UNDERSTANDING THE ANALYSIS OF COMPUTERISATION

BY
ADEBOLA ENIOLORUNDA EZEKIEL
PGD/MCS/225/96

A PROJECT SUBMITTED TO THE DEPARTMENT OF MATHEMATICS/STATISTICS/COMPUTERS SCIENCE, FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA. IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF POST GRADUATES DIPLOMA IN COMPUTER SCIENCE.

MARCH 1998.

CERTIFICATION

This Project has been examined and found acceptable in partial fulfilment of the requirements for the Post-Graduate Diploma in Computer Science of the Department of Mathematics/Computer Science of the Federal University of Technology, Minna.

PROF. K. R. ADEBOYE
(SUPERVISOR)

DATE

PROF. K. R. ADEBOYE
(HEAD OF DEPARTMENT)

DATE

EXTERNAL EXAMINER

DATE

D E D I C A T I O N

This project is dedicted to God - the Creator of Heaven and Earth who through his infinite mercies and protection enabled me to accomplish this programme. I remain indelibly grateful to His guidance.

ACKNOWLEDGEMENT

This completion of this project made it obligatory for me to express my gratitude to God and to those who contributed in one way or the other to the completion of this Project.

I particularly acknowledge and remain grateful to my Supervisor and Head of Department of Mathematics/Computer Science, Dr. K. R. Adeboye for his invaluable and unrelenting reviews of the project work despite his numerous official schedules.

I also appreciate the contribution of all the teaching and non-teaching staff particularly Mr. Badmus, Dr. Yomi Aiyesimi Dr. S. A. Reju, Mallam Habib and others who imparted the knowledge and made lecture period very interesting.

I am indebtedly grateful to Adekunle Pius Akeke for his guidance, the materials and assistance he rendered for the timely completion of the Project.

My worthy commendation and appreciation is relayed also to my colleagues at work namely - Miss Charity Iheanacho, Israel A. Ibitola and Mr. Joshua O. Ikubaje who provided me cover at work and for their moral support throughout the programme.

I doff my cap for all of you and wish you God's abundant blessings in all your endeavours in life. Amen.

ABSTRACT

The act of computerization is central to the field of information system, yet little is really known about what computerization is or is operational techniques. For understanding of computerization to be developed, it is important that the cognitive processes and the basic analysis invoked in carrying out the computerization work be clearly stated.

There is a set of basic analysis that constitute computerization in the information systems field. This analysis serves as an initial vocabulary for describing what must be accomplished in creating an information system computerization.

Computerization must reconcile the impression of human activity with that of precisely specified operations required by a computer.

This project took an overview of computers, its characterizations, classifications and its basic components. This will in no small way aid our understanding of our analysis.

In addition, it also took cognisance of the systems lifecycle, analysis, design and system implementation and review. These will serve as an initial vocabulary for describing what must be accomplished in creating an information system computerization.

The program also examined program development generally. The essence is to give us a general overview of choice of programming language, fundamentals of programming and phases of programming development.

Finally, we were also congisant of the form of computer management. This is to enable us understand the viruses and their effects, the procedures and care and recommendations which could serve as a stepping stone for initial learners of computer language.

TABLE OF CONTENT

CONTENTS		PAGES

CHAPTER ONE

1 INTRODUCTION

1.1 OVERVIEW OF COMPUTERS

A computer is any machine which can accept data in a prescribed form, process the data and supply the results of the processing in a specified format as information or as signals to control automatically some further machine or process. This machine has no personality or intelligence of its own, anything it does is as a result of human instruction(s).

A computer is a problem-solving machine/device although it is not suitable for solving all problems. It is sometimes more economical to solve relatively simple problems that occur infrequently by some other mechanical or manual ones if, on the other hand, a problem

i) requires a large volume of input or out put.
ii) is repetitive in nature
iii) requires great processing, speeds and accuracy.

Computerisation is inevitable in business growth. Thus, to a corporate head, it may be worthwhile or even necessary to use computer when the following occurs.

i) When data volume is large
ii) High rate of routine jobs
iii) Time becomes vital
iv) Accuracy is important
v) Essential information about the scope of the business for developmental purposes.

Electronic computers are linked to a chain of calculating inventions that stretches back to pre-historic times. The development of tools to aid in calculation began with early civilisation. People first used sticks, stones, shells, notches on a stick, marks in the sand or knots in a rope to aid in counting. Later, fingers were used to perform simple calculations.

One of the earliest calculating devices created by man was the ABACUS. This ancient calculating instrument has been used for the past 2000 years.
In the 17th century (1617), John Napire developed an ingenious device for multiplication and division. The device called Napier's Bones was used for many years. In 1642, the first practical calculating machine was built by a 19 years old Frenchman called Blaise Pascal. Pascal calculator was limited to performing only addition and subtraction. About 30 years later, in 1671, a German Mathematician named Leibniz developed a similar machine that could also multiply and divide.

The first machine to perform basic arithmetical operations well enough for commercial use was the ARITHMOMETER built by Charles Xavier Thomas in 1820. Charles Babbage, an English Methematician designed the ANALYTICAL Engine in 1833 which was the forerunner of the modern digital computer.
In the 19th century, several key-driven machines were developed, including the first commercially practical adding machine invented in 1884 by William Burroughs. It was not until about 20 years after Charles Babbage death that the use of punched cards was applied to data processing. Cards had been used earlier to control patterns in textile looms. In 1887, Herman Hollerith, an employee of the U.S. Bureau of the census used punched card equipment to process the 1890 census.

In 1907, James Powers developed a punched card system to process the 1910 census. These machines were forerunners of electromechanical data processing systems. Between the years 1937 - 1944, Howard Aiken of Howard University, led a group of engineers to design the ASCC (Automatic Sequence Controller Calculator) which was also called Mark 1.
This large machine used a program to guide it through a long series of calculations. By 1941, Konard Zuse had completed three relay calculators that incorporated many of the ideas of automatic computing. In 1903, John V. Atanasoft and his assistant, Cliffoed Beery devised the first digital computer to work by electronic means, the ABC (Atanasoft - Beery Computer) called COLOSSUS started.

The first large - scale general purpose electronic digital computer called ENIAC (Electronic Numerical Integrator and Computer) used to compute firing and ballistic missile tables for Army Artillery guns was put into operation in 1946 at the University of Pennsylvania (USA) by John Mauchly and J. Prosper Eckert to help people use these machine . These techniques have since become programming languages that are used extensively in solving problems on modern computers. IBM (International Business Machine), formed in 1911, moved continuously into the computing market with its first commercial computer - the 701 in 1952. IBM has been in an ideal position to dominate the world computer market which it has done consistently.

The growing presence of computers not only broadens our reach in the same physical sense that electronic communications do, but also makes the power of infinite intelligence available. The question is, will the computer change society more radically than other modern technologies have? There are strong indications that it will, for example, if the developments in computers and communications enable people to work and learn at home, erecting office buildings and schools would not be necessary.

Computers are not meant to usurp human roles, but to aid individual work(s). Although, computers can be used for complex human tasks, they should not be used to replace human tasks, they should not be used to replace human judgements. People must remain responsible for their actions and maintain control over the computer.

1.2 COMPUTER GENERATIONS AND ITS CHARACTERISTICS

Generally, advances in computer technology can be calssified into categories called computer generations. What distinguishes each generation is the main electronic logic element in use at the time. The term logic elements refer to the electronic components used to facilitate the circuit functions within the computer. The generation and their logical elements are as follows:

1) First generation (1951 - 1958): Vacuum tube
2) Second generation (1959 - 1963): Transistor

3) Third generation (1964 - 1970): Integrated Circuit.
4) Fourth generation (1971 - 1990): Mircominiatorized Circuit.
5) Fifth generation(1990 to date) : Major advances.

Each new logic element led to improvement that made computers significantly faster, smaller, more flexible, more reliable and less expensive than those of past generations.

The first generation of computers which began with Eckert and Mauchly's ENAAC is considered to span the period 1946 - 1958. This generation of computer is characterised by the use of vacuum tubes in the Central Processing Unit (CPU) and Internal Memory Units. The first commercial computers and many fundamental advances in computing emanated from this generation.

In the second generation of computers, (1959 - 1963), vacuum tubes was replaced by the transistor. The transistor, a solid - state device was the major break-through that allowed computers to have reasonable size and power.

The third generation computer era was characterised by advanced miniaturization of circuitry. This was the introduction of the integrated circuit in 1964. With this technological advance, an entire circuit board containing transistors and connecting wires could be placed on a single chip. This development meant greater reliability and compactness combined with low cost and power requirements.

The fourth and current generation of computers begain in 1971 with the introduction of the Mircoprocessor - a central processing unit on a chip. This generation includes the introduction of super compilers. These monster computers' are in heavy demands for military and metemological applications that require a high speed of operation.

Another important advance of this generation has been the introduction of the personal computer, because the power for the computer has been made available to anybody who wishes to use one.

Although, we are potentially in the fourth generation of computers, research into a fifth generation of computer systems is now underway. Japan and the United States are heavily committed to developing fifth generation computers. These computers will depend on a major advances in artifical intelligence, voice recognition and image processing. If successful, the fifth generation computers will be more powerful and easier to use. They will utilise circuit chips manufactured using ultra large scale integration (ULSI) techniques. ULSI chips contain between 1 million and 100 million transistors.

Japanese researchers predict that the fifth generation computers will be used everywhere, serving as intelligent assistants' and giving users access to a broad range of information and expertise. In the office, the machines will accept spoken requests, search through reservoirs of stored knowledge, and decide which information is most relevant to the management's decision - making. In the home, the computers could give advice on personal financial management.

1.3 COMPUTER CLASSIFICATIONS

Computers can be classified in several ways - according to size, the type of logic they use and by purpose.

a) **By Size**

In term of size, computers can be broadly classified into three types namely, mainframes, microcomputers, (also called personal computers) and minicomputers.

i) A mainframe is a very large and expensive computer (usually costing over ₦10 million) that required a special supporting staff and a special physical environment (air-conditioned area). They are generally used in large businesses, government agencies and academic environments because they can support multiple users and multiple functions.

ii) Microcomputers or Personal Computers are small, one-user computers that are relatively inexpensive (₦10,000 - ₦50,000) and do not require a special environment or special knowledge to use them. At microcomputers can be programmed or used by one user in a manner very similar to that of mainframes but they are slower and cannot store as much data as the mainframe computers.

iii) A Minicomputer can support multiple users and require a special environment and some supporting staff but not on the same level as the mainframe.

b) **By the Type of Logic the Used**

In terms of the type of logic used, a computer can be either a digital computer or an analog computer.

i) A digital computer uses numbers and is therefore a counting machine. The mainframe and personal computers are examples of digital computers.

ii) Analog computers on the other hand uses physical relationships and is a measuring machine. Analog computers are often used to convert physical measurements into values.

c) **By Purpose**

In terms of purpose, computers can be classified as special purpose or general purpose computers.

i) Special purpose computers are designed for only one purpose. The computers used for guiding space shuttles are example of special purpose computers.

ii) General purpose computers can be used for many purposes. For example, a general purpose computer can be used for playing games, to handle payroll computations, for graphics to design buildings, for solving complex mathematical problems.

It must be said that one subset of mainframes include supercomputers which are the biggest and fastest computers in use today. These very large computers are used almost exclusively by Higher Institutions, research centres and government agencies for research projects that require high speed and large storage capacities.

1.4 BASIC COMPONENTS OF A COMPUTER SYSTEM

Computer system consists of a number of components, each performing a specific function. The system consists of input devices (used to send information to the computer), the Central Processing Unit (CPU), storage devices and output equipment that communicates computer results to humans. These physical components of the Computer system are called

Hardware. Hardware includes all input, output and storage devices. Hardware may be off-line (not controlled by or in communication with the Central Processing Unit) or On-line (controlled by or in communication with the central processing unit).

Complementing the hardware is a software system which includes an operating system and other programs that direct or instruct the computer system to perform specific functions. One of such software system is the operating system, a set of programs that control the input and output, communicate with the computer operator and schedule the computer's resources to allow for continuous operation with minimum manual intervention. Figure 1.4.1 shows the relationship between the components of computer system.

a) **Input Device**

The many methods of input to a computer range from switches on a computer console to easy-to-use voice input. Other means of input include computer cards, the keyboard, optical character readers, the light pen, joy stick and a device called a mouse that communicates with the computer by its movement over a flat surface.

b) **Internal Memory**

The internal memory consists/comprises the following: Random Access Memory(RAM) and Read Only Memory(ROM).

i) Random Access Memory (RAM)

This is used for storing the instructions in the computer and the symbols that is to be manipulated. The only internal memory that is accessible to the user. The shortcoming of random access memory is that it is a volatile memory - the memory exists only when the computer is turned on or the users is connected to the mainframe.

ii) Read Only Memory (ROM)

This is the section of the memory that is placed in the computer during the manufacturing process and remains there even after the computer is turned off, that is the contents of ROM cannot be changed and are not lost when the power is turned off.

iii) Storage Devices

Computer storage sometimes called memory is actually an electronic file in which instructions and data are placed until needed. When data come into the computer through an input unit, such as keyboard, it is first converted to binary before it is placed in storage. The data remains there until it is called for by the computer control unit. Magnetic tape, Magnetic disk, floppy disk are all forms of direct access storage.

iv) Central Processing Unit (CPU)

The most complex and powerful part of a computer system is the Central Processing Unit (CPU). Like an engine in a car, the Central Processing Unit is what makes the computer system go. The Central Processing unit is made up of two parts:

a) The Control Unit (CU)

The control unit coordinates the operations of the entire computer system. The control unit performs the most vital function in the Central Processing Unit. It is the "brain behind the operation" where all program steps are interpreted and instructions are issued to carry out the programs. The Control Unit directs the overall functioning of the other units of the computer and regulates data flow. When the computer is operating under program control, the control unit brings in data as required from the input devices and directs the routine of results to the required output devices.

b) The Arithmetic/Logic Unit (ALU)

This unit performs all calculations and makes decisions. It controls all operations of arithmetic and logic. As directed by the control unit, the Arithmetic/logic unit performs additions, subtractions, multiplications and divisions on numerical data. It also compare numerical or alphabetical information to determine whether it is the same or different. The Arithmetic/Logic unit of most computers perform all arithmetic functions using binary numbers.

v) Output Device

The two most popular output devices are the printer and the video screen or monitor. The other name for a monitor are Cathode Ray Tube(CRT) or Video Display Terminal (VDT). The monitor is very useful for any computer system for two reasons. First, it shows the users, the data or instructions that are input from the keyboard or other input devices. Second, the monitor is an almost instantaneous outlet for results.

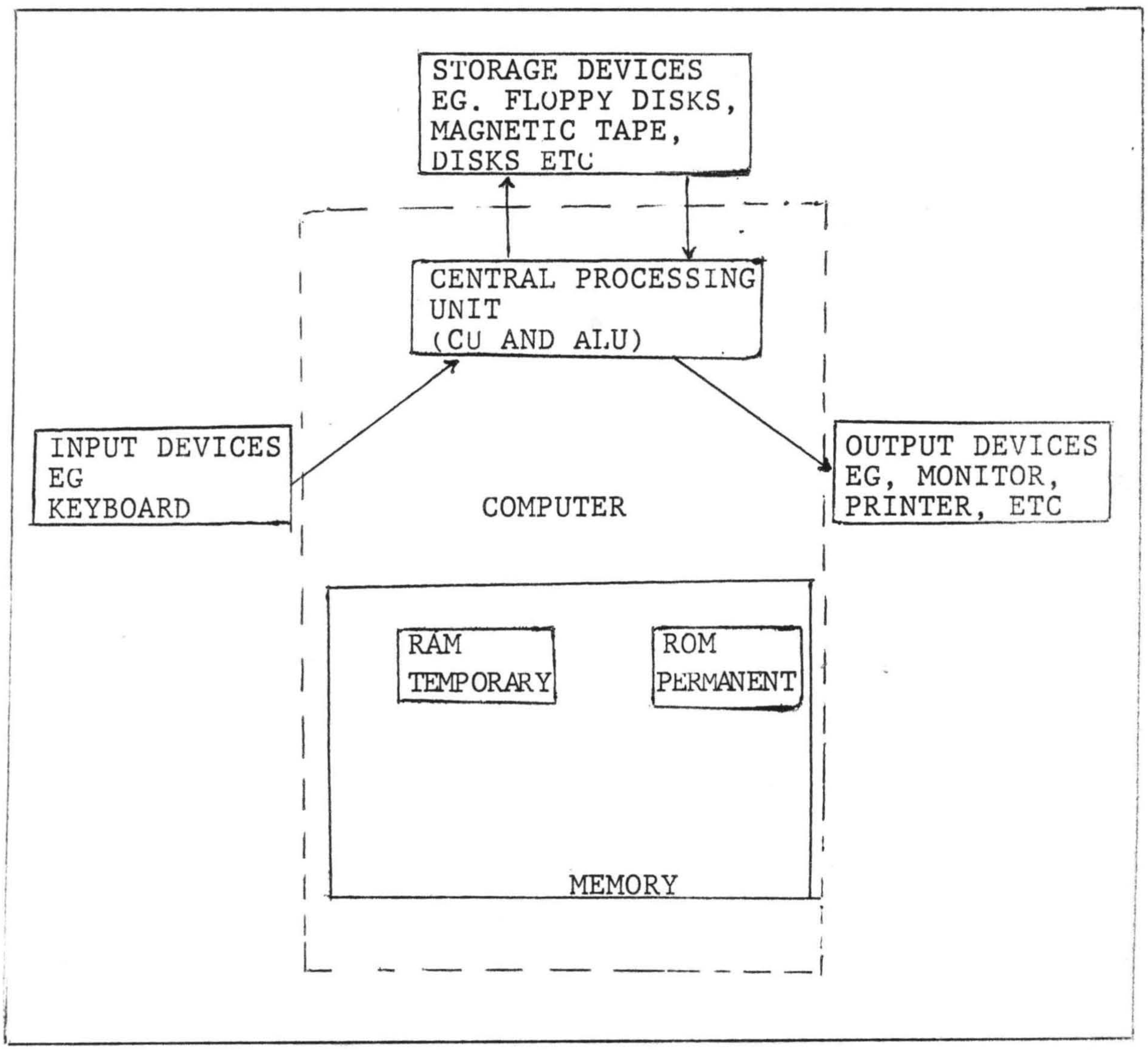


Figure 1.4.1 Components of a Computer System.

CHAPTER TWO

2.0 SYSTEM'S LIFE CYCLE APPROACH

2.1 SYSTEM ANALYSIS AND ANALYSIS'S TOOLS

Just as we are aware that the Computer is surrounded by an intricate web of software, so the procedure for computerizing problem is made more complex by a large area of computing which is called paracomputing, and of which system analysis and design form the major part.

System analysis is a discipline whose development dates back to the late 1950's during which time commercial organisations began to make use of computer.

Analysis of a system is the procedural study of its operations with an attempt to discover its basic problem areas. The analysts must examine all the facts he has gathered in order to make proper assessment of the existing system. He must resist the temptation of including ideas which have not been fully worked out in the new system.

The aim of this approach is to ensure that all feasible alternatives are eventually produced.

The present system must be criticised against the principles of procedure, after which the strength and weaknesses of the system should be apparent. The principles of procedures used are:

i) **Purpose**

Is the purpose being served? Are they still necessary? could they be achieved in any other way?

ii) **Economical**

Benefits should be related to the cost of producing them. Are there more economic methods?

iii) **Workflow**

Is the workflow satisfactory?

iv) Specialisation/Simplification/Standardisation.

v) **Flexibility**

What would be the effect of the system on the increase or decrease in the volumes to be processed.

vi) **Exception Principle**

Factors requiring action should be highlighted and not summed up in a mass of routine details.

vii) **Reliability**

What provision is there for such events as staff sickness, machine breakdown? Could more up-to-date equipments be justified.

viii) **Form**

Is the information being produced in the form best suited to the recipient? Is there a need for a hard copy?

ix) **Existing System**

If a change is made, what equipment and other facilities currently being used could be incorporated in the new procedure?

x) **Continuous Control**

What type of errors are occuring? What other type of controls could be used?

xi) **Time**

Is the information being produced in time for meaningful action to be taken?

One important aim of analysis is to produce a requirement specifications/analysis.

System analysis may therefore be defined as the methods of determining how best to use computers, with other resources to perform tasks which meet the information needs of an organisation. It was developed as a specialized branch of Organisation and Method (O & M)which is general approach to solving procedural problems. Organisation and Methods(O & M) can be defined as the systematic analysis of selected procedural problems in order to produce alternatives which would be more suitable, technically and economically. System analysis consists of a series of stages. These are what is often called the system lifecycle. They are:

1) Preliminary Survey/Study

The purpose at this stage is to be able to establish whether there is a need for a new system and if so to specify the objectives of the system.

2) Feasibility Study

This stage of the system life cycle is to investigate the project in sufficient depth to be able to provide information which either justifies the development of the new system or shows why the project should not continue. The findings are presented to the management in form of report.

3) Investigation and Fact Finding

This stage stipulates that a more detailed study be conducted. The essence is to fully understand the existing system and to identify the basic information requirements.

4) Analysis

Analysis of full description of the existing system and the objectives of the proposed system should lead to full specification of the user's requirements. This requirement specifications can be examined and approved before system design is embarked upon.

5) Design

The analysis may lead to a number of possible alternative design. Once the alternatives have been selected, the purpose of the design stage is to work from the requirement specification to produce a system specification.

6) Implementation

This involves following the detailed set out in the system specification. The two important tasks involved in this stage are programming and staff training.

7) Maintenance and Review

Once a system is implemented and is in full operation, the system is examined to find out if it has met the objectives set out in the original specification.

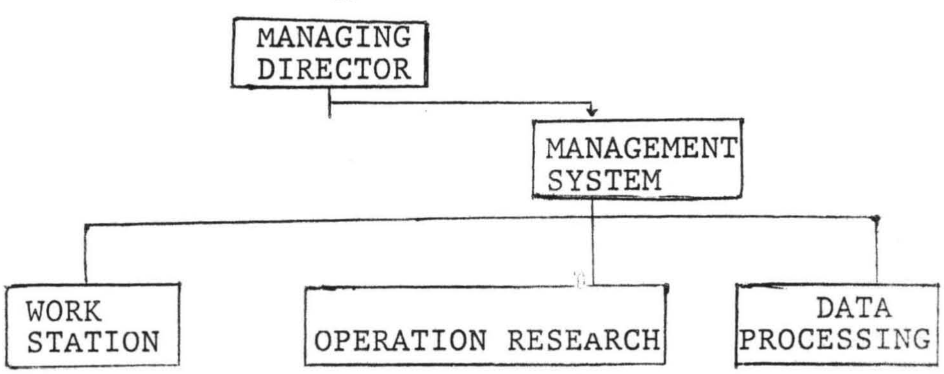
From time to time, the requirements of the organisation can change and the system have to be examined to see if it can cope with the changes. At some stage, the system lifecycle will be repeated again and yet again.

An indepth/wide range knowledge of both business methods and computing as essential to the system analyst. Since no one can be expected to be skilled in all fields, there is a tendency today in large business systems to have a business analyst who can specify the system design (in equal technical terms) and the technical analyst who takes the specification and assume responsibility for the design problem associated with the computer system.

The business analyst would have to be familiar with some of the following subjects: business structure, organisation management and administration, production planning and control, stores and stock control, accounting, operations research, conducting and analysis surveys, simulation and model building.
Technical analyst on the other hand should have a versed knowledge of the techniques of data processing, programming, computer operations, computer systems (including the current cost and performance of both hardware and software). Besides, the technical analyst should be able to give advice on the relative merits of input/outputs and the secondary storage most appropriate to the business needs.

The requirements of people engaged in system analysis involved wide technical knowledge, business knowledge and experience and these are not normally found in one individual. Thus, teams of analysts are generally formed and operated specified assignments. The way these analysts may fit into an organisation's structure is shown bwlow in A and B of figure 2.1.1.

A



(B)

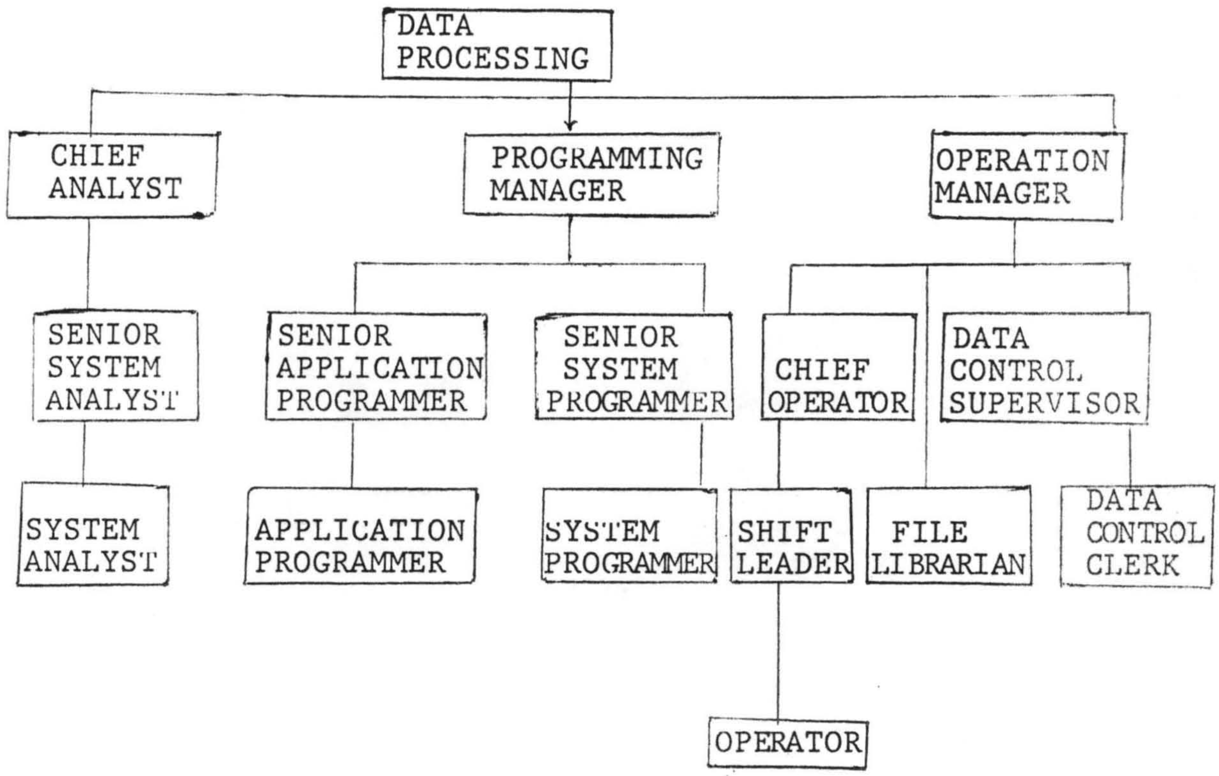


FIG 2.1.1 Organisation chart showing the link between Data Processing and Management System.

Analysis Tools

Analysis tools are the tools (not mechanical/electronic) that the system analyst uses when carrying out system analysis. They include:

a) Investigation

It is important that clear objectives are laid down by the management before the analyst begins his task. The Chief Analyst therefore has to confirm the 'terms of reference', the number of analysts required, determined and the length and breadth of the assignment will depend to a great extent on the objectives set.

b) Fact Finding

It is essential to gather all the facts about a current system to ensure that all strengths and weaknesses are discovered. Thus, when a new system is designed, as many of the weaknesses as identified are eliminated whilst retaining the strengths/the system is further strengthened. There are four main techniques used in fact finding process and the technique used depend upon the particular circumstance.

i) Interview

This is probably the most widely used tool and the most productive and reliable. During interviews, facts about what is happening come to light together with the opinion of the interview regarding the weaknesses existing in the system. The personal contacts are important in getting the cooperation of people involved as they would be convinced that they are making susbstantial contributions towards the design of the new procedure. It is vital to gain the confidence of the individual concerned at this stage in order to gather all the basic facts required.

ii) Questionnaires

The use of questionnaires is particularly useful when a little information is required from a large number of people. Moreover, when the study involves many different geographical locations questionnaires may be the only practicable method of gathering the required facts.

iii) Observation

This is best employed in conjunction with other technique and carried out after the observer has understanding of the procedures involved.

iv) Record Inspection

The study of organisational charts, procedure manuals and statistics can reveal much useful information about a procedure

Fact Recording

During the fact finding stage, unless the investigator has formulated a plan for the keeping of notes of the facts, he will end up with a mass of notes on all areas which will be difficult to examine. A good practice is to divide notes into areas of investigation such as department or operation, or by type as organisation charts, interview, procedure charts, work charts and decision table. The flowcharts will reflect the varying degree of details required.

2.2 <u>DESIGN OF A COMPUTER SYSTEM</u>

The system analyst is called upon to use his creative abilities at this stage. To achieve this objective, the analyst has to apply his judgement, skill and knowledge to interpret the requirement specifications to create one or more system specification provides detailed documentation of the new system and requires acceptance by the management just as requirement specifications do.

<u>**System design**</u> - The creative phase in the system development cycle, consists of synthesizing the requirements specification into a cohesive and focused information framework. The design activity is not a simple task since numerous designs alternatives are usually available. The optimal design is one that best suits/fits the particular set of circumstances at hand while meeting several tasks of processes.

Design, as a creative task, has its objectives as the implementation of a system creating benefits and improvements superior to those achieved by other methods. The system therefore must be designed in such a way that basic results are produced as effectively as possible in accordance with the needs of the organisation.

An effective system design corrects the weaknesses and problems that led to the system development Project. Expressed more positively, the design fulfils the objectives of the project which have been converted into specific system and information requirements during the system analysis phase.

Generally, a wide variety of design alternatives confront the system analyst /designer at the begining of the design phase. Some of these alternatives can quickly be discarded when it is discovered that they do not perhaps fit the particular set of circumstances at hand and it is discovered that they would consume a disproportionate share of resources. Other alternatives may represent plausible possibilities. However, only one of these alternatives represents the best and appropriate choice. The role of the system designer is to work with users to narrow down the alternatives until the best choice is found.

We might view the alternative design possibilities in two ways.

i) <u>To what extent do they differ from the present design</u>

According to this view point, alternative designs range from a very slight modification of the present system to a radically new design. Consider the alternative designs available when the present information system involves manual processing. The manual processing features would be retained though with new additional control or the source document revised.

ii) <u>What Features do they Exist</u>

This views designs alternatives as the combination of system features. These features include input sources, input devices, data processing devices; and approaches, storage media/devices; output destination and output media/devices.

Usually, design specifications are grouped around the components of an information system. The design specifications typically consist of the features listed in figure 2.2.1 They provide explicit outline of the capabilities and peculiarities of a specific system design.

SYSTEM COMPONENTS	FEATURES
1. Output	Name Purpose Contents(information items) General Format Output Medium
2. Database	File Name File Type File Size Content of Record(data items)
3. Data Processing	Sequence of steps or runs Processing Capabilities at each Physical location. Processing Mode Data Communication Mode
4. Data Input	Name Purpose Data Entry Method General Format Contents(data items) Method of data collection Source
5. Control	Type Purpose Method of Correcting Error

Figure 2.2.1:- Typical features included within design specifications.

The preparation of design specification begins with the information output. As the final products of the information system, the outputs are the key determinants of the remaining system components. After the output specification, data base specification is followed then by date processing and data input specifications. The system control should be specified in conjunction with the specifications for each of the other elements.

Figure 2.2.2: Shows the desired sequence for designing system components.

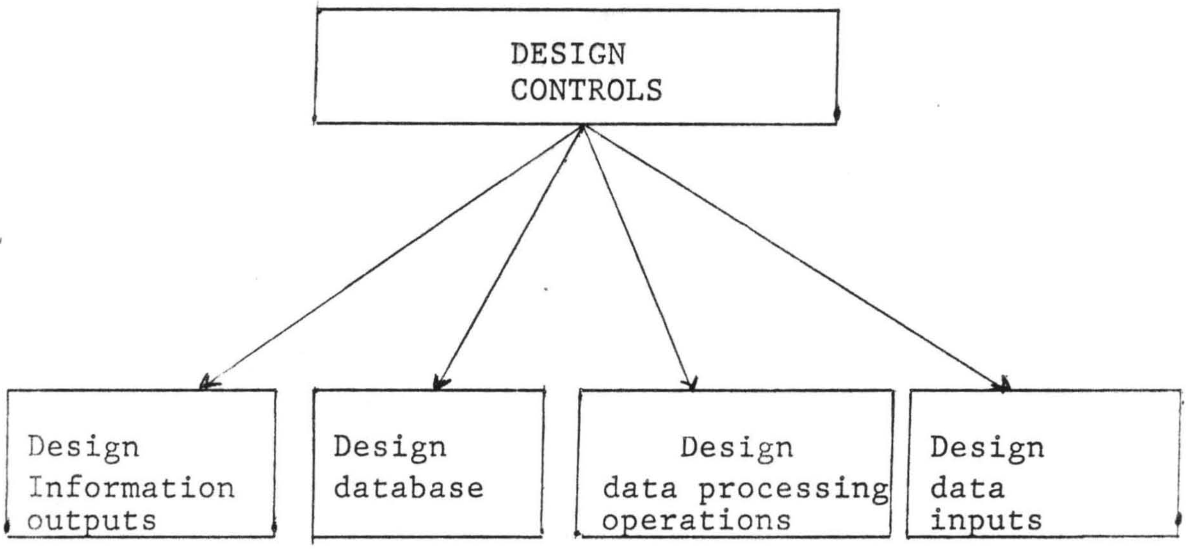


Figure 2.2.2 - Shows the sequence of designing system components.

As can be noted, various methodologies relating to system design are increasingly being used. One major advantage is that they guide a system analyst in the analysis and design phases of the system lifecycle. In effect, system methodologies provides a disciplined step-by-step approach that increases the productivity of system analysts. They also aid analysts in spotting inconsistencies and omissions. Structured techniques are incorporated to enhance these advantages. Developing new information systems in large organisations is a tremendous task. Consequently, methodologies that elicit the aid of computers are helpful and even necessary. In some cases, methodologies have been devised by individual organisations which are tailored to their particular needs. Figure 2.2.3 illustrate the steps from the system's definition to the final application programs.

Methodologies, particularly when computerized are proving to be a boom to system development. Perhaps their greatest limitations is that they do not eliminate the need for human analysts to define the requirements for new systems. Furthermore, all the methodologies require extensive training of analysts.

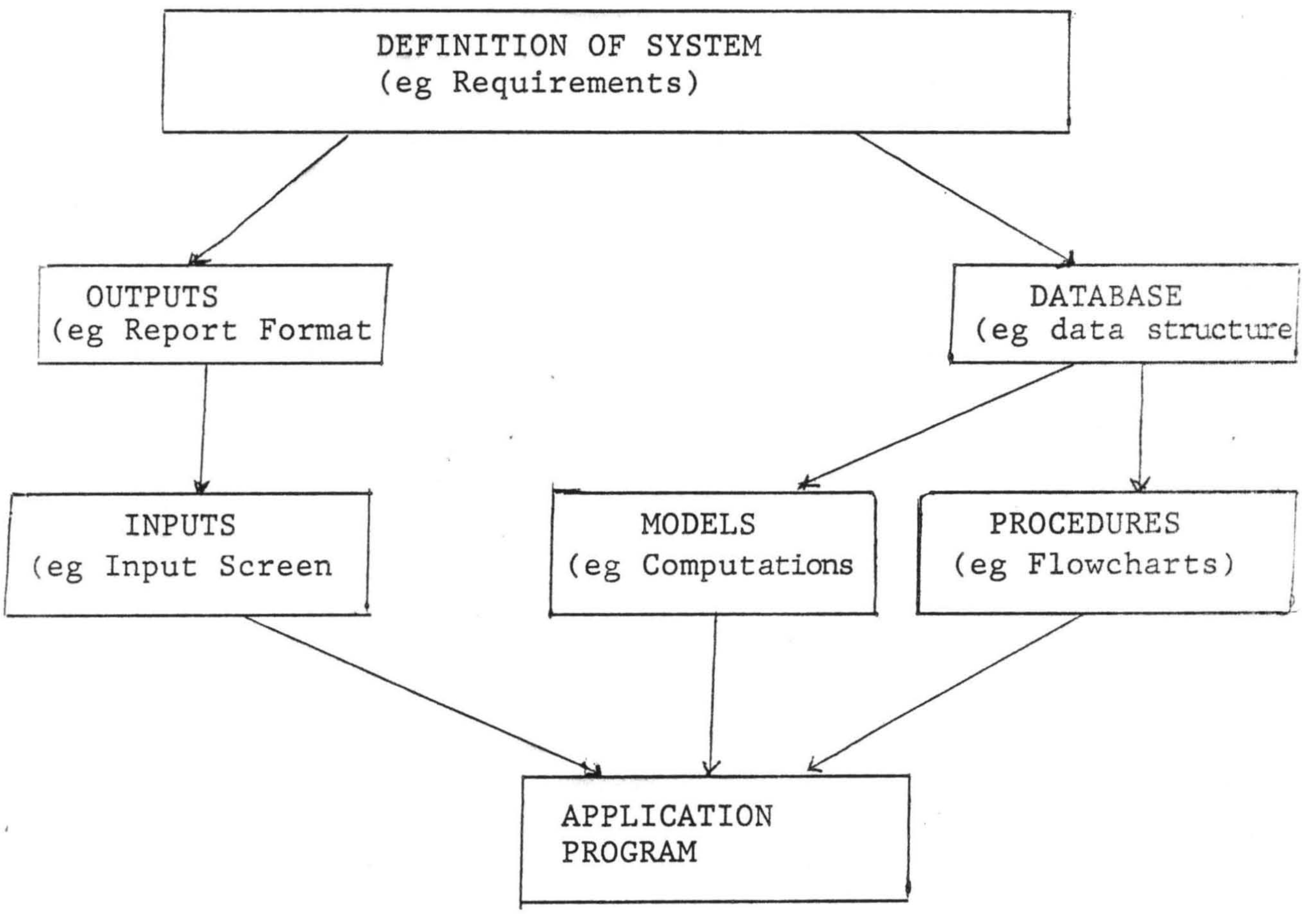


Figure 2.2.3 - Components of a system methodology.

Having analysed a particular system and prepared a problem definition that has been approved by management; the question then arises as to where the analyst would commence the design of alternatives to the current system. The following techniques gives a logical sequence in which to design the various elements of an alternative system.

It is analogous to the approach used in analysing systems with some additional considerations.

The sequence of activities to be performed by the analyst in designing a new system is as follows:

<u>A</u> Examine the objectives of the new system. It is quite possible that the analyst may recommend to the management a modification of the stated objective.

<u>B</u> Review all constraints imposed on the system and the analyst. In most cases, the constraints for the proposed system will be similar to those of the current system, except for the hardware and budgetary limitations.

<u>C</u> Design the outputs that would be produced under the new system. This facet of design work compels the analyst to work closely with the management so that their needs and desires are fully appreciated. In designing outputs for the new system, the analyst must basically consider several points

a) He must be certain that management would receive all necessary reports in the new system.

b) He must be innovative enough to recommend to management new reports that can yield important information.

c) He must be responsive to the need of the business so that he can perhaps suggests outputs other than the one intended.

The analyst must always remember one critical point that he can only make recommendations but cannot dictate.

In designing new reports to recommend to the management, it is ideal to design a sample form that has representative data on it. If a multiple copy is being suggested , all copies should be included in the sample.

<u>D</u> Design the processing steps that would be needed to procure the desired outputs. In most cases, the new design will utilise/require a computer system to

facilitate processing. The analyst must determine the most efficient method of utilizing computer equipment in the system. The use of computer allows the analyst to have greater degree of flexibility that are feasible in a manual system or a unit record time.

One common tool used for organising ideas during the designing of processing steps is the system flowcharts. These flowcharts depict the relationship between inputs, processing and outputs. It makes it easier for the management to review the new system so as to give the programmer a clear insight into the workings of the system.

E Design the necessary inputs. There are two basic types of inputs to be considered.

i) Source documents must be designed so that all information necessary for processing are supplied. These documents should be designed in such a manner analogous to the design of outputs mentioned above.

ii) The designer (analyst) must decide what files that are needed to retain data for producing desired reports whenever they are required. A decision must be made as to what file that best suit the system being designed.

F Incorporate the necessary controls. Controls provide checks on the system to determine if it is operating properly and to ascertain minor inadequacies before they develop into major ones.

G Design Feedback procedures. In theory, everything in a newly designed system should work well once the system is operational. Pragmatically, however, the analyst should envisage the possibility that everything may not run smoothly, as it may happen that adjustment must be made to the information stored in computer files.

H Prepare a plan to install the new system if the management approves the design. This plans must include the following items:

a) The system must undergo a system test once all programs have been written and the training of departmental personal is completed.

b) Time must be allowed for training the personnel who will be involved in the new system. All procedures must be explicitly explained to the users (ie people who will be using them).

c) The type if implementation must be made clear, concise and unambiguous to the management.

d) The analysis must consider the need to follow up on the new system after it has been operational for sometimes, approximately three months after the new system starts to work.

I Prepare cost analysis to show the management in quantitative rather than qualitative terms, the benefits that can accrue from the implementation of the proposed system. Cost analysis are often the determining factor in the approval or disapproval of a proposed system by the management.

2.3 SYSTEM IMPLEMENTATION AND REVIEW

Implementation follows on from the detailed design approach. This involves the co-ordination of the efforts of the management system department and data processing department in making the system operational. The analyst responsible for the design of the new system is an important member of the implementation team because of his thorough knowledge of the system, though the details of the implementation would have been stipulated in the system specification. They would cover the following:

a) **Training of Staff**

The amount of training required for various categories of personnel will depend upon the complexity of the system and the skills presently available. The following aids can be used as appropriate:

i) Handbooks (ii) Courses and (iii) Lectures.

b) **Programming**

The programmer must design programs which conform to the requirements set out in the system specification.

c) **System Testing**

There is a need to ensure that the individual programs have been written correctly and that the system as a whole will work. However, there must also be co-ordinator with clerical procedure involved. To this end, the system analyst must provide the necessary test data as follows

i) Program testing (for the program).

ii) Procedure testing (for the whole system operation). This involves the clerical procedures which proceed input, the actual machine processes themselves, and the output procedures which follow.

d) **File Conversions**

The stages of file conversion will depend on the method currently being used for keeping the files but are likely to be:

i) Transcription of all standing data to a special input document designed for ease of data entry.

ii) Insertion of all new data required into input document.

iii) Transcription of data from document to storage.

iv) Verification of transcribed data.

v) Data is then used as input to a file creation run.

vi) Printing out of files for change over to a new system.

a) Parallel and/or (b) Direct.

In the parallel method, old and new systems are runned concurrently, using the same inputs. The outputs are compared and reasons for difference resolved. At this point, the old system is discontinued and the new one takes its place.

In the direct method, the old system is discontinued altogether and the new system becomes operational immediately.

A variation of either of the two basic methods is the so called pilot changeover which involves the changing over of the system, either in parallel or directly.

Once the system has become operational, it will need to be examined to see it it has met its desired objectives. The system would need to be reviewed and maintained for the following reasons.

a) To deal with unforseen problems arising in the operation.

b) To confirm that the planned objectives are being met and to take action if they are not.

c) To ensure that the system is able to cope with the changing requirements of the business.

CHAPTER THREE

3.0 PROGRAM DEVELOPMENT

3.1 CHOICE OF PROGRAMMING LANGUAGE

A computer program is a set of instructions aimed at solving a specific problem. A specific problem usually has a set of objectives which have to be accomplished. In order to achieve these objectives, there are a set of input data, operations to be performed on the data which could be query or update type and output reports.

Programming language selection involves determining the best programming language for the application. Some important characteristics to be considered are:

i) The difficulty of the problem
ii) The technical skill required of the computer programmer.
iii) The availability of progrommers for various languages.
iv) The type of processing to be used.
v) The efficiency of the compiler or language translator.
vi) The support from the computer vendor in maintaining and updating the programming language.
vii) The ease with which program can be changed at a later date.
viii) The existing hardware and software configuration.
ix) The type of problem such as business or scientific.
x) The type of software automation packages.

Some examples of programming languages are FORTRAN, PASCAL, BASIC, DBASE, FOxPRO, CØ-LANGUAGE, ORACLE, and COBOL.

3.2 PROGRAMMING LANGUAGE AND OVERVIEW

There are at least two major ways in which programming languages can be viewed or classified. They can be classified either according to their level or their principal applications. However, these views are tampered with the historical evolution through which the language drifts.

3.2.1 By Levels

The four distinct levels of programming language are:

i) Declarative Language

The declarative language is more like English in its expressive power and functionality and this is the highest level at which comparison with others can be made. The fundamentalist command language is dominated by statements which express "what to do" rather than how to do it.

ii) High Level Languages

These are the most widely used programming languages, though not fundamentally declarative. These languages allow algorithm to be expressed in a level and style of writing which easily is understood and read by other programmers. Moreover, high level languages usually have the characteristic of 'portability'. That is, they are implemented on several machines so that a program can be easily transferred from one machine to another without substantial revision. Thus, they are termed "Machine Independent".

iii) Assembly Language

Assembly Language is simply a symbolic representation form for its associated machine language allowing less tedious programming than the machine language. However, a mastery of the underline architecture is necessary for effective programming either of these language levels.

iv) Machine Language

This is the language of the computer. It consists of series of zeros and ones. Machine language program is the language of the computer but it is very difficult to programme in machine language because the program must know the internal workings of the computer as well as their mode of communication. In whatever other language we program, a language translator must be used to convert from that form into the computer - machine language. Machine language is the fastest in execution because it does not need a translator. Any other language in which instruction is given to the computer would require translation.

3.2.2 **By Application**

a) Scientific Application

This can be characterised as those which predominantly manipulate numbers using mathematical and statistical principles as a basis for the algorithm. Often, the amount of data in such problem is relatively simple. Vast amount of data must be reduced before any meaningful and simple analysis can take place.

The Programmer must be vast in the mathematical principles underlining the algorithms in order to properly diagonise the problems or make proper refinement. Scientific problems usually require more of a computer central processing than its input - output devices.

b) Data Processing Application

This can be characterised as those programming problems whose predominant interest is the creation, maintenance, extraction and summation of data in records and files. The volume of data found in this file is generally large. A typical data processing program spent most of its computing time doing input and output operations. Data processing program must be at least protect the files from been contaminated by inaccurate data. Moreover, they must ensure that sensitive data be accessible only to thos who need to have access to it and no others.

c) Text Processing Application

This is characterised as those whose principal activity involves the manipulation of natural language text rather than numbers as their data. The evolution of modern word processing technology rely principally on text processing algorithm to perform the various formalities and other functions that an individual uses during manuscript, project, thesis-preparations. Usually, the text in such an application is in English.

d) Artificial Intelligence Application

The artificial intelligence application is characterised by programs designed principally to enumerate intelligence behaviour. This includes robotics,

natural language understanding programs and expert system. In expert system, the computer is programmed to play the role of an expert until recently when artificial intelligence confined its works to research laboratory where various pilot experiments model different kinds of artificial behaviours.
However, many of these experiments have been brought into the practical domains and their effects are shown in such diverse areas as automobile production lines and the monitoring of complex instruments.

e) <u>System Programming Application</u>
This involves the development of those programs that interface the computer system (Hardware) with the programmer and the operator. This programs include compiler, assembler, interpreter, input - output routine, program management utilities, and scheduler's for utilising and serving the various resources that comprise computer system.
The two characteristics that usually distinguished system programming and other types are:

i) The requirement to deal effectively with unpredictable events or exceptions.
ii) The need to co-ordinate the activities of various executing program.

3.3 FUNDAMENTALS OF COMPUTER PROGRAMMING

Since programming involves input, processing and output, all programming languages must contain the following

a) <u>Data</u>
It can be simply said that data consitute the raw materials which are processed by the computer. Data can be numeric (number), alphabetic (letters) alphanumeric (combinations of letters and numbers), string(characters enclosed in quotes) etc is processed by computer. Data can be constant (not changing in value) or variable (changing in value). A constant cna be numeric or string while a numeric constant can be integer(whole number) or real (with fractional parts). A variable holds or stores a constant and must be of the same type

as the constant it represents.

Data meant for processing must be properly and completely described to the computer through implicit or explicit declarations, so that the computer knows exactly what kind of data it is to operate upon and hence makes provision for storage requirements of the data. Through data declarations, a programmer gives name to the data elements in his program. These names called variables or identifiers enable the programmer to reference data in memory. Declarations merely provide information to the computer about data and do not cause the computer to carry out any action on the data during execution.

In other words, declarations are non-executable.

b) Instructions or Statements

The bedrock of any programming language is the set of instructions specified by the Programmer for the Computer to perform on data. Each instruction has a specific format or syntax (the way it must always be written) and results in a specific action as laid down by the designers of the particular language.

Instructions are categorised into:

i) Input/Outpit instructions for input and output of data.

ii) Control instructions which make the computer execute statements in the normal sequential order or branch to another statement, skipping a group of other statements.

iii) Assignment statement, which gives values to variables.

c) Operators

Data elements are combined by operators to form expressions. There are mathematical (for adding, subtraction, multiplication, division and raising to a power), relation operators (for making comparisons) and logical operations (for testing true and false conditions). The language designers decide on what symbol or word to use to denote a particular operation. Thus, the operation A + B in one language might be stated in another language as ADD A TO B. Relational operators (LESS THAN, EQUAL TO, GREATER THAN), and their varying combinations are used to compare the magnitudes of data elements yielding logical results; true or false.

d) <u>Delimeters</u>

Delimeters form boundaries for various program units such as a block (a group of statements to be performed together). A delimeter could be a punctuation mark such as a semi colon, a keyword or a blank or varying combinations of these.

e) <u>Program Organisation</u>

Elements of a program must be organised meaningfully to achieve intending objectives. This structuring of a language includes the rules for writing instructions and for relating such instructions to their associated data.

A program is organised into sub-units which may include executable and non-executable statements, loops or control structures, blocks, subprograms. Executable statements make the computer to perform some actions such as print a value, move a data value to some other locations. Non executable statements, such as declarations simply advice the compiler on the attitudes (type, storage requirement) of a data. Other non executable statements such as comments merely guide anyone reading the program so that a better understanding of it could be obtained.

Sometimes, statements are grouped together to form a block which is delimited by a begining and an ending designant. A complex program is sometimes broken into smaller units called subprograms, each of which performs a specific task. If a single value results from a subprogram, it is called a function, and if several values result from the subprogram, it is referred to as a subroutine.

Subprogram makes programming easier and more flexible, since different programmers can write up different subprograms for the various tasks involved in a very complex programs and the main program simply coordinates the activities of the different subprograms. The inability to repeat a group of statements makes programming language very powerful. The unit of a program which is repeated is called a loop and the process is referred to as looping using control structures.

Program control structures are used to regulate the order in which program statements are executed. Control structures are designed to help maintain the modula program structures. Each module has only one exit and one entry point. Control structures fall into three categories.

i) Sequence

In the sequence structure, operations are performed serially from the first to the last in their physically occuring order. This consists of sequence of two or more program statements called module that are executed in the same order in which they appear in the program. Figure 3.3.1 shows a carefully planned sequential structure.

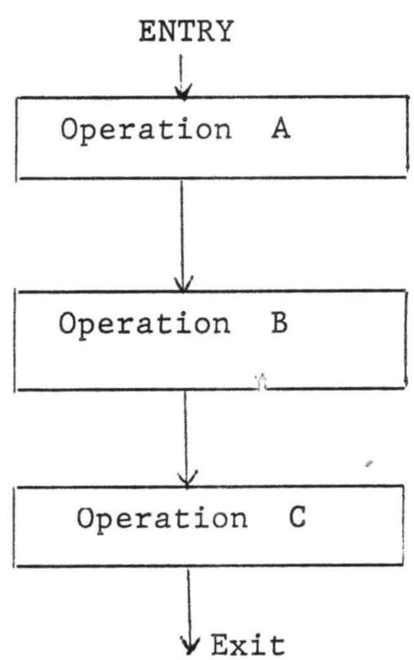


Figure 3.3.1 - Sequence Structure.

ii) Selection

In selection structure, a condition is tested and depending on the prevailing condition one of the alternative actions is performed. The alternative actions could be two - fold or many fold:

a) Two - Way Selection (Decision) Structure.

May decisions in life involved two - way choice of action. Under such situation, a condition is tested which yields the result, true or false, yes or no. One of the two actions will be taken based on whether a particular condition is met or not. The selection structure chooses one or two possible paths based on the evaluation of a condition which can be true or false, Yes or No. A double decision structure is implemented with IF .... THEN ....ELSE statement. A two-way decision method is shown in figure 3.3.2.

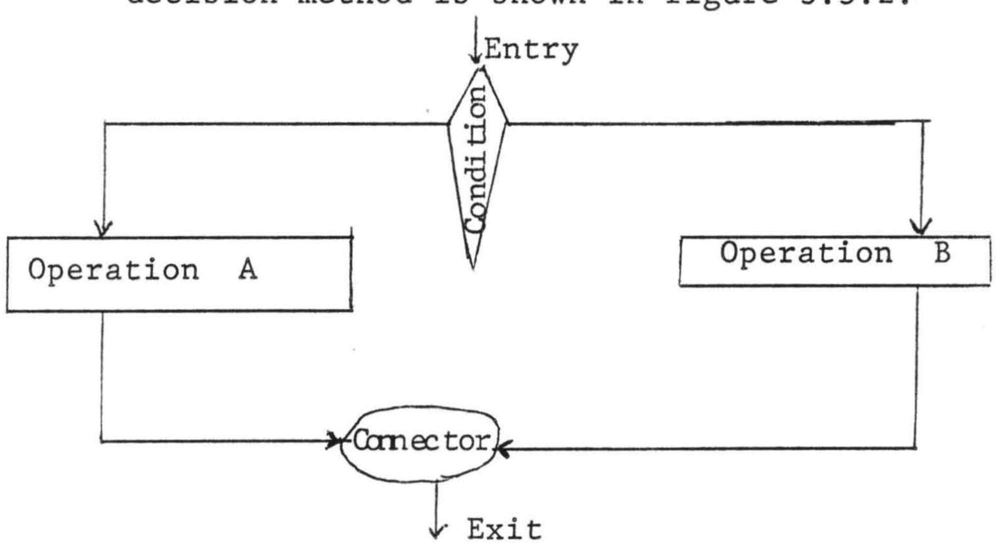


Figure 3.3.2 - Two way selection structure.

a) Multiple Decision Structure

This selects from a number of options based on the evaluation of a condition. A decision involving a choice of more than two alternatives is called a CASE structure. The illustration in figure 3.3.3 shows how the multiple decision structure (case structure) is implemented.

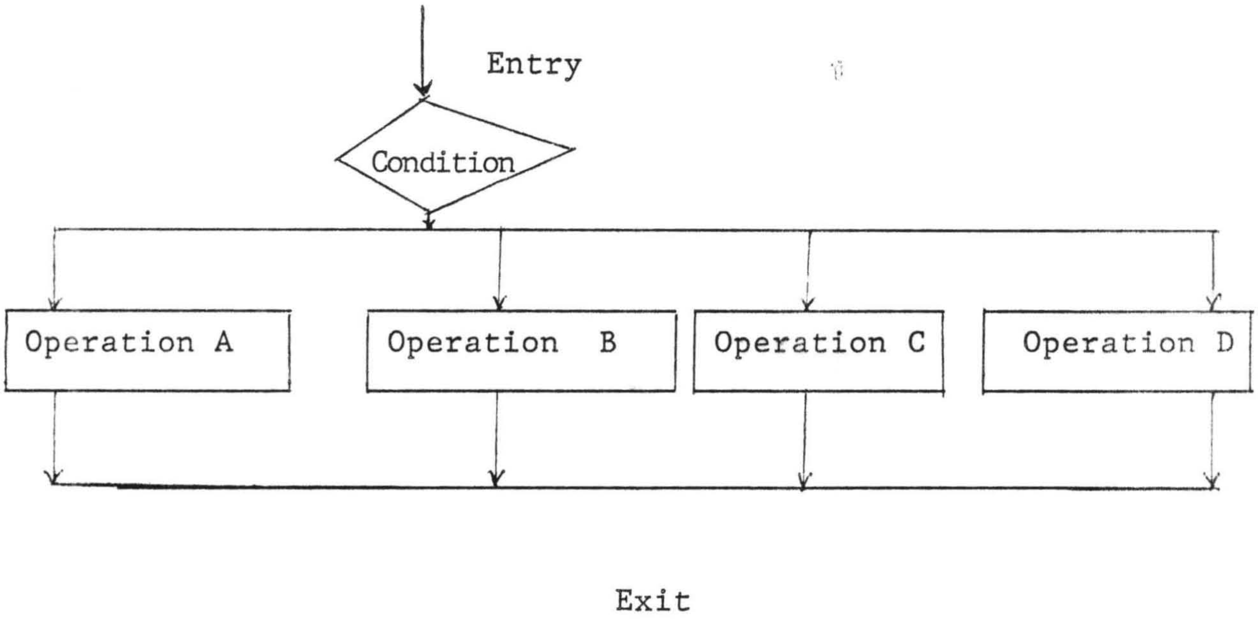


Figure 3.3.3 Multiple Decision (CASE) Structure.

iii) <u>Iteration Structure</u>

This structure is also called repetative structure. The same action occurs repeatedly as long as a condition holds true. This structure is subdivided into three categories.

a) WHILE .... DO

The processor evaluates the condition. If the result is false, execution of while statement is complete. If the result is true, the statement is executed once, condition is tested again and so on. Figure 3.3.4 shows the execution of WHILE .... DO structure.

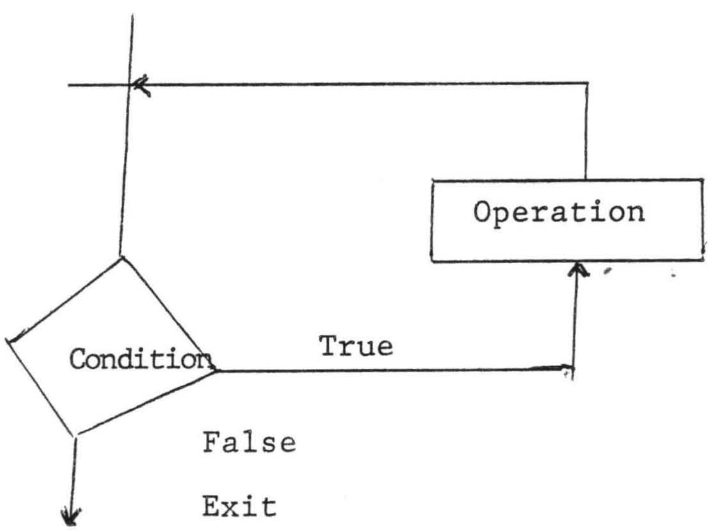


Figure 3.3.4 WHILE ..... DO Iterative Structure.

b) REPEAT .... UNTIL

The effect is that the statement sequence is executed once and the condition is evaluated. If the result is true, execution is complete. Otherwise, the statement sequence is executed again, condition is evaluated again and so on. Figure 3.3.5 shows the execution of REPEAT .... UNTIL Iterative Structure.

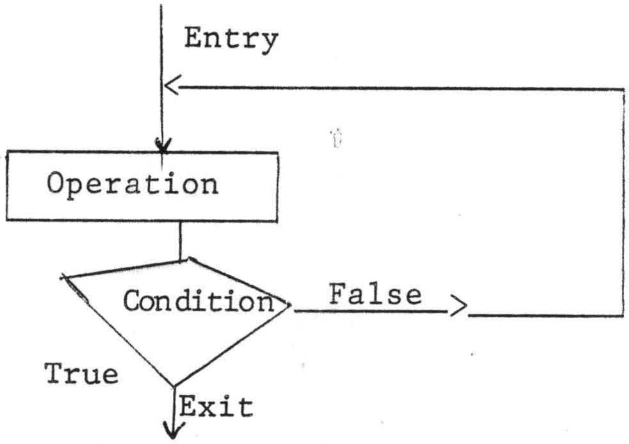


Figure 3.3.5 REPEAT ... UNTIL Iterative Structure.

c) FOR .... DO

This is used when a statement is to be executed repeatedly and the number of repetitions does not depend on the effect of statement with the loop.

3.4 PHASES OF COMPUTER PROGRAM DEVELOPMENT

i) The Presentation of Program Flowchart and Algorithm

The processes for solving the problem can be expressed using flowcharts and Algorithms. The flow charts and Algorithms are programming tools intended to aid in the visualisation of the overall flow of the program. They provide the necessary details for coding computer problems under study

ii) Flowchart

Flowchart is a pictorial representation of the logic or method used to solve a particular problem. Also, it is a graphical representation of the sequence of step that a person or machine must execute to arrive at the solution to a problem. It is composed of simple descriptions contained in a special symbols that are connected by straight lines.

There are basically two types of Flowcharts

a) <u>System Flowchart</u>

System Flowchart is concerned in the complete system. It shows the sequence of operations and relationship between them. It is an ideal communication media. It is also part of the documentation given to programmes to enable them to write the necessary programs.

b) <u>Programe Flowchart</u>

Program flowchart respresents the operation carried out by the computer. Outlined program flowcharts represents the first stage of turning the system is flowchart into necessary detail to enable the programmer to write the program, and while detailed program flowcharts are prepared from the outline charts which contains detailed steps necessary for computer to perform a particular task.

Flowchart symbols written within the symbols are instructions to indicate what operation is to be performed. The instruction reflects the general functional operations used in the computing system. The program flowchart symbols are shown in figure 3.4.1 below.

	An Oval-shaped called a terminal symbol, indicates the begining or end of a source program.
	A rhombus shape indicates an input/output operation.
	A rectangle indicates a processing, causing changes in the form, value or location of information.
	A diamond-like shape indicates a decision process or a switch type operation that determines which of a number of alternative paths to follow.
	Connector
	Arrows showing director of flow
	Page Connector
:	Comparison A : B means Compare A to B.

Figure 3.4.1 Flowchart symbols

ii) Algorithm

The second step in program development is to work out a step-by-step solution for the problem at hand. This step-by-step method or approach is called an algorithm. It expresses the logic of the program in written form. An algorithm is a finite set or sequence of instructions for carrying out a specific procedural task. On the other hand, a program is an algorithm specifically expressed in high level language capable of being executed by a computer.

The following are some of the basic characteristics of a valid algorithm.

a) Precision

Steps must be void of assumptions but must be explicit. Thus, there are varieties of pseudo-codes (logical representation of an algorith using such third generation language style such as WHILE ...DO, IF .... THEN .... ELSE etc) to express concisely some complex or technical concepts.

b) Effectiveness

An accurate precision makes an algorithm effective.

c) Finite Set of Instructions or Operation

There must be an exact number or finite set of instruction in an algorithm. For instance, if A = (i, i_2, i_3, ... in)represents a set of instruction in an algorithm, then we must have n & Co where is are the algorithm's instruction.

d) Termination

The execution of an algorithm must have a stopping criteria to terminate the algorithm especially in a case of instruction having a repeated execution.

e) Output

An algorithm should be capable of generating a result for implementation. For example, the following set of instruction is not an algorithm.

Step 1 Read x

Step 2 Set k = o

Step 3 Compute

$$Y = \frac{2e^x + 4x}{K}$$

Step 4 Output K

Since k = o, it becomes an impossible task to generate an output because step 3 cannot be divisible. It also tends to (intenite) as there is no reasonable termination.

2. **Coding of Program**

Coding the computer program is the act of writing the actual computer instructions. The programmer studies the previously prepared flowcharts and algorithm in order to be familiar or versed with the program to be written; so that the program can conform with the flowcharts B and algorithm.

3. **The Walk Through of a Program**

Once a program has been written, it is highly unlikely that it will run the first time because of the human errors during coding. A computer program is therefore usually checked by another programmer or supervisor for errors. Two major types of errors can be found in a typical program namely syntax and semantic errors.

Syntax errors are the assignment of the same symbol for different set of data. The omission of an instruction or the misuse of an operation symbol makes it impossible to produce an object program.

Semantic errors on the other hand refer to those condition in the program that do not adequately represent the data processing operations required within the program.

4. **Entry of Program into the Computer**

The most popular tool for program entry into the computer is the terminal. A computer terminal has a keyboard and a screen. The keyboard is used to type the program character by character and are displayed on the screen line by line. Error checking can be carried out line by line, otherwise, a program can be printed out on a computer sheet and text-checked for syntax and semantic errors.

5. **The Compilation of Source Program**

Compilation is the translation of a program code using a procedure-oriented language into machine language. The compilation process produced object program which performs the instructions originally represented by the code.

There are four main steps involved in the compilation of a source program to the object program. These are lexical analysis, syntax analysis, semantic analysis and code generation.

Figure 3.4.2 is a diagramatical representation of the compilation processes. The dotted arrows represent flow of information while solid arrows represent flow of program.

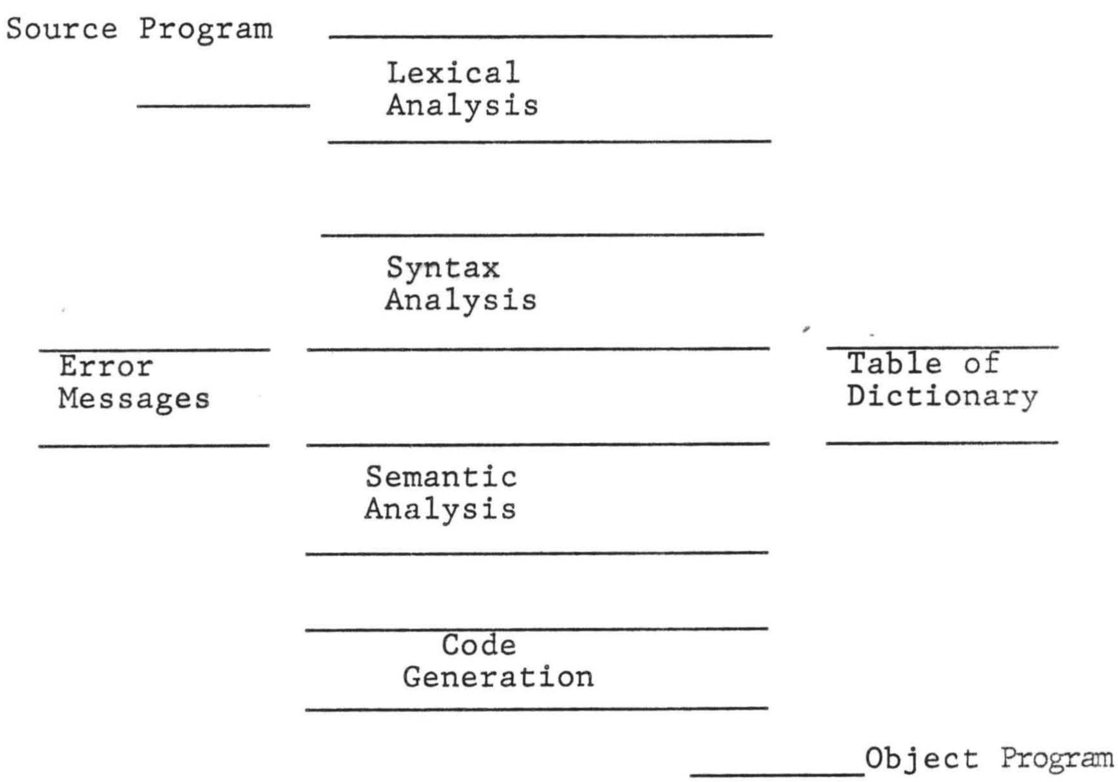


Figure 3.4.2 steps in compilation process

i) Lexical Analysis

The first step in the process of compilation is to break program into smaller basic unit usually called a token of programming language. This falls into one of the following.

a) Keywords eg DO, IF, BEGIN etc
b) Special Operators et t, c, x, - etc.
c) Delimeters eg comma, parenthesis, semicolon, etc
d) Identifiers or Variable names.
e) Literals, both numeric (eg 99) and string (eg 'Hello')
f) Separators eg blanks.

At the lexical analysis stage, a program called a scanner scans the characters of a source program from left to right and builds up the token of a program. The tokens are usually passed to the next phase of the compilation in an internal form. These internal form/representations and the actual values of tokens are passed to the next stage.

In addition, the scanner must recognise and delete blanks and redundant statements like comments. It is also the responsibility of the scanner to note and report lexical errors such as invalid character or improperly formed identifiers.

ii) Syntax Analysis

The syntax analysis stage is where a complete syntatic check is performed on the source program. The syntax of a programming language relates to the grammatical rules governing the formulation of legal statements in the language.
The table of dictionary is a table containing the identifier used in the source program along with their attributes. The attributes of an identifiers it types (eg integer, real, string) form (eg a constant simple variable) and other attributes relevant to the language. Each time an identifier is encountered, the symbol table is searched to

see whether the name has been previously entered. If the identifier is new, it is entered into the table. The information collected in a symbol table is used in semantic analysis to check whether the use of variable names are consistent with their explicit and implicit declaration. Syntax analyser also report syntatic errors such as multiple declaration of identifiers.

iii) Semantic Analysis

The semantic analysis deals with the interpretation of the meaning associated with a syntatic unit of the source program. When a parser has recognised a source language construct, the semantic analyser will check for semantic correctness and store necessary information about it into the symbol table or create an intermidiate form of source program for it. The intermidiate form of a source program depends largely on how it is to be manipulated later. These include:

a) Parsing and Parse Tree

Parsing is the process of analysing the program and extracting its syntax parts. A syntax analyser is also known as Parser. When we parse or draw a tree-like diagram form, the arithmetical expressions of such trees are called parse tree or syntax tree. Figure 3.4.3 shows the parse tree of the expression Y = b * c + a.

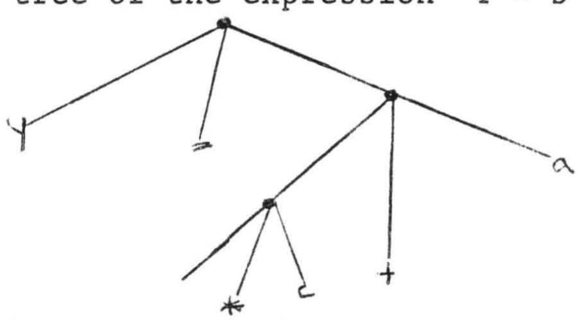


Figure 3.4.3 shows a parse tree.

b) Polish Notation

Polish Notation is used to represent arithmetic or logical expressions in a manner which specifies simply and exactly the order in which operators are to be evaluated. In this notation, the operator come immediately after the operands. The expression below is the corresponding polish notation for the parse tree in figure 3.4.3 above.

Infix Notation	Polish Notation
Y = b * c + 1	Ybc a * + =

c) Quadruple

A convinient form of a single binary operation is a quadruple. (operator, Operand 1, Operand 2,) specify the arguments and results of an operation. The expression below is the corresponding form of the parse tree in 3.4.3 above.

Y = b * X + a

This could be represented by the sequence.

*, b, c, T,
+, a, T, T_2
=, Y, T_2

Where T_1, T_2 are arbitrary temporary variable which the result is assigned by the Compiler.

4 Code Generation

The last step in a compilation process is the code generation. This is the actual translation of the intermediate form of source program into machine language. It can be generated in either symbolic language or machine language programs.

Code generation in symbolic assembly language is much easier than actual machine language, though it adds another extra step (assembly) to the process of compilation and the extra time required may be as long as the compilation itself.

The primary difference between this intermediate symbolic code and assembly code is that intermediate code need not specify the registers to be used for each operation word length or number of instructions. This can be illustrated by generating code for the assignment statement.

Y = b * c + a.

The intermediate form of this assignment statement in quadruple is the sequence as shown in 3(i) above. The symbolic code corresponding to this quadruple is given below.

```
LDA b
LUL c
STA T1
LDA T1
ADD a
STA T2
LDA T2
STA Y
```

During the intermediate code generation, it is possible to generate some redundant codes that their elimination does not have any adverse effect on the expected result. The removal of these redundant code is called code optimization.
To illustrate this, it will be observed that eight codes are generated for the statement.

Y = b * x + a

For instance, the third, the fourth, the sixth and the seventh codes are not necessary. This is because the content of the memory location b and c are already in the accumulator, all that is needed is to just add the content of memory location a to it.
This gives

```
LDA  b
MUL  c
ADD  a
STA  Y
```

This is a more efficient code than the first and could be called Machine - Independent optimization; though executed/done without regards to the feature of the machine.

6. **Program Test Run and Debugging**

The ultimate test of the object program is to determine how accurate its results are when processed on the computer. This can be determined by applying hypothetical data which are representation of the real world. Basically, the outputs are designed to test various branches and subroutines of the program.

The correct output of these hypothetical input transactions should be pre-determined.

Debugging routines are designed to eliminate costly computer time. The routines permit the programmer to search for errors on his desk. Once an error has been discovered, an investigation to localise the error begins (that is to find out which modules is causing it).

Debugging is the process of identifying those areas of the program which are in error so as to modify the errors to corrent them. The process of debugging involves firstly, locating those parts of the program code which are incorrect. Secondly, correct the error so that it meets its requirements. Program testing must be repeated to ensure that the change has been carried out correclt.y

The popular methods used to correct errors(debugging techniques) are as follows

a) Symbolic Dump Programs

It takes two forms - an interactive form or a batch form. In a batch form, the system lists the names of all program terminates. Interactive program for analysing the symbolic information allows the users to request the valve of individual variables by name so that only relevant values will be examined.

b) Program Trace Packages

Trace package provides information about the dynamic execution of the program; printing procedure entry and exit transfer of controls branch selection in IT statement after the program has been executed. Trace package involve instrumenting (space and speed) the program automatically so that relevant information can be collected.

c) Static Program Analyzer

A static program analyzer scans, text and searches for anomalities which are likely to result in errors in the program. It checks subroutines interfaces to ensure consistency with the routine declaration. They also locate common block errors and flag error - prone.

d) Dynamic Program Analyzer

This examines the outputs from a flow summariser, which identifies loops which do not terminate properly and allow the user to find sections of code which are wrongly executed.

7) **The Documentation Program**

This phase is not a step as in the preceeding ones. It is indeed an integral part of each of the other phases. It consists of gathering all documents associated with the program and placing them in a program manual. A typical program manual might include among others the following items:

- Problem Description
- Program Flowcharts
- Algorithms
- Input and Output Records
- File Formats
- Printed copy of the Program
- Copy of the Object Program.

CHAPTER FOUR

4.0 FILE PROCESSING AND MANAGEMENT SYSTEM

4.1 FILE DESCRIPTION

File Management system are designed so that the computer files they support are similar to the clerical files that they replace. Each file relates to 9 specific type of data in a specific application area.

File Management System (FMS) helps to create, store and access information efficiently using computers. It also enables data to be accessed conviniently and quickly. File Management System is very acceptable and yet simple to use in that:

i) Data can be retrieved in a number of ways.

ii) Data can be sorted in a sequence specified by the user and printed on a screen.

iii) It allows users to look at more than one file at a time

iv) It generates reports.

File Management System that caters for this degree of sophistication is frequently referred to as Database management.

A file is made up of a collection of records containing the same type of information. These records include a number of files (data items). These three constituents elements - files, records and fields exist in computer and clerical files. Files can be stores on floppy diskettes, hard disk drives, on tape or other storage media.

Each file name must be unique. The operating system does not allow two files with the same name to reside on the same diskette or the same sub-directory on a hardware. When a file is created with the same name as a previous file, it will overwrite the previous file.

A file name can be divided into 3 parts:

Filename. EXT

i) The first part is called the 'Filename' The file can be from 1 to 8 characters in length. "File Name" refers to the overall file while 'Filename' refers only to the first part of the file name.

ii) The second section of a file name is called 'extension' which is optional and can be up to 3 characters in length.

Extensions are used to organise and identify different types of files. Some programs add an extension automatically as data files are created.

iii) A period or dot is used to separate the filename and the extension so that DOS (Disk Operating System) knows where the filename ends and where the extension begins . It is however not visible when the directory is displayed. The valid characters for filenames and extensions include:

i) The letters A to Z
ii) The numbers 0 to 9
iii) The special characters and punctuation symbols.
$ # & ! ( ) - [[@

File names cannot contain spaces of the following characters; +, /, [ ], " :, ;. ,. ?, *, =, .
< >.

The following list contains some common extensions.

. BAK	A backup File
. BAS	Basic Program File
. CFG	Configuration file which saves program settings.
. COM	Program Files.
. DBF	A database file, established by Ashton-Tate for databse software.
. DOC	Document File
. HLP	Help File which usually contains documentation associated with a specific software product.
. NDX	Index File usually associated with Dbase software program.
. SYS	System Files also device drivers. A device driver resembles a translator it contains instructions that let DOS know how to work with peripherals.
. WK1	Worksheet File, established by lotus Development Corporation

There are several types of files.

a) A data file is created by a program to contain the information it manages. Most data files can only be used from the program that created them. The contents of a data file are not usually readable and would not make sense if not seen through the program that created them. In order to see the contents of a data file, the program has to be run.

b) A program or command file performs functions or tasks and is written in computer language, and only the computer may be able to decipher the contents of the files. Program files are usually indicated by the extension . COM or .EXE. Users can create codes that can be used to organise and keep files straight. Generally, file names state what is in a file, and try to relate the file's name to something relevant about the file.

To make the eight characters in a filename relevant as much as possible:

i) Use as few letters as possible to identify the file's group. For example: B for Budget, F for Forcast, R for Report.

ii) Include a date only for files that get updated regularly.

iii) Monthly versions can be identified by 1 through 6 for January to June and J through D for July to December. For example in the file called F 12 GS.B, it could depict that:

F stands for February, followed by Modany's date, then the salesperson's initials.

B Stands for Budget.

There is no definite method through which files can be organised, files can be grouped by client's names, month, types of transactions; although there is need to maintain a great degree of consistency.

4.2 **LOGICAL FILE ORGANIZATION**

The way a file is organised depends on the storage medium used and the way the file is processed. The need for the presence of more than one file organisation arises from a number of reasons.

All current physical storage media imposed a physical sequence on the stored records of a file. This physical structured is very often incompatible with the logical view of the user, and therefore, a way must be found for mapping the logical requirement on to the imposed rigid physical structure.
In addition, the data processing requirements of different users differ considerably in most cases, which means that a file organisation for A may be completely useless for the needs of users B. Criteria that affect the choice of a file organisation are operations, that are to be performed on the storage data, such as, how fast a record needs to be retrieve.

The objective of any designer is to organise a file in such a way as to give a user the facilities that is required while using minimum computer resources, such as time and storage. One important criterion in the designer's decision would by the file activity which is the percentage of records updated on each computer run.

There are 3 ways in which files can be logically organised.

i) **Serial File Organisation**
A serial file organisation refers to a file with the simplest possible organisation of record within it. Records are stored in physical adjacent location on a storage medium without references to any particular sequence.

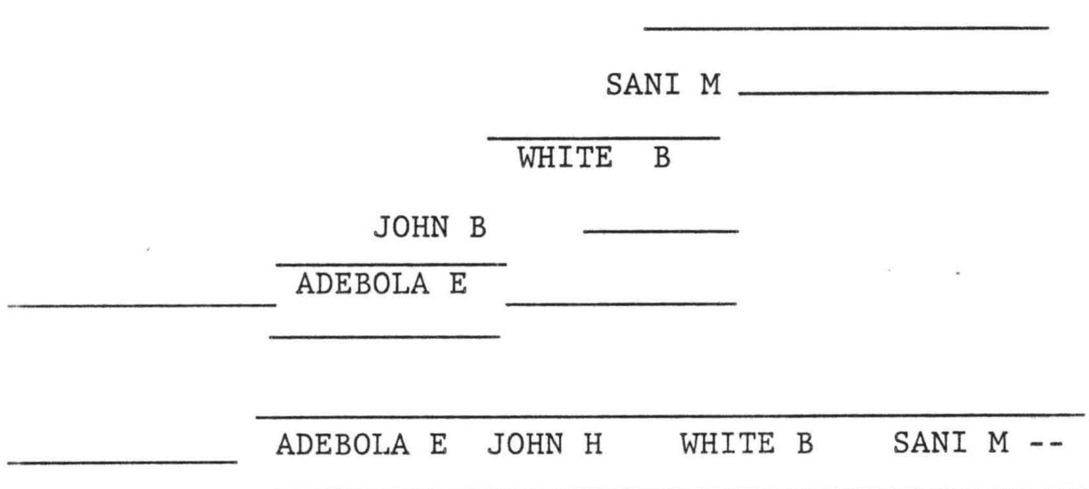


Figure 4.2.1 Serial File Organisation.

There are no conditions imposed on the storing of each record except the availability of free space in the medium for storing this record.

ii) **Sequential File Organisation**

A sequential File Organisation is simply a serial file having its records sorted out in certain order, depending on a sort key. Like the serial file organisation, records are stored physically one after the other in sequential file organisation except that they are logically ordered on a key. Unlike the serial files however, sequential files imposed the condition of some ordering of records depending on the sort key.

ADEBOLA E	JOHN H	WHITE B	SANI M	--------

S O R T I N G

JOHN H	SANI M	ADEBOLA E	WHITE B	---------

Files organised sequentially can be updated considerably faster than those with other organisation especially when majority of their records are to be updated.

iii) **Random File Organisation**

Records in a randomly organised files do not need to be stores sequentially rather they may be stored in a random sequence. Each storage location on the storage medium has an address and it is this address that is used in order to access a record stored on that particular location. In random file organisation unlike serial or sequential organisation, records can be accessed directly regardless of whether or not the previous records have been accessed.

4.3 FEATURES OF FILE MANAGEMENT

File Management Systems allow users to design the structure of records (formatting); validate data as it is inputed into the computer (data entry and validation); store the data on disk (data storage), allow data to be accessed in the format required, sorted, and selected as required (retrieval and printed off/out (reporting).

a) Formatting

Before data can be entered into a file management system, the user must decide on the type of file, and data that go into them. For example, data relating to customers is normally put together in a customer's file while product data are stored in product file etc.

The file management systems will require details of fields. These details could include:

i) Length of Characters
ii) Format (numeric, alpha numeric, character, date)
iii) Password etc.

b) Data Entry and Validation

Once the fields are formatted, the data has to be keyed in and validated before they are added to the master file. The purpose of data validation is to ensure that no invalid data enters the databse. Most data can be input using the conventional keyboard.

c) Data Storage

Most file management systems require disk storage. Floppy disks are presently the most common choice, but hard disks are faster, more reliable and have greater capacities.

d) Retrieval

Ideally, many users will require processing via the key field of a file. Sorting facilities are important to enable the output of the errors to be made in the order of the keys and therefore in a different sequence from the one in which they are stored.

Many systems offer browsing facilities which list the records in a file in the quickest way possible and allow the user to obtain information quickly. Simple searches may be achieved from the user's point of view by processing through a series of menus. A menu contains a number of options and the users choose the particular option required.

e) Reporting

Most file management systems offer facilities for settting up and print reports using the data held in the database. The reports could be discplayed on the Video Display Unit (VDU) as well as printed on paper. Part of the reports include:

i) Heading

On each hardcopy or UDU, there will be headings which help the readers to understand the contents of the report and give other information.

ii) Main Body

The main body of the report consists of the raw data which are extracted from the database.

iii) Total and Summary

This is the last part of designing a report. It is to set up the various total fields. These will include the subtotals, grandtotals and other information which summariese the report.

4.4 COMPUTER VIRUSES AND THEIR EFFECTS

A computer virus is a program (or instruction hidden with a program) that can "infect" files and programs on a computer. Unlike most other programs, viruses are specifically designed to spread themselves. Some virus can display a message or cause erratic screen behaviour. Others are destructive, erase or damage files or can overload the memory or communication networks.

The various/different types of virus are:

i) **Boot Sector Viruses**

These are perhaps the most common kinds of virus. They infect the boot sectors of floppy disks and either the Master Boot Record (MBR) or the Dos Boot Record or Dos Boot Sector (DBR) of hard disks.

ii) **TSR File Viruses**

The Terminate and Stay Resident (TSR) File Virus is another common type of virus. As the name denotes, it infects files especially the COM and EXE files and their are some device driver viruses that infect overlay files. TSR virus spreads when users run an infected program. It spreads to the memory resident and infect each program run if such program has not been infected already.

iii) **Overwriting Viruses**

An overwrite virus simply overwrites each file it infects with itself and causes the program to cease to function. Though as overwriting viruses are glaringly obvious, they can not successfully spread.

iv) **Miscellaneous Objects of Infection**

There is a virus that infects OBJ files. There is also another virus (DIR II) that infects file systems by changing the directories so that files on the hard disks are all cross linked to the virus.

The most likely routes by which a virus gets into an organisation are engineers and parents.

a) Hardware Engineers visit a large number of computer and like the busy bee, could pick up some pollen here and deposit it in another place. Hardware engineers do not have all their software diskettes protected, thus causing some infection on other programs.

b) Parents have children and if there is a personal computer (PC) at home, these children often quite possibly swap software at school. The diskettes the children bring home might well be infected and if the parents take the infected diskette to work, this will result in the spread of the virus which will however affect his programs at work.

c) A boot sector virus could arrive on a data diskette from a colleague.

d) Some purchased hardware that come with diskettes containing drivers.

e) Salesmen running demos could unknowingly install the virus they picked up from the last place they ran their demo.

Some of the damages done by viruses include:

i) **Trivial Damage**
This is done by a virus such as form virus. On the 18th of every month, each key that users hit the speake beep. All that has to be done is to get off the virus.

ii) **Minor Damage**
A good example is the way that Jerusalem virus deletes any program that is being run after the virus has gone to the memory resident on Friday the thirteeenth. At worst, the deleted program have to be installed so that the damage is unlikely to be more than 30 minutes per computer.

iii) **Moderate Damage**
This is a situation where virus hits the backups as well as the harddisks. It might take a few weeks before a clean data files can be found and when a six weeks old

backup is restored, it might be difficult to find the original document to work from.

iv) **Severe Damage**

Severe Damage is done when a virus makes gradual and progressive changes (so that backups are also corrupted), but the changes are not obvious. Users therefore wind up simply not knowing whether the data is correct or changed.

v) **Unlimited Damage**

Some viruses aim at getting the system manager password and pass it along to a third party. The damage is them by the third party, who can log in to the system and do anything he likes.

It is however recommended that everything be virus checked before it is used and such checks include used of floppy disks with a data as well as software. This could be done by using a scanner such as finds virus which can be installed on every computer.

Alternatively, a virus guard can be installed everything is automatically scanned without the users being aware of it.

4.5 PROPER CARE AND COMPUTER PROCEDURE

a) Ensure that there are backups for all data and store the backup and the original program disks in a fore proof box. Write 'protect your original program disk'.

b) Identify and label your diskettes.

c) Insert disk slowly with label facing up and out towards the door handle. Handle it by the edge only and avoid to touch the open portion of the floppy.

d) Store diskettes in a cool, dry environment and keep it away from excessive heat, cold, smoke or direct sunlight.

e) Avoid to remove disk from drive when red or green light is on.

f) Save your work frequently to disk for about every 15-30 minutes,

g) Avoid to turn off the system unless the AT or CT system prompt is displayed. Remove the plug of the computer and modem when leaving the office or home for an extended period of time.

h) Where there is need to transport harddisk from one location to another, and it is not a portable unit, the drive heads might need to be packed.

i) Leaving your system running prevents damage and prolong the computer's life. Turning the system on and off causes fluctuations in the temperature of the system. The fluctuations result in expansion and contraction of the metal, plastics and even the harddisk of your system.

j) Make sure nothing blocks the air intake vents such as ribbon cables. This allows more air movement in the system.

k) Keep a static-free dust cover over the keyboard and printer and invest in an anti-static mat if static electricity from carpet is a problem in your office.

CHAPTER FIVE

5.0 CONCLUSIONS AND RECOMMENDATION

In order to achieve the set and desired goals of computerization, there is need to effectively co-ordinate the scarce resources of the environment, people and computers. This can be achieved by:

i) Linking and reporting specific functions of the computers to other specific department of the organisation.

ii) Periodic evaluations of the functions of the computers in view to expand, control and for security.

iii) Collecting user views, complaints and conflicts.

iv) Putting user's views into action after acceptable assessment.

v) Resolving users conflicts and responding to complaints positively.

vi) Establishing user's news bulletin or common forum for user development.

vii) Computer Installation securities :
- Physical building security
- Staff Physical security
- Computers physical security
- Periodic security checkups.

viii) Computer installation policies on:
a) Computer Usage
b) External Memory Usage
c) Staff Functional Schedules
d) Computer building entrances on:
   - Computer room entrances
   - Computer staff rooms entrances

e) Computer Maintenance
   - employ maintenance engineers on full time service or contract maintenance engineers cost is the decider.
   - the danger of retaining manufacturers as your maintenance engineer.
   - the advantages of keeping manufacturers as your maintenance engineers.

f) Computer staff.

- encourage conference attendance.
- encourage inter-installation visits.
- encourage staff training on both short and long time.
- always have a trainee
- encourage staff research and development.

The computer installation organisational structure must be constantly overhauled to check redundancies and conflicts of authorities. The management of computer accessories and stationeries must have standards and accessment criterion for the accessories in terms of compability with user's computer system.
However, there are limitations under which computer components work. These limitations should not be allowed or exceeded to degenerate into the lower limit or grow into the upper limits. These limitations include:

i) Owner limitations as to what the computer can do for them. There is an agreed workload before computers are chosen for purchase and installation.

ii) There are limitations to the Computer installation as a whole. No computer can do everything. It is not possible to assemble every kind or type or level of computers there in the world into one computer installation.

iii) Computers have life span which once reached exhaustion has overpowered it into permanent packing.

iv) Computer staff should not be seen attempting computer usage beyond their knowledge can allow them.

Finally, having computer installation entrusted into one's hand is like entrusting the lives of the populace in one's hand. Anything that happens to the computers happens to the data inside it. The data inside the computers are information which are often vital to human life. It is like a situation whereby whatever happens to a car happens to its occupants/passengers and this is often not healthy.

REFERENCES

1. Adelman, L. (1992) - Evaluating decision Support and Expert Systems, John Willey and Sons, New York.

2. Boland, R. J., - and Hirschheim, R.A. (1993) - critical issues in information systems research, John Willey and Sons, New York.

3. Clare, C.P., - and Loucopoulas (1983) - Data Processing Current theories and practices, Abacus Press, England.

4. Edward, J. L. (1979)- Computers, Automation and Society, Richard Irwin, INC. Geogetown, Ontario.

5. FOY, N (1972) - Computer Management, Auerbach publishers, New York.

6. Garcia, R. R.(1988) - Human Factors in Systems development, NCC Publications, Manchester England.

7. Geoff, S. (1989) - Viruses, Bugs and Star Wars, NCC Blackwell, Manchester, England.

8. Girish, P. (1984) - Programmer Productivity, Reston Publications, Reston, Virgina.

9. Sergeev, A. - and Vashkevich, N. (1976) - An introduction to computer, MIR Publishers, MOSCOW.

10. William, S. (1987) - Computer Organisation and architecture, Macmillan Publishing Company, New York.

APPENDIX 1
(SCREEN DESIGN AND OUTPUT)

COMPUTERISED ADDRESS BOOK

A .... ADDRESS DETAILS

B .... ADDRESS LISTING

C .... REPORT PRODUCTION

Q .... EXIT

SELECT YOUR CHOICE:

ADDRESS DETAIL OPTION

A .... NEW ADDRESS DETAIL

B .... VIEW ADDRESS DETAIL

C .... CHANGE ADDRESS DETAIL

D .... CANCEL ADDRESS DETAIL

Q .... EXIT

SELECT YOUR CHOICE:

ADD ADDRESS DETAIL

SERIAL NO (PRESS XXXX TO EXIT): 0004

OLADIMEJI OLAYINKA
SURNAME OTHER NAMES

ADDRESS: ADVANCE MICROCOMPUTER SYSTEM

NO 12 BAYAGIDA

KADUNA

STATE: KADUNA PHONE: 062/238464

SAVE ADDRESS DETAIL (Y/N):

VIEW ADDRESS DETAIL

SERIAL NO (PRESS XXXX TO EXIT): 0004

OLADIMEJI OLAYINKA
SURNAME OTHER NAMES

ADDRESS: ADVANCE MICROCOMPUTER SYSTEM

NO 12 BAYAGIDA

KADUNA

STATE: KADUNA PHONE: 062/238464

PRESS ANY KEY TO CONTINUE

CHANGE ADDRESS DETAIL

SERIAL NO (PRESS XXXX TO EXIT): 0004

OLADIMEJI   OLAYINKA
SURNAME   OTHER NAMES

ADDRESS: ADVANCE MICROCOMPUTER SYSTEM

NO 12 BAYAGIDA

KADUNA

STATE: KADUNA   PHONE: 062/238464

SAVE CHANGES (Y/N):

CANCEL ADDRESS DETAIL

SERIAL NO (PRESS XXXX TO EXIT): 0004

OLADIMEJI OLAYINKA
SURNAME OTHER NAMES

ADDRESS: ADVANCE MICROCOMPUTER SYSTEM

NO 12 BAYAGIDA

KADUNA

STATE: KADUNA PHONE: 062/238464

REMOVE ADDRESS DETAIL (Y/N):

ADDRESS LISTING

SERIAL NO	NAME	STATE	PHONE
0001	ATTEH S.	NIGER	066/221984
0002	IBRAHIM A.	NIGER	066/223344
0003	DANTANI M.	NIGER	066/226633
0004	OLADIMEJI O.	KADUNA	062/238464

PRESS ANY KEY TO EXIT

GENERAL LISTING OF ADDRESS BOOK AS AT 23/02/99

SERIAL NO	NAME	STATE	PHONE
0001	ATTEH S.	NIGER	066/221984
0002	IBRAHIM A.	NIGER	066/223344
0003	DANTANI M.	NIGER	066/226633
0004	OLADIMEJI O.	KADUNA	062/238464

APPENDIX 2
(PROGRAM LISTING)

ADDRESS.PRG

```
SET TALK OFF
SET SCOREBOARD OFF
SET STATUS OFF
DO WHILE .T.
  CLEAR
  C = SPACE(1)
  @ 3,15 TO 20,64 DOUBLE
  @ 5,27 SAY "COMPUTERISED ADDRESS BOOK"
  @ 6,27 TO 6,51
  @ 16,16 TO 16,63
  @ 8,22 SAY "A ............... ADDRESS DETAILS"
  @ 10,22 SAY "B ............... ADDRESS LISTING"
  @ 12,22 SAY "C ............... REPORT PRODUCTION"
  @ 14,22 SAY "Q ............... EXIT"
  @ 18,22 SAY "          SELECT YOUR CHOICE:" GET C
  READ
  DO CASE
    CASE UPPER(C) = 'A'
      DO DETAIL
    CASE UPPER(C) = 'B'
      DO LISTING
    CASE UPPER(C) = 'C'
      DO REPORT
    CASE UPPER(C) = 'Q'
      EXIT
  ENDCASE
ENDDO
CLEAR
RETURN
```

DETAIL.PRG

```
DO WHILE .T.
  CLEAR
  C = SPACE(1)
  @ 3,15 TO 22,64 DOUBLE
  @ 5,29 SAY "ADDRESS DETAIL OPTION"
  @ 6,29 TO 6,49
  @ 18,16 TO 18,63
  @ 8,22 SAY "A .......... NEW ADDRESS DETAIL"
  @ 10,22 SAY "B .......... VIEW ADDRESS DETAIL"
  @ 12,22 SAY "C .......... CHANGE ADDRESS DETAIL"
  @ 14,22 SAY "D .......... CANCEL ADDRESS DETAIL"
  @ 16,22 SAY "Q .......... EXIT"
  @ 20,22 SAY "          SELECT YOUR CHOICE:" GET C
  READ
  DO CASE
    CASE UPPER(C) = 'A'
      DO ADD
    CASE UPPER(C) = 'B'
      DO VIEW
    CASE UPPER(C) = 'C'
      DO CHANGE
```

```
    CASE UPPER(C) = 'D'
      DO CANCEL
    CASE UPPER(C) = 'Q'
      EXIT
  ENDCASE
ENDDO
CLEAR
RETURN
```

ADD.PRG

```
USE ADDRESS
DO WHILE .T.
  CLEAR
  @ 1,15 TO 23,64
  @ 3,31 SAY 'ADD ADDRESS DETAIL'
  @ 4,31 TO 4,48
  @ 19,16 TO 19,63
  MSERIAL=SPACE(4)
  MSURNAME=SPACE(15)
  MOTHERNAME=SPACE(25)
  MADDRESS1=SPACE(35)
  MADDRESS2=SPACE(35)
  MADDRESS3=SPACE(35)
  MSTATE=SPACE(15)
  MPHONE=SPACE(11)
  @ 6,18 SAY 'SERIAL NO (PRESS XXXX TO EXIT):' GET MSERIAL
PICTURE 'XXXX'
  READ
  IF MSERIAL='XXXX'
    EXIT
  ENDIF
  @ 8,18 GET MSURNAME PICTURE '@!'
  @ 8,37 GET MOTHERNAME PICTURE '@!'
  @ 9,22 SAY 'SURNAME'
  @ 9,42 SAY 'OTHER NAMES'
  @ 11,18 SAY 'ADDRESS:' GET MADDRESS1 PICTURE '@!'
  @ 13,27 GET MADDRESS2 PICTURE '@!'
  @ 15,27 GET MADDRESS3 PICTURE '@!'
  @ 17,18 SAY 'STATE:' GET MSTATE PICTURE '@!'
  @ 17,44 SAY 'PHONE:' GET MPHONE PICTURE '999/9999999'
  READ
  CHOICE=SPACE(1)
  @ 21,26 SAY 'SAVE ADDRESS DETAIL (Y/N):' GET CHOICE PICTURE
'!'
  READ
  IF CHOICE='Y'
    APPEND BLANK
    REPLACE SERIAL WITH MSERIAL,SURNAME WITH MSURNAME
    REPLACE OTHERNAME WITH MOTHERNAME,ADDRESS1 WITH MADDRESS1
    REPLACE ADDRESS2 WITH MADDRESS2,ADDRESS3 WITH MADDRESS3
    REPLACE STATE WITH MSTATE,PHONE WITH MPHONE
  ENDIF
ENDDO
USE
```

```
CLEAR
RETURN
```

VIEW.PRG

```
USE ADDRESS
DO WHILE .T.
  CLEAR
  @ 1,15 TO 23,64
  @ 3,30 SAY 'VIEW ADDRESS DETAIL'
  @ 4,30 TO 4,48
  @ 19,16 TO 19,63
  MSERIAL=SPACE(4)
  @ 6,18 SAY 'SERIAL NO (PRESS XXXX TO EXIT):' GET MSERIAL
PICTURE 'XXXX'
  READ
  IF MSERIAL='XXXX'
    EXIT
  ENDIF
  LOCATE FOR SERIAL=MSERIAL
  MSURNAME=SURNAME
  MOTHERNAME=OTHERNAME
  MADDRESS1=ADDRESS1
  MADDRESS2=ADDRESS2
  MADDRESS3=ADDRESS3
  MSTATE=STATE
  MPHONE=PHONE
  @ 8,18 GET MSURNAME PICTURE '@!'
  @ 8,37 GET MOTHERNAME PICTURE '@!'
  @ 9,22 SAY 'SURNAME'
  @ 9,42 SAY 'OTHER NAMES'
  @ 11,18 SAY 'ADDRESS:' GET MADDRESS1 PICTURE '@!'
  @ 13,27 GET MADDRESS2 PICTURE '@!'
  @ 15,27 GET MADDRESS3 PICTURE '@!'
  @ 17,18 SAY 'STATE:' GET MSTATE PICTURE '@!'
  @ 17,44 SAY 'PHONE:' GET MPHONE PICTURE '999/9999999'
  CLEAR GETS
  @ 21,27 SAY 'PRESS ANY KEY TO CONTINUE'
  SET CONSOLE OFF
  WAIT
  SET CONSOLE ON
ENDDO
USE
CLEAR
RETURN
```

CHANGE.PRG

```
USE ADDRESS
DO WHILE .T.
  CLEAR
  @ 1,15 TO 23,64
  @ 3,29 SAY 'CHANGE ADDRESS DETAIL'
  @ 4,29 TO 4,49
```

```
  @ 19,16 TO 19,63
  MSERIAL=SPACE(4)
  @ 6,18 SAY 'SERIAL NO (PRESS XXXX TO EXIT):' GET MSERIAL
PICTURE 'XXXX'
  READ
  IF MSERIAL='XXXX'
    EXIT
  ENDIF
  LOCATE FOR SERIAL=MSERIAL
  MSURNAME=SURNAME
  MOTHERNAME=OTHERNAME
  MADDRESS1=ADDRESS1
  MADDRESS2=ADDRESS2
  MADDRESS3=ADDRESS3
  MSTATE=STATE
  MPHONE=PHONE
  @ 8,18 GET MSURNAME PICTURE '@!'
  @ 8,37 GET MOTHERNAME PICTURE '@!'
  @ 9,22 SAY 'SURNAME'
  @ 9,42 SAY 'OTHER NAMES'
  @ 11,18 SAY 'ADDRESS:' GET MADDRESS1 PICTURE '@!'
  @ 13,27 GET MADDRESS2 PICTURE '@!'
  @ 15,27 GET MADDRESS3 PICTURE '@!'
  @ 17,18 SAY 'STATE:' GET MSTATE PICTURE '@!'
  @ 17,44 SAY 'PHONE:' GET MPHONE PICTURE '999/9999999'
  READ
  CHOICE=SPACE(1)
  @ 21,29 SAY 'SAVE CHANGES (Y/N):' GET CHOICE PICTURE '!'
  READ
  IF CHOICE='Y'
    REPLACE SERIAL WITH MSERIAL,SURNAME WITH MSURNAME
    REPLACE OTHERNAME WITH MOTHERNAME,ADDRESS1 WITH MADDRESS1
    REPLACE ADDRESS2 WITH MADDRESS2,ADDRESS3 WITH MADDRESS3
    REPLACE STATE WITH MSTATE,PHONE WITH MPHONE
  ENDIF
ENDDO
USE
CLEAR
RETURN
```

CANCEL.PRG

```
USE ADDRESS
DO WHILE .T.
  CLEAR
  @ 1,15 TO 23,64
  @ 3,29 SAY 'CANCEL ADDRESS DETAIL'
  @ 4,29 TO 4,49
  @ 19,16 TO 19,63
  MSERIAL=SPACE(4)
  @ 6,18 SAY 'SERIAL NO (PRESS XXXX TO EXIT):' GET MSERIAL
PICTURE 'XXXX'
  READ
  IF MSERIAL='XXXX'
    EXIT
```

```
  ENDIF
  LOCATE FOR SERIAL=MSERIAL
  MSURNAME=SURNAME
  MOTHERNAME=OTHERNAME
  MADDRESS1=ADDRESS1
  MADDRESS2=ADDRESS2
  MADDRESS3=ADDRESS3
  MSTATE=STATE
  MPHONE=PHONE
  @ 8,18 GET MSURNAME PICTURE '@!'
  @ 8,37 GET MOTHERNAME PICTURE '@!'
  @ 9,22 SAY 'SURNAME'
  @ 9,42 SAY 'OTHER NAMES'
  @ 11,18 SAY 'ADDRESS:' GET MADDRESS1 PICTURE '@!'
  @ 13,27 GET MADDRESS2 PICTURE '@!'
  @ 15,27 GET MADDRESS3 PICTURE '@!'
  @ 17,18 SAY 'STATE:' GET MSTATE PICTURE '@!'
  @ 17,44 SAY 'PHONE:' GET MPHONE PICTURE '999/9999999'
  CLEAR GETS
  CHOICE=SPACE(1)
  @ 21,25 SAY 'REMOVE ADDRESS DETAIL (Y/N):' GET CHOICE
PICTURE '!'
  READ
  IF CHOICE='Y'
    DELETE
    PACK
  ENDIF
ENDDO
USE
CLEAR
RETURN


LISTING.PRG

CLEAR
USE ADDRESS
@ 0,32 SAY 'ADDRESS LISTING'
@ 1,32 TO 1,46
@ 2,8 TO 24,71
@ 3,11 SAY 'SERIAL NO'
@ 3,25 SAY 'NAME'
@ 3,43 SAY 'STATE'
@ 3,60 SAY 'PHONE'
@ 4,9 TO 4,70
@ 22,9 TO 22,70
ROW=4
DO WHILE .NOT. EOF()
  ROW=ROW+1
  MSERIAL=SERIAL
  MSURNAME=SURNAME
  MOTHERNAME=OTHERNAME
  MNAME=RTRIM(MSURNAME)+' '+LEFT(MOTHERNAME,1)+'.'
  MSTATE=STATE
  MPHONE=PHONE
  @ ROW,13 SAY MSERIAL
```

```
  @ ROW,22 SAY MNAME
  @ ROW,41 SAY MSTATE
  @ ROW,58 SAY MPHONE
  SKIP
ENDDO
@ 23,29 SAY 'PRESS ANY KEY TO EXIT'
SET CONSOLE OFF
WAIT
SET CONSOLE ON
USE
CLEAR
RETURN
```

REPORT.PRG

```
CLEAR
USE ADDRESS
SET DEVICE TO PRINTER
@ 0,17 SAY 'GENERAL LISTING OF ADDRESS BOOK AS AT
'+DTOC(DATE())
@ 1,17 SAY REPLICATE ('=',46)
@ 2,8 SAY REPLICATE ('=',64)
@ 3,8 SAY '|'
@ 3,11 SAY 'SERIAL NO'
@ 3,25 SAY 'NAME'
@ 3,43 SAY 'STATE'
@ 3,60 SAY 'PHONE'
@ 3,71 SAY '|'
@ 4,8 SAY '|'
@ 4,9 SAY REPLICATE ('=',62)
@ 4,71 SAY '|'
ROW=4
DO WHILE .NOT. EOF()
  ROW=ROW+1
  @ ROW,8 SAY '|'
  @ ROW,71 SAY '|'
  MSERIAL=SERIAL
  MSURNAME=SURNAME
  MOTHERNAME=OTHERNAME
  MNAME=RTRIM(MSURNAME)+' '+LEFT(MOTHERNAME,1)+'.'
  MSTATE=STATE
  MPHONE=PHONE
  ROW=ROW+1
  @ ROW,8 SAY '|'
  @ ROW,13 SAY MSERIAL
  @ ROW,22 SAY MNAME
  @ ROW,41 SAY MSTATE
  @ ROW,58 SAY MPHONE
  @ ROW,71 SAY '|'
  SKIP
ENDDO
ROW=ROW+1
@ ROW,8 SAY '|'
@ ROW,71 SAY '|'
ROW=ROW+1
```

```
@ ROW,8 SAY REPLICATE ('=',64)
SET DEVICE TO SCREEN
USE
CLEAR
RETURN
```