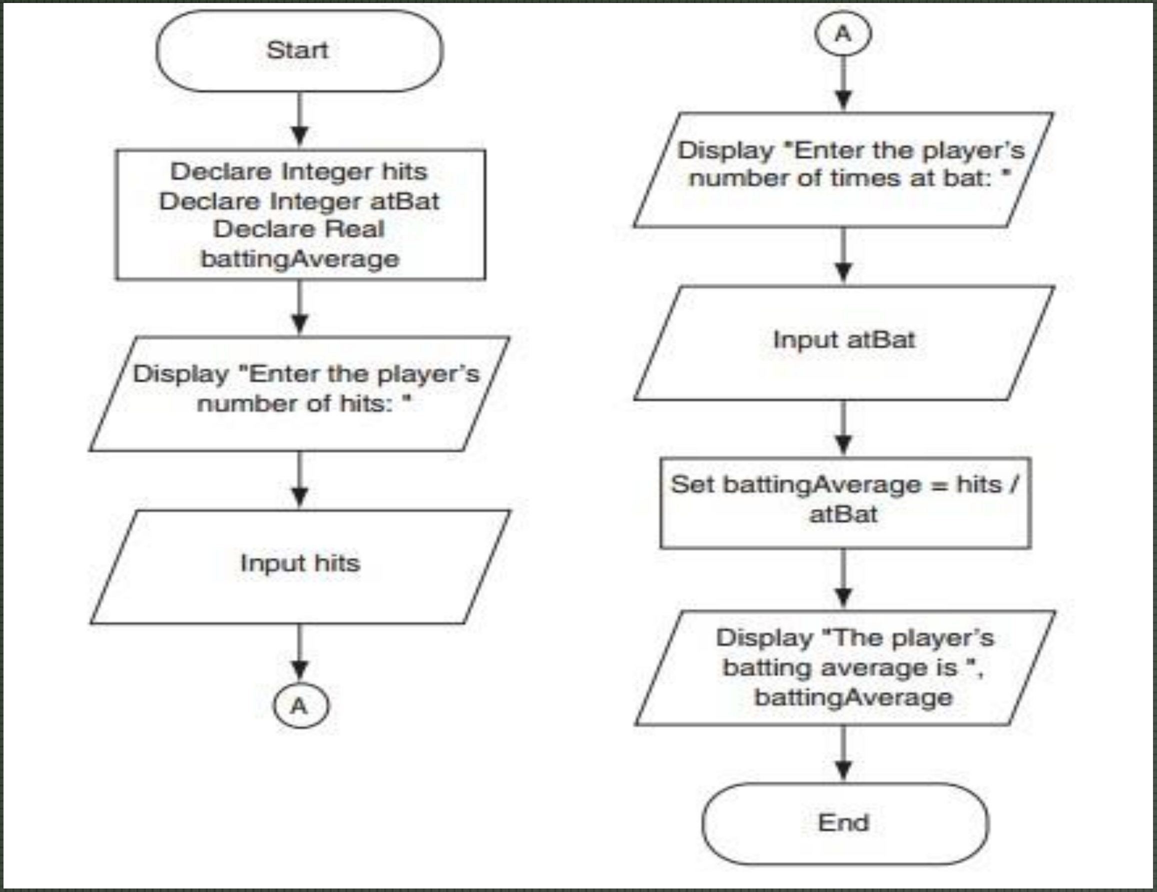
```
// Declare the necessary variables.
Declare Integer hits
Declare Integer atBat
Declare Real battingAverage
```

Chapter 2 – Input , Processing & Output

```
// Get the number of hits.
Display "Enter the player's number of hits."
Input hits

// Get the number of times at bat.
Display "Enter the player's number of times at bat."
Input atBat

// Calculate the batting average.
Set battingAverage = hits / atBat

// Display the batting average.
Display "The player's batting average is ", battingAverage
```

Chapter 2 – Input , Processing & Output

Flowchart :



Chapter 2 – Input , Processing & Output

Programming Exercises : (Pseudocode /Flowchart)

1. Sales Prediction

A company has determined that its annual profit is typically 23 percent of total sales. Design a program that asks the user to enter the projected amount of total sales, and then displays the profit that will be made from that amount.
Hint: Use the value 0.23 to represent 23 percent.

2. Sales Tax

Design a program that will ask the user to enter the amount of a purchase. The program should then compute the state and county sales tax. Assume the state sales tax is 4 percent and the county sales tax is 2 percent. The program should display the amount of the purchase, the state sales tax, the county sales tax, the total sales tax, and the total of the sale (which is the sum of the amount of purchase plus the total sales tax).
Hint: Use the value 0.02 to represent 2 percent, and 0.04 to represent 4 percent.

3.Tip, Tax, and Total

Design a program that calculates the total amount of a meal purchased at a restaurant. The program should ask the user to enter the charge for the food, and then calculate the amount of a 15 percent tip and 7 percent sales tax. Display each of these amounts and the total.

Chapter 2 – Input , Processing & Output