

CHAPTER ONE

A STRUCTURED APPROACH TO PROGRAMMING

LEARNING OBJECTIVES

- * Understanding of the importance and implication of the knowledge worker and the systems analyst in the Information Technology Revolution.
- * Understanding of the basic structure and working of a microcomputer system.
- * Knowledge and understanding of the need and importance for the Systematic Approach to programming and the Program Development Cycle.

INTRODUCTION

THE INFORMATION TECHNOLOGY REVOLUTION

The Information Technology (IT) revolution is claimed to be the second phase of the industrial revolution. The IT revolution has created many radical changes in the industrialised nations. In some case, these changes have resulted in a transformation to a post-industrial service oriented society. Approximately 90% of UK labourers were employed in Agriculture before the industrial revolution. The shift from an Agricultural to an industrial society resulted in a drastic reduction in the percentage of people employed in Agriculture to around 10%. Similar changes in the movement of lifetime occupation has taken place with the IT revolution. The use of IT in industry and commerce results in higher levels of productivity and hence per capita income. The United States is the principal country that leads in the development and application of IT in Commerce and Industry. The United States is being transformed from an industrialised society to a service oriented society where the Knowledge Worker plays a Key role in the level of productivity and in the generation of new wealth. The graph in figure 1-1 illustrates the trend in the distribution of workers in the USA employed in Agriculture, Goods production, and the Service industry since 1860.

TRENDS IN EMPLOYMENT DISTRIBUTION

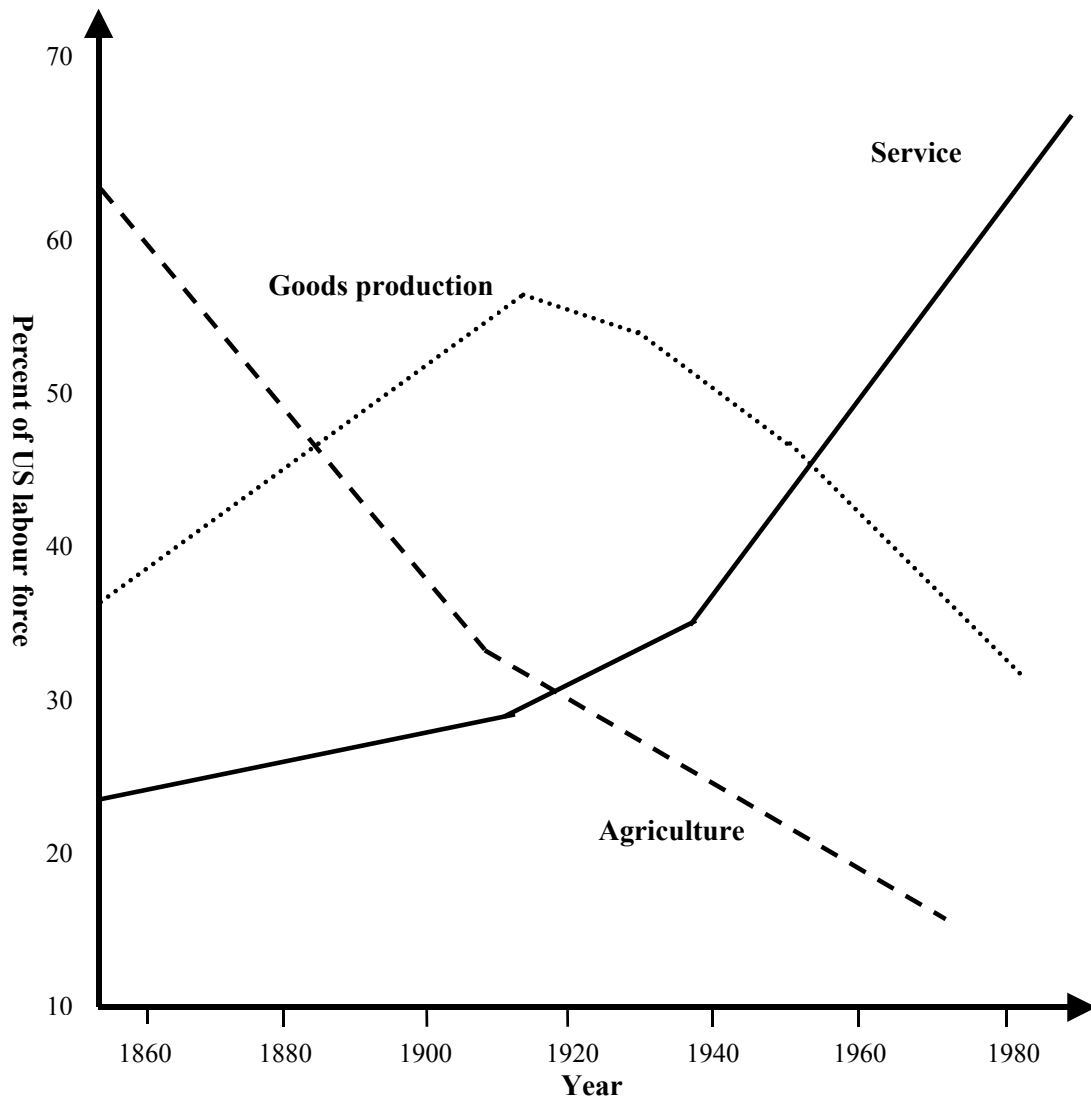


Figure 1-1

The use of Information Technology have resulted in rapid changes in the United States from an Industrial, Goods producing to a post industrial, service oriented society. Approximately two third of the labour force was employed in the service industries by 1976, with 50% of the labour force engage in information service occupations. These occupations include the manufacture, sales, and services of Computers and other related services.

THE BUSINESS SYSTEM

The development and application of commercial and industrial IT base systems usually require the service of the **SYSTEM ANALYST**. A system analyst is a professional person who analyses the information problem of a business or organisation system and then synthesises new systems to solve those problems or to meet other information needs. The definition for a system is: A combination of resources working together to convert some input resource to useful output. A Business system converts data into information and includes the following resources: Personnel, Facilities, Materials and Equipment.

Business System



Some Business Information Systems (BIS) are large systems composed of smaller sub-systems. These sub-systems can themselves be composed of smaller sub-systems. For example; A retailing business can consist of the following sub-systems:

- (a) Marketing
- (b) Finance
- (c) Product Development and
- (d) Administration

Each of these sub-systems can themselves be composed of smaller sub-systems.

(a) Marketing
Sub-systems
Sales
Distribution

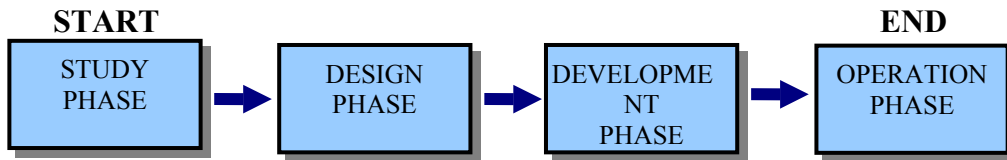
(b) Finance
Sub-systems
Billing
Collection
Paying

(c) Product Development
Sub-systems
Purchasing
Receiving
Inventory
Production

(d) Administration
Sub-Systems
Personnel
Contracts

THE SYSTEMS DEVELOPMENT LIFE CYCLE

Systems Analysis has evolved as a structured process for the development of IT systems. The development of a business information system must pass through a sequence of necessary activities. These activities consist of four phases called the **SYSTEMS DEVELOPMENT LIFE CYCLE**.



Each phase of the system development life cycle can in turn be composed of one or more cyclical sub-phases. The characteristic systematic approach to solving complex IT problem by way of the principle of life cycle phases is a fundamental Engineering Principle. This principle is applicable at all levels of abstraction for very complex systems. Most Systems Analyst begins their career as programmers. It is very important that serious considerations is given to this systematic and discipline approach to programming if you are considering a career as a professional programmer. This systematic and discipline approach should be supplemented with a fundamental and basic knowledge of computer systems and their operation. We call this systematic approach to programming: **THE PROGRAM DEVELOPMENT LIFE CYCLE**. This is explained further on in the chapter.

THE COMPUTER SYSTEM

A computer can best be defined as: Any machine or device that will accept an input amount of data, process it under the control and direction of a program and then output the process result to some other device or medium. Hence, a computer is made out of the basic elements illustrated below.



Computer systems can be classified in terms of

- (a) Type
- (b) Size
- (c) Generation

(a) There are three basic types of computers:

- (i) Analogue
- (ii) Digital and
- (iii) Hybrid

An analogue computer will only accept, process and output data in an analogue form. An analogue signal is a signal with magnitude that varies continuously with time. Figure 1-2 illustrates an example analogue signal.

AN EXAMPLE ANALOGUE SIGNAL

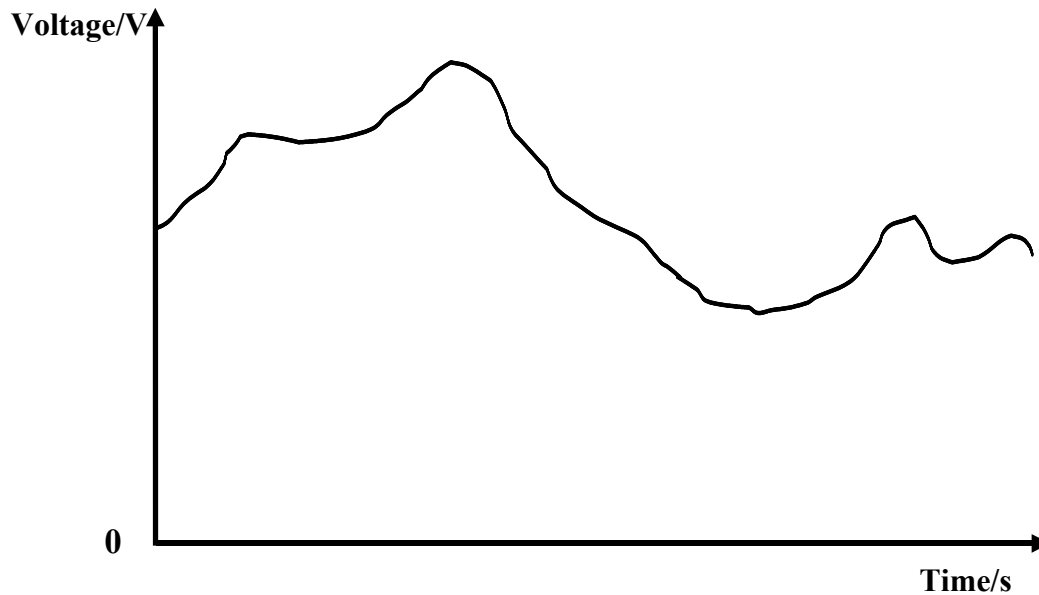


Figure 1-2
The amplitude of an analogue signal varies continuously with time.

In the case illustrated in figure 1-2, the signal is an electrical voltage or current. Therefore, the voltage varies continuously with time. I.e. there is no sudden break in the magnitude of the signal at any instant in time.

A digital computer can only accept, process and output data in a digital form. A digital signal is a signal with magnitude that can vary discontinuously between two fixed voltage levels at fixed time intervals. A digital signal is used for representing data and instructions on a digital computer system. The digital signal affects the bistable devices that make up the electrical system of the computer.

AN EXAMPLE DIGITAL SIGNAL

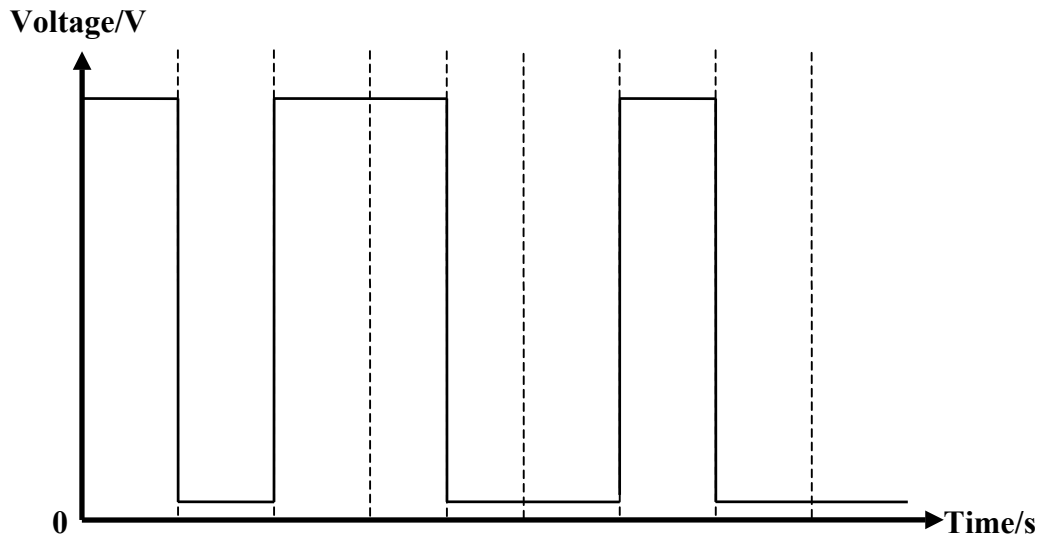


Figure 1-3
A digital signal varies discontinuously at regular time intervals.

A hybrid computer is made out of digital and analogue sub-systems. They are most used for research or in process control systems. In this text, we are only concern with the digital computer system, mainly the microcomputer system.

WHAT IS DATA?

The operations performed by a computer require data. Data values are represented as digital signals like that illustrated in figure 1-3. Data is the representation of facts, concepts, or instructions in a structured and logical manner suitable for processing, communication, and interpretation by humans or some programmed device. Examples of data are: The balance of a bank account, The score a student receive on a test, or The list of all male students doing a particular subject at a college. The purpose of a computer is to accept and process data to produce useful information for human use. In the context of a digital computer system, we define data as Information in a form that can be used by a computer program.

MULTIPLE OPERATIONS AND PRECEDENCE

A combination of arithmetic operators can be used to implement multiple operations in a single LET instruction. In figure 3-10, the addition and subtraction operators are used to determine the current balance on a savings account. The current balance is calculated by adding the deposit **D** to the old balance **O** and then subtracting the withdrawals **W**. The current balance is stored in the variable name **N**.

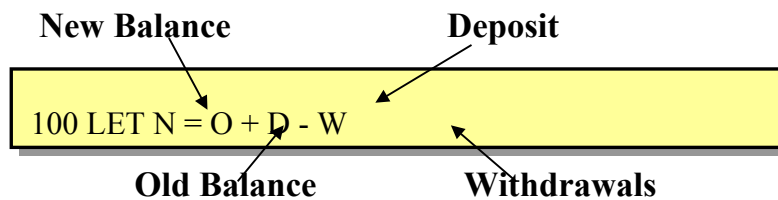


Figure 3-10

The operation of Subtraction and Addition is performed in a single arithmetic expression. Multiplication and division can also be included in a single expression.

If an expression is written with multiple arithmetic operators, each operator is executed in the order of precedence assigned to it and not necessarily the order in which it occurs in the expression. As an example, the arithmetic expression $Y + B^4$ is evaluated by first raising the value in **B** to the power of 4. The result is then added **Y**. If it is required to add three values stored in the variable names **S1**, **S2** and **S3** and then divide the result by 3, an expression like that in figure 3-11 will not produce the correct result.

140 LET A = S1 + S2 + S3/3

Figure 3-11

Because Division carries the highest order of precedence in this expression, when the expression is evaluated, the correct average of the values in S1, S2 and S3 is not calculated.

In this example, the value stored in **S3** is first divided by 3. The result is then added to the sum of **S1** and **S2**. For example, if the value stored in **S1** is 78, the value stored in **S2** is 76 and the value stored in **S3** is 72, the result in **A** is $(78 + 76 + 72/3) = 178$. This is incorrect. To write expressions that overrule the order of precedence for an operator require the use of brackets. Figure 3-12 illustrates the correct expression for calculating the average of the values stored in **S1**, **S2** and **S3**.

```
140 LET A = (S1 + S2 + S3)/3
```

Figure 3-12

The use of brackets in this expression causes the values in S1, S2 and S3 to be added together first. The result is then divided by 3 to give the correct value.

In the correct expression, as illustrated in figure 3-12, the values in **S1**, **S2** and **S3** are first added together. The result is then divided by 3 to determine the correct average.

USING ARITHMETIC RESULTS FROM AN EXPRESSION

The end result of an arithmetic expression in a given instruction can be used in subsequent instructions in a BASIC program. Figure 3-13 illustrates an example of this.

```
160 LET T1 = C1 + C2
170 LET T2 = T1 + M3
180 PRINT T1, T2
```

Figure 3-13

The value from the calculation at line 160 is stored in T1. The variable T1 can be used in any other instruction such as at line 170 and 180 where a numeric variable is allowed.

In figure 3-13, the value calculated at line 160 is stored in the variable name T1. The value in T1 is then used at line 170 to do another calculation. The result from this calculation is stored in the variable name T2. The value stored in the variable names T1 and is printed by the PRINT instruction at line 180.

ROUNDING WITH THE INT FUNCTION

Consider a retail operation in which tax is calculated at 8%. If for example, a sale value of 45.68 is made by a customer, then the sales tax is calculated as follows: **$45.68 \times 0.08 = 3.6544$**

Because this result should be expressed as a decimal value with two digits in the decimal fraction, the result need to be rounded to two digits to the right of the decimal point. Therefore, all digits that follow the second digit after the decimal point should be truncated. Adding a 1 to the 2nd digit after the decimal point if the 3rd digit after the decimal point is greater than or equal to 5 does

this. This is called rounding up. A 0 is added to the 2nd digit after the decimal point if the 3rd digit after the decimal point is less than 5. This is called rounding down. For example, the value of 3.6544 is rounded down to 3.65.

The BASIC interpreter does not provide an explicit facility for rounding numbers. Therefore, coded instruction must be written that will round off numbers to the appropriate number of digits after the decimal point. The following steps illustrate how this can be done.

Consider the calculation $34.874/10 = 3.4874$. This is to be rounded to two digits after the decimal point.

STEP 1: Add **.005** to **3.4874** to get a carry from the **3rd** digit after the decimal point. This carry is added to the **2nd** digit to the right of the decimal point. The carry will always be **0** or **1**. In this case, $3.4874 + .005 = 3.4924$. No carry occurs if the **3rd** digit after the decimal point is less than **5**.

STEP 2: Shift the decimal point two positions towards the right by multiplying the result by **100**: $3.4924 \times 100 = 349.24$

STEP 3: Truncate all digits that follow the decimal point.

$349.24 \longrightarrow 349$

This can be done by use of the INT function. This is illustrated below.

STEP 4: The last step is to shift back the decimal point two positions towards the left by dividing the result by **100**.

$349/100 = 3.49$. The end result is now the desired rounded value having two digits after the decimal point.

BASIC provides functions that you can use in an expression in a similar way to which you use variable names. Therefore, we can use the INT function in a single instruction to carry out the operation of rounding. The code in figure 3-14 illustrates this.

```
290 LET T = INT((T + .005)*100)/100
```

Figure 3-14

All the steps necessary to round a number is implemented in a single instruction. Brackets used in the expression determine the sequence of these operations.

BINARY ARRAY SEARCH ALGORITHM

The array search given above is done in a sequential manner. In this method, the first element in the array is examined, then the second element is examined followed by the third element and so on. This array search method is useful when the number of element in the array is relatively small and the data values are not sorted. When an array with a large number of elements is to be search, a faster search technique should be used. In this case, if the data values in the array are arrange in ascending or descending order, then the binary search technique can be used.

Portions of the array that can not contain the data value been searched for is eliminated when the binary search technique is used. Figure 7-18 illustrates an example flight number array that stores 11 data values. The data values are place in ascending order.

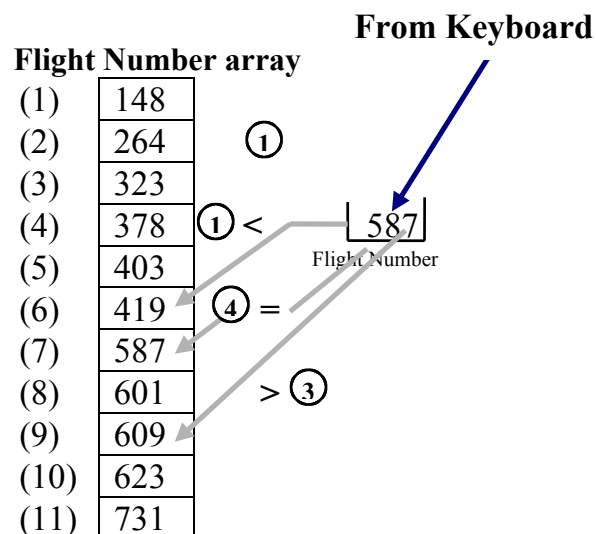


Figure 7-18

The values in an array must first be placed in ascending or descending order before the binary search can be used. The search is done by iteratively eliminating portions of the array in which the data value can not be found until the equal data value is found.

The first element of the array stores a flight number that is less than the flight number in the second element of the array. The second element of the array stores a flight number that is less than the flight number in the third element of the array and so on. The steps involve during the binary search are:

1. The user enters the flight number to be searched for. In this example the user enters 587.

2. The flight number stored in the middle element of the array is compared to the user flight number. The flight number in the middle element of the array, is less than the user flight number. Therefore, since all the flight numbers are arranged in ascending sequence, the flight numbers stored in the array elements 1 to 5 are also less than the user flight number. Therefore, there is no need to compare the data values in these elements of the flight number array. By this comparison technique, it is unnecessary to compare more than half the data values in the array.

3. At this point, the user flight number is compared to the flight number in the middle element of the remaining half of the array. The middle element between 7 and 11 is 9. Therefore, in the second comparison the user flight number is compared to flight number 609 in the 9th element of the array. Flight number 587 is less than 609. Therefore, only element 7 or element 8 can store the flight number that is equal to the user flight number. This is because the user flight number is already found to be greater than the flight number in the 6th element and is less than the flight number in the 9th element.

4. Since there is no precise middle value between 6 and 9, the lower of the two values (7 and 8) is chosen. Therefore, in the next comparison the user flight number is compared to the flight number in the 7th element of the array. The flight number stored in the 7th element is equal to the user flight number therefore, the array search is terminated. In this example, only three comparisons are required to find the equal flight number. Seven comparisons would be needed to locate the required element if a sequential search was done. More saving on search time is obtained in arrays that have a larger number of elements. In an array with 1,000 elements, the maximum number of comparisons necessary to locate a particular data value is 110.

LOGIC FOR THE BINARY SEARCH

The logic used in the binary search calculates the subscript value of the next element in the array that is to be checked on each pass of the search loop. The search continues until all elements in the array are examined and no equal value is found. The flowchart in figure 7-19 illustrates the logic used to implement the binary search. Further explanation of the steps involved in the search for user flight number 587 is given below.

1. The first upper limit used to calculate the first search subscript value is set to 12. This upper limit is obtained by adding 1 to the number of elements in the array. The lower limit is set to zero.

2. The search subscript used in each pass of the search is calculated by adding the upper and lower limits and then dividing the result by 2. In the first pass the search subscript is calculated as $(12 + 0)/2 = 6$. Therefore, the value in the 6th element is the first value that is checked.

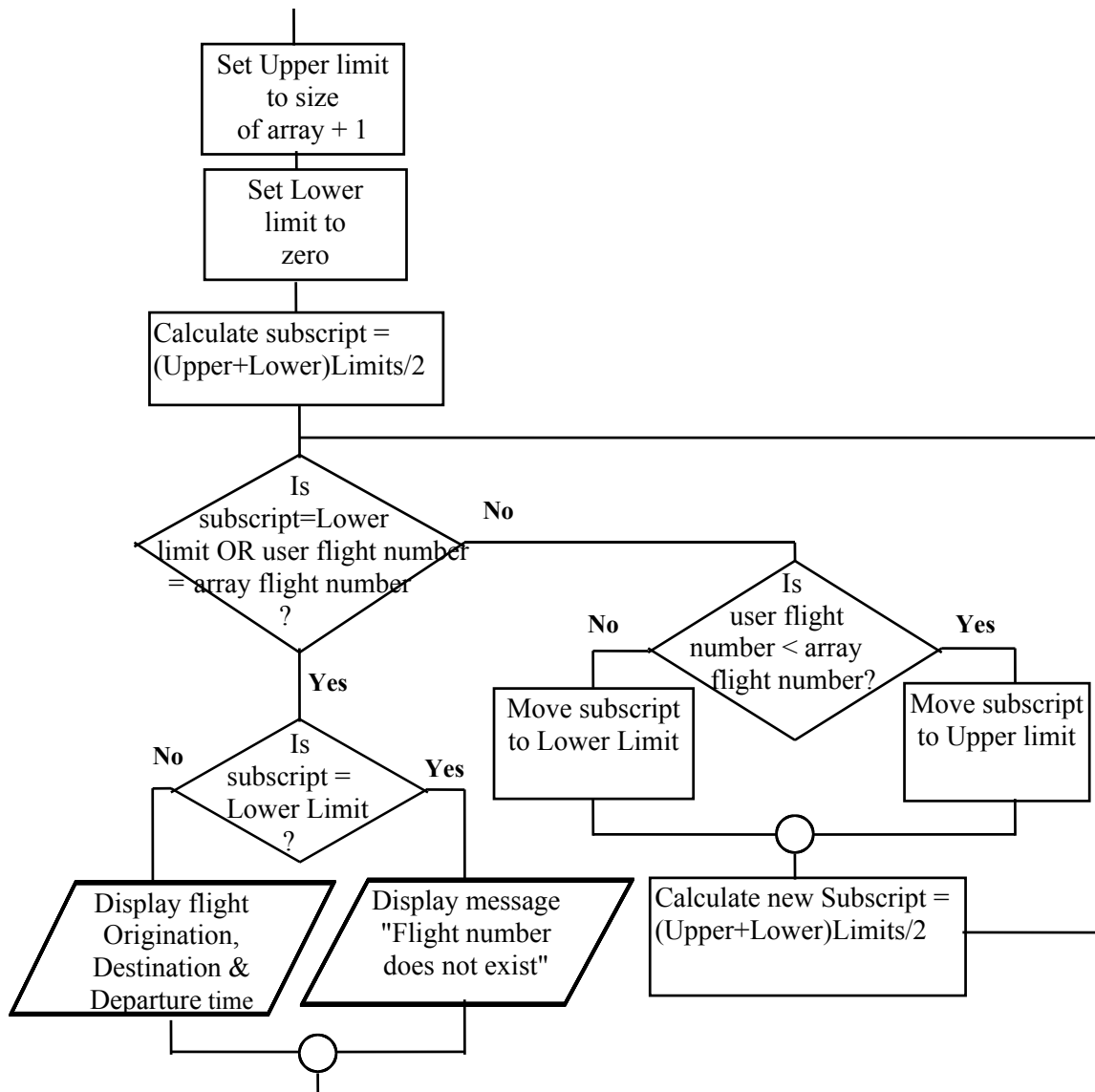


Figure 7-19

In the binary search, a loop is used to perform the search and the IF-THEN logic is used to determine the outcome of the search. This is the same basic control structure used in the linear search.

3. The calculated subscript in step 2 is check against the lower limit and the user flight number is compare to the flight number in the array element at the calculated subscript. If the calculated subscript is not equal to the lower limit and the user flight number is less than the flight number in the element reference by the subscript value, the calculated subscript value is moved to the

variable that stores the lower limit subscript. If the user flight number is greater than the flight number in the array element with the calculated subscript, the calculated subscript is moved to the variable that stores the upper limit subscript. In this case, when the first search is done, the calculated subscript is not equal to the lower limit. The user flight number; 587, is not equal to the data in the 6th element. Therefore, a check is done to determine if the user flight number is less than the flight number in the 6th element of the flight array. Since 587 is greater than 419, the calculated subscript value is moved to the lower limit. Hence the lower limit is now set to 6 and the upper limit remains at 12.

4. The next subscript value is calculated by adding the values in the upper and lower limit variables and then dividing the result by 2. In this example, the value calculated is $(12 + 6)/2 = 9$. The subscript value 9 is used in the next comparison.

5. The processing in step 3 and step 4 are repeated. In this example, the subscript value 9 is not less than the lower limit and the user flight number is not equal to the flight number stored in the 9th array element. Therefore, the search loop instructions are executed once more. The user flight number is less than the flight number in the 9th array element therefore, the calculated subscript value; 9 is used to replace the value in the upper limit variable name. The lower limit value; 6 does not change. The value of the next search subscript is calculated by adding the upper and the lower limit values and then dividing the result by 2: $(9 + 6)/2 = 7.5$. The next search subscript value is set by extracting the integer portion of this result. Therefore, the next search subscript value is 7. The value stored in the 7th element of the flight array is 587. This is equal to the user flight number. Therefore, an exit is made from the loop at the end of the next search.

6. When an exit is made from the loop, it must be determine if an equal flight number has been found or if the user flight number does not exist. The user flight number does not exist when all element of the array is searched and no equal flight number is located. The entire array has been searched and no equal flight number is found whenever the value of the search subscript is equal to the lower limit. In this example, the value of the lower limit is not equal to the search subscript when exit from the loop occurs. Therefore, an equal flight number is found. Therefore, the value of the search subscript is used for extracting related data from other arrays for display.

The logic used for the binary search is different from that used for the sequential search but the fundamental principle used in both of them is the same. Elements in the array are searched by looping until an element that store an equal value is found or until the entire array is searched. At the end of looping it is determine if an equal value is found; if so, print the data values, otherwise print an error message.

CHAPTER SEVEN

QUESTIONS AND EXERCISES

1. An element of an array is addressed by specifying the name of the array. (True or False).
2. A FOR-NEXT loop is not used to place data values in an array because it violates single exit point rules for the loop logic structure. (T or F).
3. A binary search is faster than a sequential search. (T or F).
4. The steps involved in the searching of data values in an array are:
 - a) Create the array, load data into the array, and search the array.
 - b) Create the array, search the array, and calculate the subscript.
 - c) Create the array, load data into the array, and divide the array in half.
 - d) Load data into the array, search the array, and calculate the subscript.
5. DATA instructions:
 - a) Are used to load data values into an array.
 - b) Must be used in conjunction with the FOR-NEXT loop.
 - c) Define a list of data values to be reference by READ instructions.
 - d) Must be subscripted when being used to define data to be used by an array.
6. The Boolean OR operator can not be used in an IF-THEN instruction that test the size of the search subscript at the start of a search loop because:
 - a) It causes a syntax error.
 - b) It is too complex and violates good program design practices.
 - c) It may cause a subscript error, and the program will be terminated.
 - d) No such thing exist in the BASIC language.
7. A binary search is faster than a sequential search. The only stringent requirement in a binary search is:
 - a) The data values of the array must be arrange in ascending or descending sequence.
 - b) The computer system that is being used must use binary coding instead of octal or the hexadecimal.

- c) It can only be used for small arrays. Large arrays require too much memory and the search becomes inefficient.
 - d) It must be used for large arrays with 1000 or more elements.
8. The _____ instruction is used to reserve storage for an array.
 9. The _____ instruction and the _____ instruction are used to establish the search loop.
 10. When an array is searched, it is possible that the value that is searched for does not exist in the array. What step must be taken in the program to take account of this possibility?
 11. The air line company using the sample program in this chapter have requested that the program be modified to also show the arrival time of each flight. The arrival times for the cities used are:

Boston - 3:30 PM
Chicago- 2:40 PM
Atlanta- 5:45 PM
Seattle - 5:30 PM
Miami - 3:10 PM

Make the required changes to the sample program in this chapter to accomplish these modifications.

12. Four more cities are to be added to the flight schedule used in the sample program. The information is:

A hierarchy chart is used to display the relationship among a series of subroutines used in a program. The hierarchy chart that display the relationship between the major subroutines used in the sample program is given in figure 8-37.

THE PROGRAM HIERARCHY CHARTS

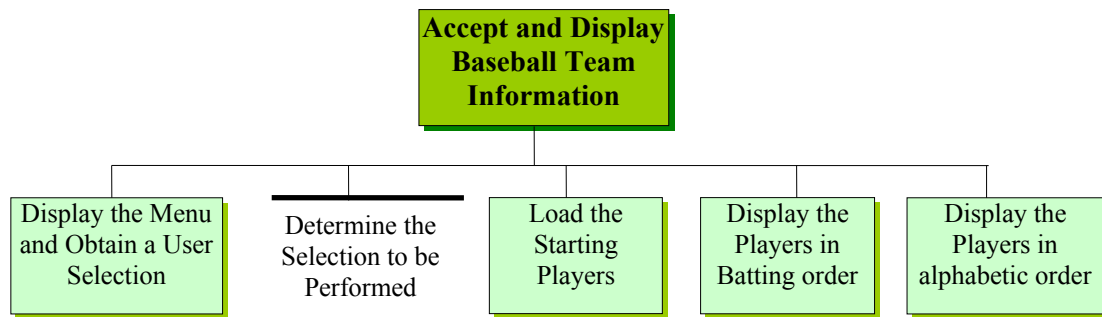


Figure 8-37

A hierarchy chart is a top-down design tool used to display the relationship between subroutines in a program. The tasks specify within the rectangles are performed in subroutines. The task below the horizontal line is not performed in a subroutine.

The task to be performed by each subroutine is written in a separate rectangle of the hierarchy chart. The overall program objective is written in the top-most rectangle of the hierarchy chart.

THE MAIN CONTROL LOGIC

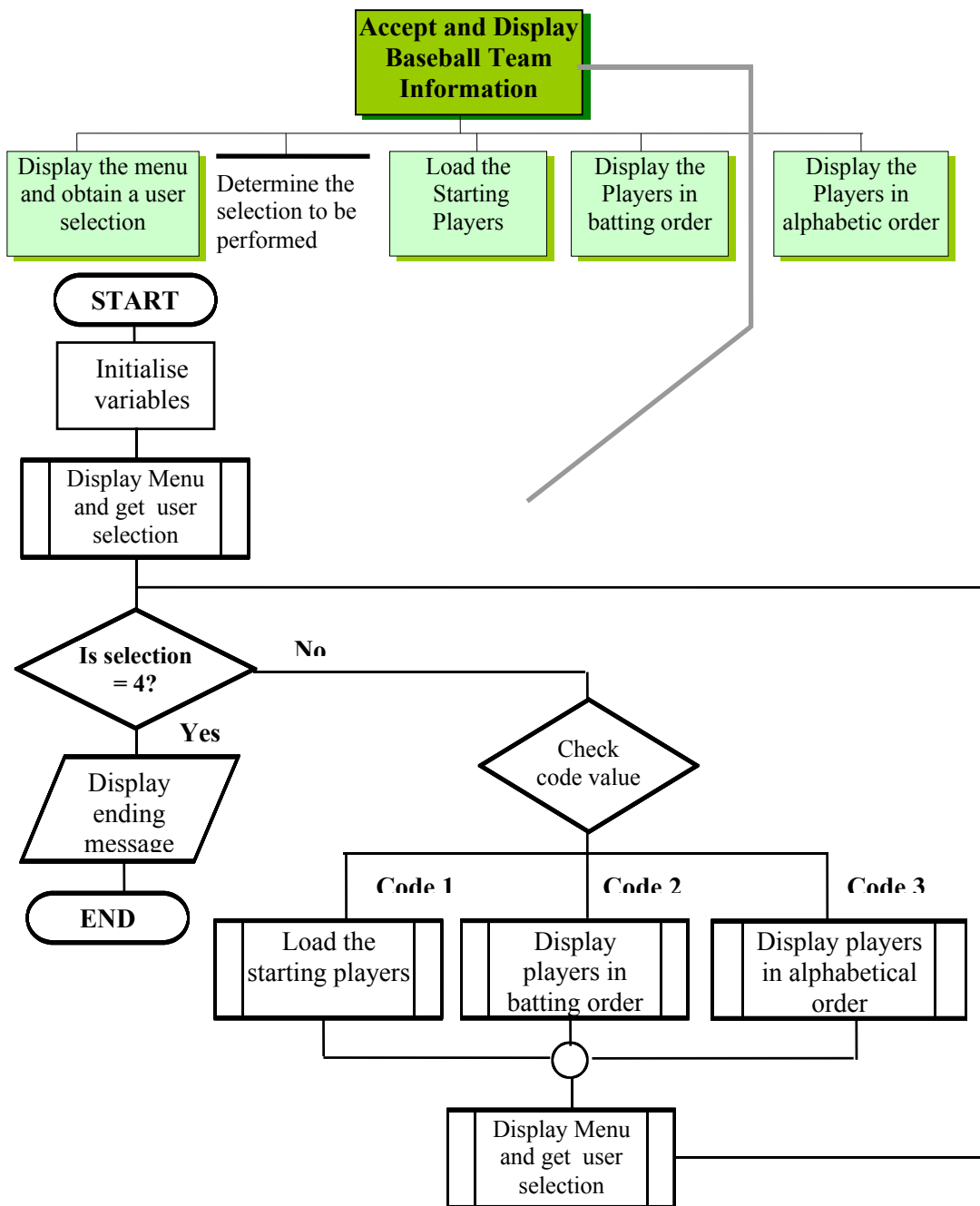


Figure 8-44

The logic of the topmost module is used to control the overall function of the program. It uses the multiple selection case structure to determine the processing requested by the user in response to the main menu. The rectangles with two vertical lines indicate that the specified tasks are subroutines in the program. Hence, four subroutines are called in the main control program.

Each module at the second and third levels of the hierarchy chart also have their own control logic. This control logic is illustrated in figure 8-45 to 8-49.