UNIVERSITY OF CROSS RIVER STATE

FACULTY OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

COURSE TITLE: COMPUTER PROGRAMMING LANGUAGE

COMPUTER SOFTWARE

1.1 Computer Software

The physical components of the computer are called the hardware while all the other resources or parts of the computer that are not hardware, are referred to as the software. The software is the set of programs that make the computer system active. In essence, the software is the set of programs that run on the computer.

Then, what is a program? A program is a series of coded instructions showing the logical steps the computer follows to solve a given problem.

1.2 Classification of Computer Software

The computer software could be divided into two major groups, namely system software (programs), and application software (programs).

1.2.1 System Software

This refers to the suits of programs that facilitate the optimal use of the hardware systems and/or provide a suitable environment for the writing, editing, debugging, testing and running of user programs. Usually, every computer system comes with a collection of these suits of programs which are provided by the hardware manufacturer.

1.2.2 Operating Systems

An operating system is a program that acts as an interface between a user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user may execute programs.

The operating system is the first component of the systems programs that interest us here. Systems programs are programs written for direct execution on computer hardware in order to make the power of the computer fully and efficiently accessible to applications programmers

and other computer users. Systems programming is different from application programming because the former requires an intimate knowledge of the computer hardware as well as the end users' needs. Moreover, systems programs are often large and more complex than application programs, although that is not always the case. Since systems programs provide the foundation upon which application programs are built, it is most important that systems programs are reliable, efficient and correct.

In a computer system the hardware provides the basic computing resources. The applications programs define the way in which these resources are used to solve the computing problems of the user. The operating system controls and coordinates the use of the hardware among the various systems programs and application programs for the various users.

The basic resources of a computer system are provided by its hardware, software and data. The operating system provides the means for the proper use of these resources in the operation of the computer system. It simply provides an environment within which other programs can do useful work.

We can view an operating system as a resource allocator. A computer system has many resources (hardware and software) that may be required to solve a problem: CPU time, memory space, file storage space, input/output devices, etc.

The operating system acts as the manager of these resources and allocates them to specific programs and users as necessary for their tasks. Since there may be many, possibly conflicting, requests for resources, the operating system must decide which requests are allocated resources to operate the computer system fairly and efficiently. An operating system is a control program. This program controls the execution of user programs to prevent errors and improper use of the computer.

Operating systems exist because they are a reasonable way to solve the problem of creating a usable computing system. The fundamental goal of a computer system is to execute user programs and solve user problems.

The primary goal of an operating system is convenience for the user. Operating systems exist because they are supposed to make it easier to compute with an operating system than without an operating system. This is particularly clear when you look at the operating system for small personal computers.

A secondary goal is the efficient operation of computer system. This goal is particularly important for large, shared multi-user systems. Operating systems can attain this goal. It is known that sometimes these two goals, convenience and efficiency, are contradictory.
While there is no universally agreed definition of the concept of an operating system, we offer the following as a reasonable starting point:

A computer's operating system (OS) is a group of programs designed to serve two basic purposes:

- To control the allocation and use of the computing system's resources among the various users and tasks.
- To provide an interface between the computer hardware and the programmer that simplifies and makes feasible the creation, coding, debugging, and maintenance of application programs.

Specifically, we can imagine that an effective operating system should accomplish all of the following:

- Facilitate creation and modification of program and data files through an editor program.
- Provide access to compilers to translate programs from high-level languages to machine language.
- Provide a loader program to move the complied program code to the computer's memory for execution.
- Provide routines that handle the intricate details of I/O programming.
- Assure that when there are several active processes in the computer, each will get fair and non-interfering access to the central processing unit for execution.
- Take care of storage and device allocation.
- Provide for long term storage of user information in the form of files.
- Permit system resources to be shared among users when appropriate, and be protected from unauthorised or mischievous intervention as necessary.

Though systems programs such as editor and translators and the various utility programs (such as sort and file transfer program) are not usually considered part of the operating system, the operating system is responsible for providing access to these system resources.

1.3 Types of Operating Systems

Modern computer operating systems may be classified into three groups, which are distinguished by the nature of interaction that takes place between the computer user and his or her program during its processing. The three groups are called batch, time-shared and real time operating systems.

1.3.1 Batch Processing Operating System

In a batch processing operating system environment users submit jobs to a central place where these jobs are collected into a batch, and subsequently placed on an input queue at the computer where they will be run. In this case, the user has no interaction with the job during its processing, and the computer's response time is the turnaround time - the time from submission of the job until execution is complete, and the results are ready for return to the person who submitted the job.

1.3.2 Time Sharing Operating System

Another mode for delivering computing services is provided by time sharing operating systems. In this environment a computer provides computing services to several or many users concurrently on-line. Here, the various users are sharing the central processor, the memory, and other resources of the computer system in a manner facilitated, controlled, and monitored by the operating system. The user, in this environment, has nearly full interaction with the program during its execution, and the computer's response time may be expected to be no more than a few seconds.

1.3.3 Real Time Operating System

The third class of operating systems, the real time operating systems, are designed to service those applications where response time is of the essence in order to prevent error, misrepresentation or even disaster. Examples of real time operating systems are those which handle airlines reservations, machine tool control, and monitoring of a nuclear power station. The systems, in this case, are designed to be interrupted by external signal that require the immediate attention of the computer system.

In fact, many computer operating systems are hybrids, providing for more than one of these types of computing service simultaneously. It is especially common to have a background batch system running in conjunction with one of the other two on the same computer.

A number of other definitions are important towards gaining an understanding of operating systems:

1.3.4 Multiprogramming Operating System

A multiprogramming operating system is a system that allows more than one active user program (or part of user program) to be stored in main memory simultaneously.
Thus, it is evident that a time-sharing system is a multiprogramming system, but note that a multiprogramming system is not necessarily a time-sharing system. A batch or real time operating system could, and indeed usually does, have more than one active user program simultaneously in main storage. Another important, and all too similar, term is ‘multiprocessing’.

A multiprocessing system is a computer hardware configuration that includes more than one independent processing unit. The term multiprocessing is generally used to refer to large computer hardware complexes found in major scientific or commercial applications.

A networked computing system is a collection of physically interconnected computers. The operating system of each of the interconnected computers must contain, in addition to its own stand- alone functionality, provisions for handing communication and transfer of program and data among the other computers with which it is connected.

A distributed computing system consists of a number of computers that are connected and managed so that they automatically share the job processing load among the constituent computers, or separate the job load as appropriate to particularly configured processors. Such a system requires an operating system which, in addition to the typical stand- alone functionality, provides coordination of the operations and information flow among the component computers.

The networked and distributed computing environments and their respective operating systems are designed with more complex functional capabilities. In a network operating system the users are aware of the existence of multiple computers, and can log in to remote machines and copy files from one machine to another. Each machine runs its own local operating system and has its own user (or users).

1.3.5 Distributed Operating System

A distributed operating system, in contrast, is one that appears to its users as a traditional uniprocessor system, even though it is actually composed of multiple processors. In a true distributed system, users should not be aware of where their programs are being run or where their files are located; that should all be handled automatically and efficiently by the operating system.

1.3.6 Network Operating Systems

Network operating systems are not fundamentally different from single processor operating systems. They obviously need a network interface controller and some low-level software to drive it, as well as programs to achieve remote login and remote files access, but these additions do not change the essential structure of the operating systems.

True distributed operating systems require more than just adding a little code to a uniprocessor operating system, because distributed and centralised systems differ in critical ways. Distributed systems, for example, often allow programs to run on several processors at the same time, thus requiring more complex processor scheduling algorithms in order to optimize the amount of parallelism achieved.

1.4 Operating System Components

An operating system provides the environment within which programs are executed. To construct such an environment, the system is partitioned into small modules with a well-defined interface. The design of a new operating system is a major task. It is very important that the goals of the system be will defined before the design begins. The type of system desired is the foundation for choices between various algorithms and strategies that well be necessary.

A system as large and complex as an operating system can only be created by partitioning it into smaller pieces. Each of these pieces should be a well defined portion of the system with carefully defined inputs, outputs, and functions. Obviously, not all systems have the same structure. However, many modern operating systems share the system components outlined below.

1.4.1 Process Management

A process is the unit of work in a system. Such a system consists of a collection of processes, some of which are operating system processes, those that execute system code, and the rest being user processes, those that execute user code. All of those processes can potentially execute concurrently.

The operating system is responsible for the following activities in connection with processes managed:

- The creation and deletion of both user and system processes
- The suspension and resumption of processes.
- The provision of mechanisms for process synchronisation
- The provision of mechanisms for deadlock handling.

1.4.2 Memory Management

Memory is central to the operation of a modern computer system. Memory is a large array of words or bytes, each with its own address. Interaction is achieved through a sequence of reads or writes of specific memory address that the CPU fetches from and stores in memory.

In order for a program to be executed, it must be mapped to absolute addresses and loaded in to memory. As the program executes, it accesses program instructions and data from memory by generating these absolutes is declared available, and the next program may be loaded and executed.

The operating system is responsible for the following activities in connection with memory management:

- Keeping track of which parts of memory are currently being used and by whom.
- Deciding which processes are to be loaded into memory when memory space becomes available.
- Allocating and deallocating memory space as needed.

1.4.3 Secondary Storage Management

The main purpose of a computer system is to execute programs. These programs, together with the data they access, must be in main memory during execution. Since the main memory is too small to permanently accommodate all data and programs, the computer system must provide secondary storage to back-up main memory. Most modem computer systems use disks as the primary on-line storage of information, of both

programs and data. Most programs, like compilers, assemblers, sort routines, editors, formatters, and so on, are stored on the disk until loaded into memory, and then use the disk as both the source and destination of their processing. Hence the proper management of disk storage is of central importance to a computer system.

The operating system is responsible for the following activities in connection with disk management:

- Free space management
- Storage allocation
- Disk scheduling.

1.4.4 I/O System

One of the purposes of an operating system is to hide the peculiarities of specific hardware devices from the user. For example, in Unix, the peculiarities of I/O devices are hidden from the bulk of the operating system itself by the I/O system. The I/O system consists of:

- A buffer caching system
- A general device driver code
- Drivers for specific hardware devices

Only the device driver knows the peculiarities of a specific device.

1.4.5 File Management

File management is one of the most visible services of an operating system. Computers can store information in several different physical forms. The magnetic tape, disk, and drum are the most common forms. Each of these devices has it own characteristics and physical organisation.

For the convenient use of the computer system, the operating system provides a uniform logical view of information storage. The operating system abstracts from the physical properties of its storage devices to define a logical storage unit, the file. Files are mapped, by the operating system, onto physical devices.

A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data. Data files may be numeric, alphabetic or alphanumeric. Files may be free-form, such as text files, or may be rigidly formatted. In general a file is a sequence of bits, bytes, lines or records whose meaning is defined by its creator and user. It is a very general concept.

The operating system is responsible for the following activities in connection with file management:

- The creation and deletion of files
- The creation and deletion of directory
- The support of primitives for manipulating files and directories
- The mapping of files onto disk storage.
- Backup of files on stable (non volatile) storage.

1.4.6 Protection System

The various processes in an operating system must be protected from each other's activities. For that purpose, various mechanisms are used to ensure that the files, memory segment, CPU and other resources can be operated on only by those processes that have gained proper authorisation from the operating system.

For example, memory addressing hardware ensures that a process can only execute within its own address space. The timer ensures that no process can gain control of the CPU without relinquishing it. Finally, no process is allowed to do its own I/O, to protect the integrity of the various peripheral devices.

Protection refers to a mechanism for controlling the access of programs, processes, or users to the resources defined by a computer controls to be imposed, together with some means of enforcement.

Protection can improve reliability by detecting latent errors at the interfaces between component subsystems. Early detection of interface errors can often prevent contamination of a healthy subsystem by a subsystem that is malfunctioning. An unprotected resource cannot defend against use (or misuse) by an unauthorised or incompetent user.

1.4.7 Networking

A distributed system is a collection of processors that do not share memory or a clock. Instead, each processor has its own local memory, and the processors communicate with each other through various communication lines, such as high speed buses or telephone lines. Distributed systems vary in size and function. They may involve microprocessors, workstations, minicomputers, and large general purpose computer systems.

The processors in the system are connected through a communication network, which can be configured in a number of different ways. The network may be fully or partially connected. The communication

network design must consider routing and connection strategies, and the problems of connection and security.

A distributed system provides the user with access to the various resources the system maintains. Access to a shared resource allows computation speed-up, data availability, and reliability.

1.4.8 Command Interpreter System

One of the most important components of an operating system is its command interpreter. The command interpreter is the primary interface between the user and the rest of the system.

Many commands are given to the operating system by control statements. When a new job is started in a batch system or when a user logs in to a time-shared system, a program which reads and interprets control statements is automatically executed. This program is variously called (1) the control card interpreter, (2) the command line interpreter,
(3) the shell (in UNIX), and so on. Its function is quite simple: get the next command statement, and execute it.

The command statement themselves deal with process management, I/O handling, secondary storage management, main memory management, file system access, protection, and networking.

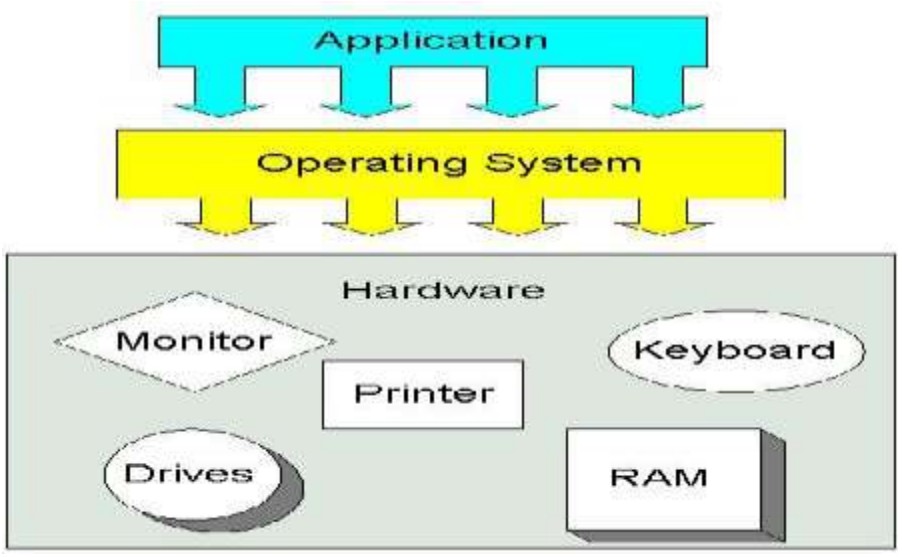


Fig.10: Relationship between the operating system and other components of the computer system

1.1 Language Translators

A programming language is a set of notations in which we express our instructions to the computer. At the initial stage of computer development, programs were written in machine language conducting the binary system i.e. 0 and 1. Such programs were hard to write, read, debug and maintain. In an attempt to solve these problems, other computer languages were developed. However, computers can run programs written only in machine language. There is therefore the need to translate programs written in these other languages to machine language. The suites of languages that translate other languages to machine language are called language translators. The initial program written in a language different from machine language is called the source program and its equivalent in machine language is called object program.

Three examples of classes of language translators are assemblers, interpreters and compilers.

1. **Assemblers:** An assembler is a computer program that accepts a source program in assembly language program and reads and translates the entire program into an equivalent program in machine language called the object program or object code. Each machine has its own assembly language, meaning that the assembly language of one machine cannot run on another machine.

2. **Interpreters:** An interpreter is a program that accepts program from a source language, reads, translates and executes it, line by line, into machine language.

3. **Compilers:** A compiler is a computer program that accepts a source program in one high-level language, reads and translates the entire user's program into an equivalent program in machine language, called the object program or object code.

The stages in compilation include:

- Lexical analysis
- Syntax analysis
- Semantic analysis
- Code generation

For each high-level language, there are different compilers. We can therefore talk of COBOL Compilers, FORTRAN Compilers, BASIC Compilers, etc. A compiler also detects syntax errors, that is, errors that arise from the use of the language. Compilers are portable, i.e. a COBOL Compiler on one machine can run on a different machine with minimum changes.

1.2 Utility Software

This is a set of commonly used programs in data processing departments, also called service or general-purpose programs. They perform the following operations.

- File Conversion: This covers data transfer from any medium to another, making an exact copy or simultaneously editing and validating. For example, copying from a hard disk to a diskette.

- File Copy: It makes an exact copy of a file from one medium to another or from an area of a medium to another area of the same medium.

- Housekeeping Operations: These include programs to clear areas of storage, writing file labels and updating common data. They are not involved in solving the problem in hand. They are operations that must be performed before and after actual processing.

1.3 Application Software

Application software is a set of programs designed to solve problems of a specific nature. It could either be supplied by the computer manufacturer, or in some cases, the users produce their own application program called user programs. Hence, application software could be subdivided into two classes, namely; generalized software and user- defined software.

Under generalised software, we have as examples: Word Processing Programs e.g. Word Perfect, Word Star, Microsoft Word. Also included are Desktop Publishing e.g. Ventura, PageMaker, CorelDraw, likewise the Spreadsheet program e.g. LOTUS 1,2,3, Excel, Super-Q. Under the user-defined, software, we could have some user-defined packages for a particular company or organisation, for accounting, payroll or some other specialised purposes.

- **The Word Processor:** A word processor is used to create, edit, save and print reports. It affords the opportunity to make amendments before printing is done. During editing a character, word, sentence or a number of lines can be removed or inserted as the case may be. Another facility possible is spell checking. A document can be printed as many times as possible. Word processors are mainly used to produce letters, mailing lists, labels, greeting cards, business cards, reports, manuals and newsletters. Examples are: WordPerfect, WordStar, Display Writer, Professional Writer, LOTUS Manuscript, Ms-Word, LOCO Script and MM Advantage II etc.

- **The Spreadsheet:** This is an application mainly designed for numerical figures and reports. Spreadsheets contain columns and rows, in which numbers can be entered. It is possible to change numbers before printing is done. Other features of spreadsheets are the ability to use formulas to calculate, use sum and average function, ability to perform automatic recalculation, and the capacity to display reports in graphical modes. The spreadsheet is used for budgets, tables, cost analysis, financial reports, tax and statistical analysis. Examples are: LOTUS 123, Supercalc, MS Multiplan, MS-Excel, VP Planner etc.

- **Integrated Packages:** They are programs or packages that perform a variety of different processing operations using data that is compatible with whatever operation is being carried out. They perform a number of operations like word processing, database management and spread sheeting. Examples are: Office Writer, Logistic Symphony, Framework, Enable, Ability, Smart Ware II, and Microsoft Works V2.

- **Graphic Packages:** These are packages that enable you to bring out images, diagrams and pictures. Examples are PM, PM Plus, Graphic Writer, Photoshop.

- **Database Packages:** These are software for designing, setting up and subsequently managing a database. (A database is an organised collection of data that allows for modification, taking care of different users view). Examples are Dbase II, III, IV, FoxBASE, Base Data Perfect, Paradox III, Revelation Advanced and MS-Access.

- **Statistical Packages:** These are packages that can be used to solve statistical problems, e.g. Stat Graphical, and SPSS (Statistical Packages for Social Scientists).

- **Desktop Publishing:** These are packages that can be used to produce books and documents in standard form. Examples are PageMaker, Ventura, Publishers, Paints Brush, Xerox Form Base, News Master II, and Dbase Publisher.

- **Game Packages:** These are packages that contain a lot of games for children and adults. Examples are Chess, Scrabble, Monopoly, Tune Trivia, Star Trek 2, California Game, Soccer Game, War Game, Spy Catcher and Dracula in London.

- **Communication Packages:** Examples are Carbon Plus, Data Talk V3.3, Cross Talk, SAGE Chit Chat and Data Soft.

  There are so many packages around, virtually for every field of study but these are just to mention a few of them. Advantages of these packages include that they are quicker to and cheaper implement, time saving, minimum time for its design, they have been tested and proven to be correct, they are usually accompanied by full documentation, and are also very portable.

User Programs

This is a suite of programs written by programmers for computer users. They are required for the operation of their individual business or tasks. An example is a payroll package developed for the salary operation of a particular company.

PROGRAMMING THE COMPUTER

3.0 An Overview of Computer Programming Languages

Basically, human beings cannot speak or write in computer language, and since computers cannot speak or write in human language, an intermediate language had to be devised to allow people to communicate with the computers. These intermediate languages, known as programming languages, allow a computer programmer to direct the activities of the computer. These languages are structured around a unique set of rules that dictate exactly how a programmer should direct the computer to perform a specific task. With the powers of reasoning and logic of human beings, there is the capability to accept an instruction and understand it in many different forms. Since a computer must be programmed to respond to specific instructions, instructions cannot be given in just any form. Programming languages standardise the instruction process. The rules of a particular language tell the programmer how the individual instructions must be structured and what sequence of words and symbols must be used to form an instruction.

- An operation code
- Some operands.

The operation code tells the computer what to do such as add, subtract, multiply and divide. The operands tell the computer the data items involved in the operations. The operands in an instruction may consist of the actual data that the computer may use to perform an operation, or the storage address of data. Consider for example the instruction: a = b + 5. The '=' and '+' are operation codes while 'a', 'b' and '5' are operands. The 'a' and 'b' are storage addresses of actual data while '5' is an actual data.

Some computers use many types of operation codes in their instruction format and may provide several methods for doing the same thing. Other computers use fewer operation codes, but have the capacity to perform more than one operation with a single instruction. There are four basic types of instructions, namely:

(a) input-output instructions;
(b) arithmetic instructions;
(c) branching instructions; and
(d) logic instructions.

An input instruction directs the computer to accept data from a specific input device and store it in a specific location in the store. An output instruction tells the computer to move a piece of data from a computer storage location and record it on the output medium.

All of the basic arithmetic operations can be performed by the computer. Since arithmetic operations involve at least two numbers, an arithmetic operation must include at least two operands.

Branch instructions cause the computer to alter the sequence of execution of instruction within the program. There are two basic types of branch instructions; namely unconditional branch instruction and conditional branch instruction. An unconditional branch instruction or statement will cause the computer to branch to a statement regardless of the existing conditions. A conditional branch statement will cause the computer to branch to a statement only when certain conditions exist.

Logic instructions allow the computer to change the sequence of execution of instruction, depending on conditions built into the program by the programmer. Typical logic operations include: shift, compare and test.

3.1 Types of Programming Language

The effective utilisation and control of a computer system is primarily through the software of the system. We note that there are different types of software that can be used to direct the computer system. System software directs the internal operations of the computer, and applications software allows the programmer to use the computer to solve user made problems. The development of programming techniques has become as important to the advancement of computer science as the developments in hardware technology. More sophisticated programming techniques and a wider variety of programming languages have enabled computers to be used in an increasing number of applications.

Programming languages, the primary means of human-computer communication, have evolved from early stages where programmers entered instructions into the computer in a language similar to that used in the application. Computer programming languages can be classified into the following categories:

(a) Machine language
(b) Assembly language
(c) High level symbolic language
(d) Very high level symbolic language

3.1.1 Machine Language

The earliest forms of computer programming were carried out by using languages that were structured according to the computer stored data, that is, in a binary number system. Programmers had to construct programs that used instructions written in binary notation 1 and 0. Writing programs in this fashion is tedious, time-consuming and susceptible to errors.

Each instruction in a machine language program consists, as mentioned before, of two parts namely: operation code and operands. An added difficulty in machine language programming is that the operands of an instruction must tell the computer the storage address of the data to be processed. The programmer must designate storage locations for both instructions and data as part of the programming process. Furthermore, the programmer has to know the location of every switch and register that will be used in executing the program, and must control their functions by means of instructions in the program.

A machine language program allows the programmer to take advantage of all the features and capabilities of the computer system for which it was designed. It is also capable of producing the most efficient program as far as storage requirements and operating speeds are concerned. Few programmers today write applications programs in machine language. A machine language is computer dependent. Thus, an IBM machine language will not run on NCR machine, DEC machine or ICL machine. A machine language is the First Generation (computer) Language (IGL).

3.1.2 Assembly (Low Level) Language

Since machine language programming proved to be a difficult and tedious task, a symbolic way of expressing machine language instructions is devised. In assembly language, the operation code is expressed as a combination of letters rather than binary numbers, sometimes called mnemonics. This allows the programmer to remember the operations codes easily than when expressed strictly as binary numbers. The storage address or location of the operands is expressed as a symbol rather than the actual numeric address. After the computer has read the program, operations software are used to establish the actual locations for each piece of data used by the program. The most popular assembly language is the IBM Assembly Language.

Because the computer understands and executes only machine language programs, the assembly language program must be translated into a machine language. This is accomplished by using a system software program called an assembler. The assembler accepts an assembly

language program and produces a machine language program that the computer can actually execute. The schematic diagram of the translation process of the assembly language into the machine language is shown in fig.9.1. Although, assembly language programming offers an improvement over machine language programming, it is still an arduous task, requiring the programmer to write programs based on particular computer operation codes. An assembly language program developed and run on IBM computers would fail to run on ICL computers. Consequently, the portability of computer programs in a computer installation to another computer installation which houses different makes or types of computers were not possible. The low level languages are, generally, described as Second Generation (Computer) Language (2GL).

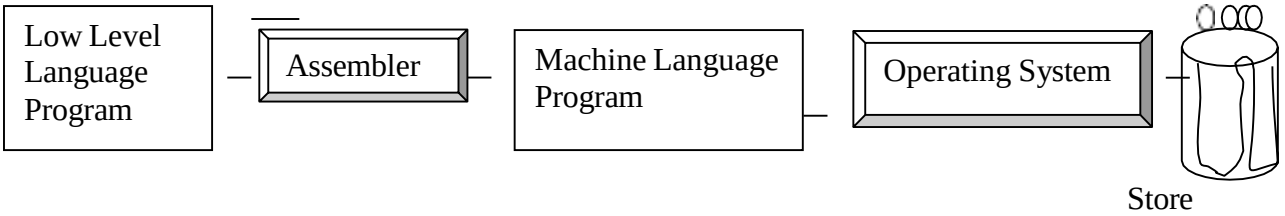


Fig.11: The assembly language program translation process

3.1.3 High Level Language

The difficulty of programming and the time required to program computers in assembly languages and machine languages led to the development of high-level languages. The symbolic languages, sometimes referred to as problem oriented languages reflect the type of problem being solved rather than the computer being used to solve it. Machine and assembly language programming is machine dependent but high level languages are machine independent, that is, a high-level language program can be run on a variety of computers.

While the flexibility of high level languages is greater than that of machine and assembly languages, there are close restrictions in exactly how instructions are to be formulated and written. Only a specific set of numbers, letters, and special characters may be used to write a high level program and special rules must be observed for punctuation. High level language instructions do resemble English language statements and the mathematical symbols used in ordinary mathematics. Among the existing and popular high level programming languages are Fortran, Basic, Cobol, Pascal, Algol, Ada and P1/1. The schematic diagram of the translation process of a high level language into the machine language is shown in fig.9.2. The high level languages are, generally, described as Third Generation (Computer) Language (3GL).

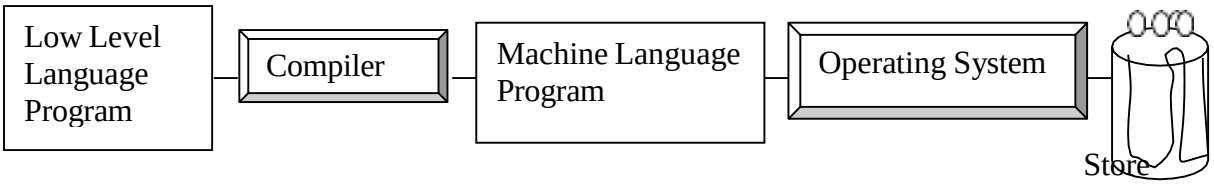


Fig. 12: The high level language program translation process

The general procedure for the compilation of a computer program coded in any high level language is conceptualised in Fig. 13.

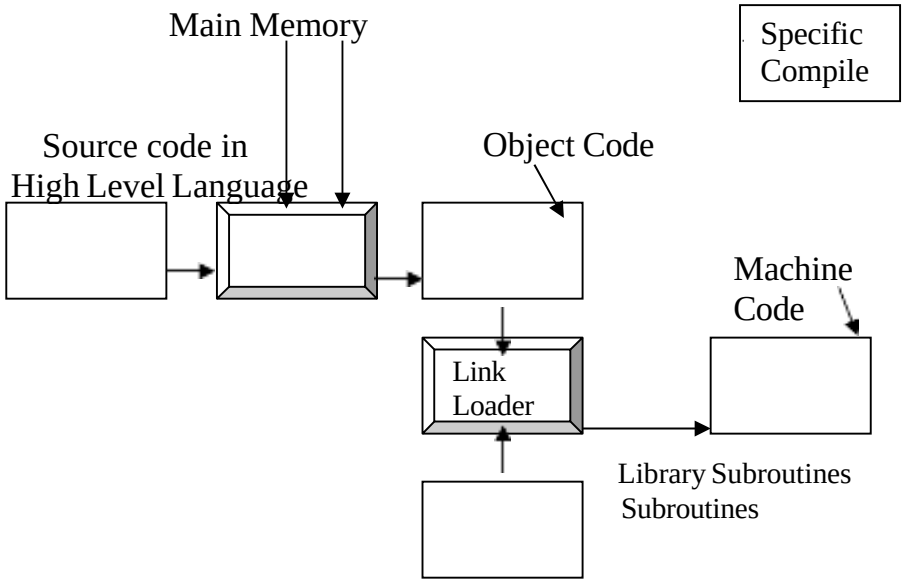


Fig. 13: General procedure for compiling a high level language program

3.1.4 Very High Level Language

Programming aids or programming tools are provided to help programmers do their programming work more easily. Examples of programming tools are:

- Program development systems that help users to learn programming, and to program in a powerful high level language. Using a computer screen (monitor) and keyboard under the direction of an interactive computer program, users are helped to construct application programs.
- A program generator or application generator that assists computer users to write their own programs by expanding simple statements into program code'.
- A database management system
- Debuggers that are programs that help the computer user to locate errors (bugs) in the application programs they write.

The very high level language generally described as the Fourth Generation (computer) Language (4GL), is an ill-defined term that refers to software intended to help computer users or computer programmers to develop their own application programs more quickly and cheaply. A 4GL, by using a menu system for example, allows users to specify what they require, rather than describe the procedures by which these requirements are met. The detailed procedure by which the requirements are met is done by the 4GL software which is transparent to the users.

A 4GL offers the user an English-like set of commands and simple control structures in which to specify general data processing or numerical operations. A program is translated into a conventional high- level language such as COBOL, which is passed to a compiler. A 4GL is, therefore, a non-procedural language. The program flows are not designed by the programmer but by the fourth generation software itself. Each user request is for a result rather than a procedure to obtain the result. The conceptual diagram of the translation process of very high level language to machine language is given Fig.14.

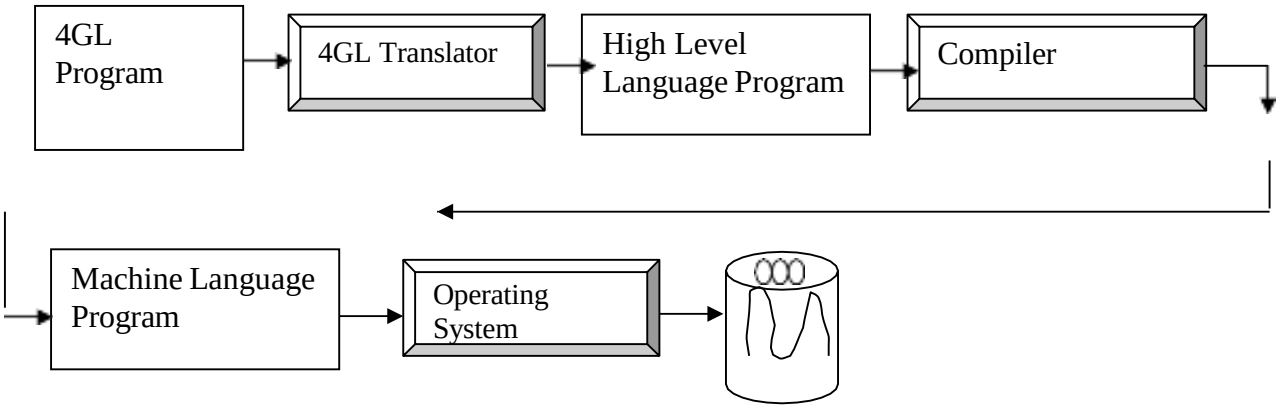


Fig. 14: The program translation process

The 4GL arose partly in response to the applications backlog. A great deal of programming time is spent maintaining and improving old programs rather than building new ones. Many organisations, therefore, have a backlog of applications waiting to be developed. 4GL, by stepping up the process of application design and by making it easier for end-users to build their own programs, helps to reduce the backlog.

BASIC PRINCIPLES OF COMPUTER PROGRAMMING

1.1 Problem Solving With the Computer

The computer is a general-purpose machine with a remarkable ability to process information. It has many capabilities, and its specific function at any particular time is determined by the user. This depends on the program loaded into the computer memory being utilised by the user.

There are many types of computer programs. However, the programs designed to convert the general-purpose computer into a tool for a specific task or applications are called "Application programs". These are developed by users to solve their peculiar data processing problems. Computer programming is the act of writing a program which a computer can execute to produce the desired result. A program is a series of instructions assembled to enable the computer to carry out a specified procedure. A computer program is the sequence of simple instructions into which a given problem is reduced and which is in a form the computer can understand, either directly or after interpretation.

Programming Methodology

Principles of Good Programming

It is generally accepted that a good computer program should have the characteristics shown below:

- **Accuracy:** The program must do what it is supposed to do correctly and must meet the criteria laid down in its specification.
- **Reliability:** The program must always do what it is supposed to do, and never crash.
- **Efficiency:** Optimal utilisation of resources is essential. The program must use the available storage space and other resources in such as way that the system speed is not wasted.
- **Robustness:** The program should cope with invalid data and not stop without an indication of the cause of the source of error.
- **Usability:** The program must be easy enough to use and be well documented.
- **Maintainability:** The program must be easy to amend, having good structuring and documentation.
- **Readability:** The code of a program must be well laid out and explained with comments.

1.2 Stages of Programming

The preparation of a computer program involves a set of procedure. These steps can be classified into eight major stages, viz

(i) Problem definition
(ii) Devising the method of solution
(iii) Developing the method using suitable aids, e.g. pseudo code or flowchart.
(iv) Writing the instructions in a programming language
(v) Transcribing the instructions into "machine sensible" form
(vi) Debugging the program
(vii) Testing the program
(viii) Documenting all the work involved in producing the program.

(i) Problem definition

The first stage requires a good understanding of the problem. The programmer (i.e. the person writing the program) needs to thoroughly understand what is required of a problem. A complete and precise unambiguous statement of the problem to be solved must be stated. This will entail the detailed specification which lays down the input, processes and output required.

(ii) Devising the method of solution

The second stage involved is spelling out the detailed algorithm. The use of a computer to solve problems (be it scientific or business data processing problems) requires that a procedure or an algorithm be developed for the computer to follow in solving the problem.

(iii) Developing the method of solution

There are several methods for representing or developing methods used in solving a problem. Examples of such methods are: algorithms, flowcharts, pseudo code, and decision tables.

(iv) Writing the instructions in a programming language

After outlining the method of solving the problem, a proper understanding of the syntax of the programming language to be used is necessary in order to write the series of instructions required to get the problem solved.

(v) Transcribing the instructions into machine sensible form

After the program is coded, it is converted into machine sensible form or machine language. There are some manufacturers written programs that translate users programs (source programs) into machine language (object

code). These are called translators and instructions that machines can execute at a go, while interpreters accept a program and execute it line-by-line.

During translation, the translator carries out syntax check on the source program to detect errors that may arise from wrong use of the programming language.

(vi) Program debugging

A program seldomly executes successfully the first time. It normally contains a few errors (bugs). Debugging is the process of locating and correcting errors. There are three classes of errors.

- **Syntax errors:** Caused by mistake coding (illegal use of a feature of the programming language)

- **Logic errors:** Caused by faulty logic in the design of the program. The program will work but not as intended.

- **Execution errors:** The program works as intended but illegal input or other circumstances at run-time makes the program stop. There are two basic levels of debugging. The first level called desk checking or dry running is performed after the program has been coded and entered or key punched. Its purpose is to locate and remove as many logical and clerical errors as possible.

The program is then read (or loaded) into the computer and processed by a language translator. The function of the translator is to convert the program statements into the binary code of the computer called the object code. As part of the translation process, the program statements are examined to verify that they have been coded correctly, if errors are detected, a series of diagnostics referred to as an error message list is generated by the language translator. With this list in the hand of the programmer, the second level of debugging is reached.

The error message list helps the programmer to find the cause of errors and make the necessary corrections. At this point, the program may contain entering errors, as well as clerical errors or logic errors. The programming language manual will be very useful at this stage of program development.

After corrections have been made, the program is again read into the computer and again processed by the language translator. This is repeated over and over again until the program is error-free.

(vii) Program testing

The purpose of testing is to determine whether a program consistently produces correct or expected results. A program is normally tested by executing it with a given set of input data (called test data), for which correct results are known.

For effective testing of a program, the testing procedure is broken into three segments.

a. The program is tested with inputs that one would normally expect for an execution of the program.

b. Valid but slightly abnormal data is injected (used) to determine the capabilities of the program to cope with exceptions. For example, minimum and maximum values allowable for a sales- amount field may be provided as input to verify that the program processed them correctly.

c. Invalid data is inserted to test the program's error-handling routines. If the result of the testing is not adequate, then minor logic errors still abound in the program. The programmer can use any of these three alternatives to locate the bugs.

Other methods of testing a program for correctness include:

- **Manual walk-through:** The programmer traces the processing steps manually to find the errors, pretending to be the computer, following the execution of each statement in the program, noting whether or not the expected results are produced.

- **Use of tracing routines:** If this is available for the language, this is similar to (1) above but is carried out by the computer; hence it takes less time and is not susceptible to human error.

- **Storage dump:** This is the printout of the contents of the computer's storage locations. By examining the contents of the various locations, when the program is halted, the instruction at which the program is halted can be determined. This is an important clue to finding the error that caused the halt.

- **Program documentation:** Documentation of the program should be developed at every stage of the programming cycle. The following are documentations that should be done for each program.

(a) Problem Definition Step

- A clear statement of the problem
- The objectives of the program (what the program is to accomplish)
- Source of request for the program.
- Person/official authorising the request

(b) Planning the Solution Step

- Flowchart, pseudo code or decision tables
- Program narrative
- Descriptive of input, and file formats

(c) Program source coding sheet

(d) User's manual to aid persons who are not familiar with the program to apply it correctly. The manual it contains a description of the program and what it is designed to achieve.

(e) Operator's manual to assist the computer operator to successfully run the program. This manual contains:

(i) Instructions about starting, running and terminating the program.
(ii) Message that may be printed on the console or VDU (terminal) and their meanings.
(iii) Setup and take down instruction for files.

Advantages of Program Documentation

a. It provides all necessary information for anyone who comes in contact with the program.
b. It helps the supervisor in determining the program's purpose, how long the program will be useful and future revision that may be necessary.
c. It simplifies program maintenance (revision or updating).
d. It provides information as to the use of the program to those unfamiliar with it.
e. It provides operating instructions to the computer operator.

FLOWCHARTS AND ALGORITHMS

1.1 Flowcharts

A flowchart is a graphical representation of the major steps of work in process. It displays in separate boxes the essential steps of the program and shows by means of arrows the directions of information flow. The boxes, most often referred to as illustrative symbols, may represent documents, machines or actions taken during the process. The area of concentration is on where or who does what, rather than on how it is done. A flowchart can also be said to be a graphical representation of an algorithm, that is, it is a visual picture which gives the steps of an algorithm and also the flow of control between the various steps.

1.2 Flowchart Symbols

Flowcharts are drawn with the help of symbols. The following are the most commonly used flowchart symbols and their functions:

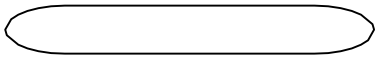
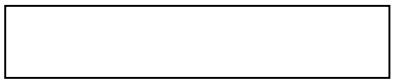
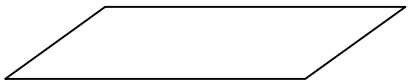
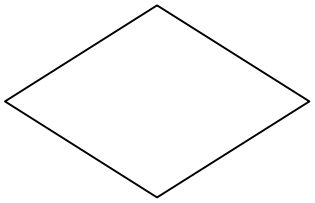
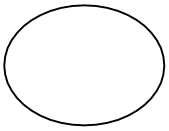
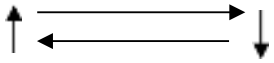
Symbols	Function
 **Terminator**	Used to show the START or STOP May show exit to a closed subroutine.
	Used for arithmetic calculations of process. E.g. Sum = X + Y + Z
	Used for Input and Output instructions, PRINT, READ, INPUT AND WRITE.
	Used for decision making. Has two or more lines leaving the box. These lines are labeled with different decision results, that is,' Yes', 'No', 'TRUE' or 'FALSE' or 'NEGATIVE' or 'ZERO'.
	Used for one or more named operations or program steps specified in a subroutine or another set of flowchart.
	Used for entry to or exit from another part of flowchart. A small circle identifies a junction point of the program
	Used for entry to or exit from a page
	Used to show the direction of travel. They are used in linking symbols. These show operations sequence and data flow directions.

Fig.15: Flowchart symbols and their functions

1.3 Guidelines for Drawing Flowcharts

- Each symbol denotes a type of operation: Input, Output, Processing, Decision, Transfer or Branch or Terminal.
- A note is written inside each symbol to indicate the specific function to be performed.
- Flowcharts are read from top to bottom.

- A sequence of operations is performed until a terminal symbol designates the end of the run or "branch" connector transfers control.

1.4 Flowcharting the Problem

The digital computer does not do any thinking and cannot make unplanned decisions. Every step of the problem has to be taken care of by the program. A problem which can be solved by a digital computer need not be described by an exact mathematical equation, but it does need a certain set of rules that the computer can follow. If a problem needs intuition or guessing, or is so badly defined that it is hard to put into words, the computer cannot solve it. You have to define the problem and set it up for the computer in such a way that every possible alternative is taken care of. A typical flowchart consists of special boxes, in which are written the activities or operations for the solution of the problem. The boxes, linked by means of arrows, show the sequence of operations. The flowchart acts as an aid to the programmer, who follows the flowchart design to write his programs.

1.5 Algorithms

Before a computer can be put to any meaningful use, the user must be able to come out with or define a unit sequence of operations or activities (logically ordered) which gives an unambiguous method of solving a problem or finding out that no solution exists. Such a set of operations is known as an ALGORITHM.

Definition: An algorithm, named after the ninth century scholar Abu Jafar Muhammad Ibn Musu Al-Khowarizmi, is defined as follows:

- An algorithm is a set of rules for carrying out calculations either by hand or a machine.
- An algorithm is a finite step-by-step procedure to achieve a required result.
- An algorithm is a sequence of computational steps that transform the input into the output.
- An algorithm is a sequence of operations performed on data that have to be organised in data structures.
- An algorithm is an abstraction of a program to be executed on a physical machine (model of computation).

The most famous algorithm in history dates well before the time of the ancient Greeks: this is Euclids algorithm for calculating the greatest common divisor of two integers. Before we go into some otherwise

complex algorithms, let us consider one of the simplest but common algorithms that we encounter everyday.

The classic multiplication algorithm

For example to multiply 981 by 1234, this can be done using two methods (algorithms) viz:

a. Multiplication the American way:

Multiply the multiplicand one after another by each digit of the multiplier taken from right to left.

```
      981
      1234
  ------------
      3924
       2943
      1962
      981
      1210554
  ------------
```

b. Multiplication , the British way:

Multiply the multiplicand one after another by each digit of the multiplier taken from left to right.

```
      981
      1234
  ------------
      981
      1962
       2943
      3924
      1210554
  ------------
```

An algorithm therefore can be characterised by the following:

(i) A finite set or sequence of actions
(ii) This sequence of actions has a unique initial action
(iii) Each action in the sequence has unique successor
(iv) The sequence terminates with either a solution or a statement that the problem is unresolved.

An algorithm can therefore be seen as a step-by-step method of solving a problem.

Examples

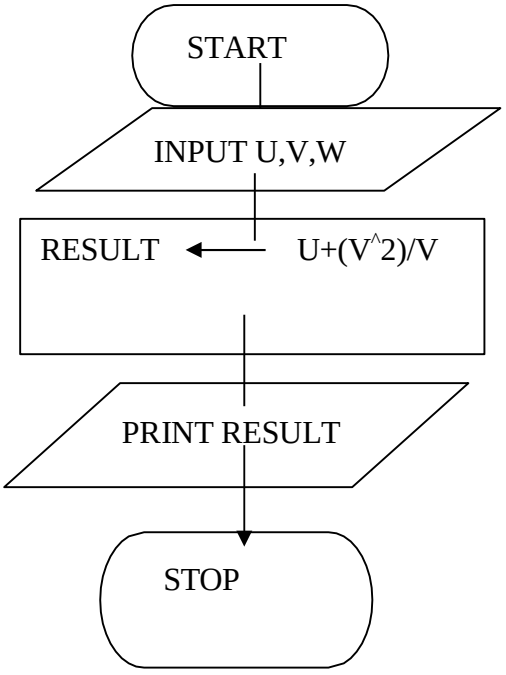
1. Write an algorithm to read values for three variables. U, V, and W and find a value for RESULT from the formula: RESULT = U + V^2/W. Draw the flowchart.

Solution

Algorithm

(i) Input values for U, V, and W
(ii) Computer value for result
(iii) Print value of result
(iv) Stop

Flowchart



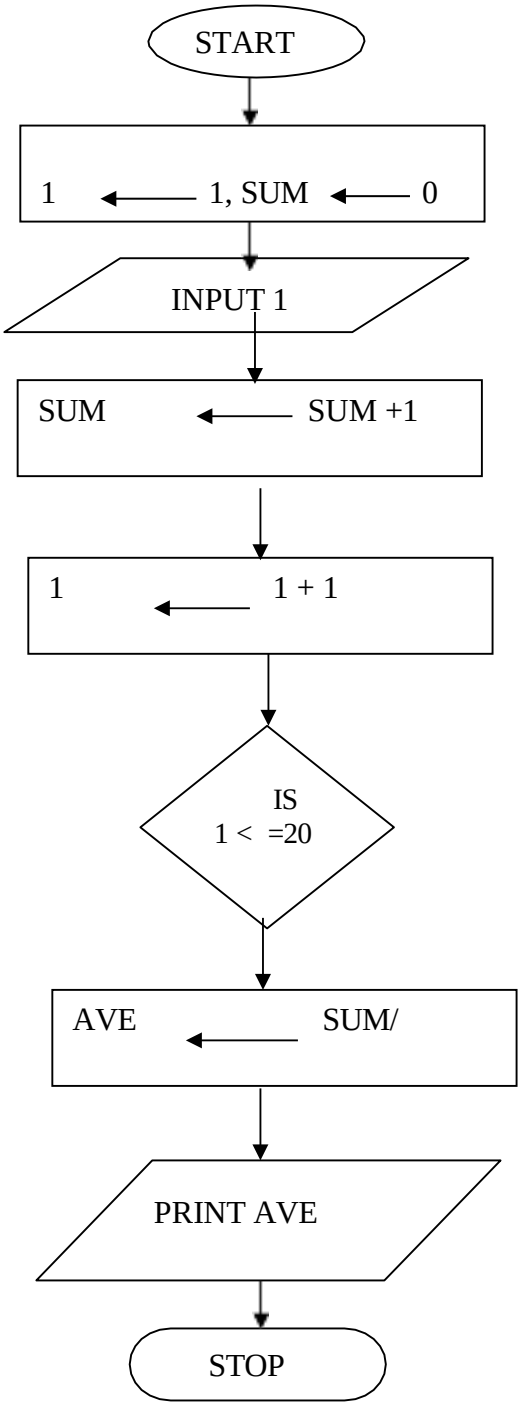
2. Suppose you are given 20 numbers. Prepare the algorithm that adds up these numbers and find the average. Draw the flowchart.

Solution

Algorithm

- Set up a Counter (1) which counts the number of times the loop is executed. Initialise Counter (1) to 1.
- Initialize sum to zero.
- Input value and add to sum.

- Increment the counter (1) by 1.
- Check how many times you have added up the number, if it is not up to the required number of times, to step (iii).
- Compute the average of the numbers.
- Print the average.
- Stop.



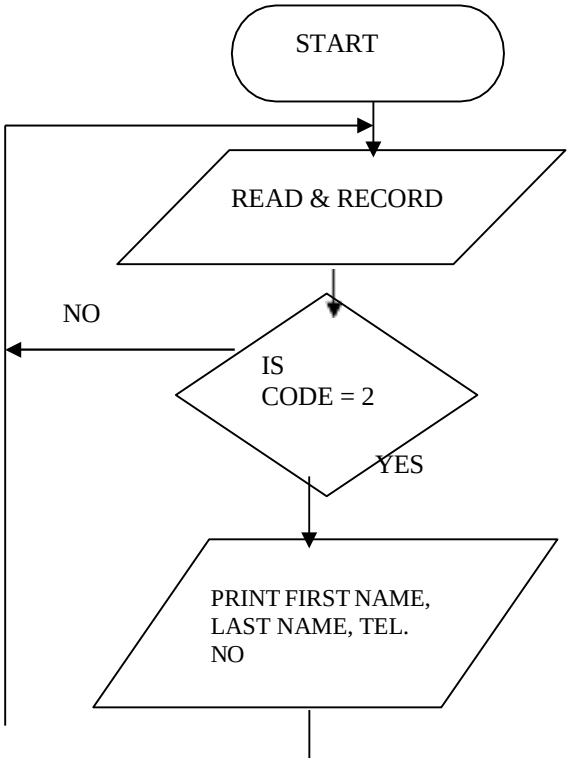
3. Prepare an algorithm that indicates the logic for printing the name and telephone number for each female in a file (Code field is 2 for female). Draw the flowchart.

Solution

Algorithm

(i) Read a record
(ii) Determine if the record pertains to a female (that is, determine if the code field is equal to 2).
(iii) If the code field is not equal to 2, then do not process this record any further, since it contains data for a male. Instead, read the next record; that is, go back to step (i).
(iv) If the record contains data for a female (that is, code is equal to 2), then print out the following fields: first name, last name, telephone number.
(v) Go back to step (i) to read the next record.

Flowchart



Note: Nothing indicates the end of records processing here.

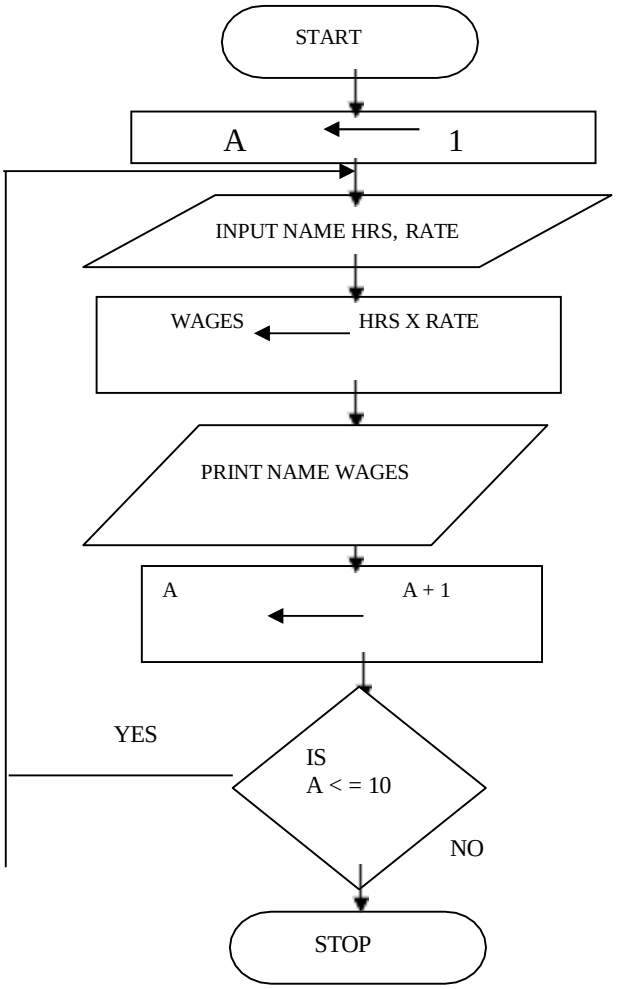
4. Prepare an algorithm that prints name and weekly wages for each employee out of 10 where name, hours worked, and hourly rate are read in. Draw the flowchart.

Solution

Algorithm

(i) Initialise Counter (A) to 1
(ii) Read name, hours and rate and number of workers
(iii) Let the wage be assigned the product of hours and rate
(iv) Print name and wages.
(v) Increment the counter (A) by 1.
(vi) Make a decision (Check how many times you have calculated the wages).
(vii) Stop processing, if you have done it the required number of times.

Flowchart



1.6 Pseudocodes

A pseudocode is a program design aid that serves the function of a flowchart in expressing the detailed logic of a program. Sometimes a program flowchart might be inadequate for expressing the control flow and logic of a program. By using Pseudocodes, program algorithms can be expressed as English-language statements. These statements can be used both as a guide when coding the program in a specific language and as documentation for review by others. Because there is no rigid rule for constructing pseudocodes, the logic of the program can be expressed in a manner that does not conform to any particular programming language. A series of structured words is used to express the major program functions. These structured words are the basis for writing programs using a technical term called "structure programming".

Example

Construct a pseudocode for the problem in the example above.

```
BEGIN
STORE 0 TO SUM
STORE 1 TO COUNT
            DO WHILE COUNT not greater than 10
            ADD COUNT to SUM
      INCREMENT COUNT by 1
      ENDWILE
      END
```

1.7 Decision Tables

Decision tables are used to analyse a problem. The conditions applying in the problem are set out and the actions to be taken, as a result of any combination of the conditions arising are shown. They are prepared in conjunction with or in place of flowcharts. Decision tables are a simple yet powerful and unambiguous way of showing the actions to be taken when a given set of conditions occur. Moreover, they can be used to verify that all conditions have been properly catered for. In this way they can reduce the possibility that rare or unforeseen combinations of conditions will result in confusion about the actions to be taken.

Decision tables have standardised formats and comprise of four sections.

(a) Conditions Stub: This section contains a list of all possible conditions which could apply in a particular problem.

(b) Conditions Entry: This section contains the different combination of the conditions, each combination being given a number termed a 'Rule'.

(c) Action Stub: This section contains a list of the possible actions which could apply for any given combinations of conditions.

(d) Action Entry: This section shows the actions to be taken for each combination of conditions. **Writing the instructions in a programming language (program coding)**

The instructions contained in the algorithm must be communicated to the computer in a language it will understand before it can execute them. The first step is writing these instructions in a programming language (program coding). Program coding is the process of translating the planned solution to the problems, depicted in a flowchart, pseudocode or decision table, into statements of the program. The program flowchart, pseudocode decision table as the case may be, is as a guide by the programmer as he describes the logic in the medium of a programming language. The coding is usually done on coding sheets or coding forms.

3.0 Learning to Run Visual Basic Applications

These sessions will include learning how to work with graphical objects in the visual basic environment and using general visual basic programming concepts.

3.1 How to Design a Project from the Application Wizard

A project is a collection of files that make up your application. A single application might consist of several files, and the project is the collection of those files.

The application wizard can be selected from the New Project dialog box. If you cancel the New Project dialog box, and then later want to start the Application wizard, select File, New Project to display the New Project dialog box once again. The screen you see looks like that in Figure 16.

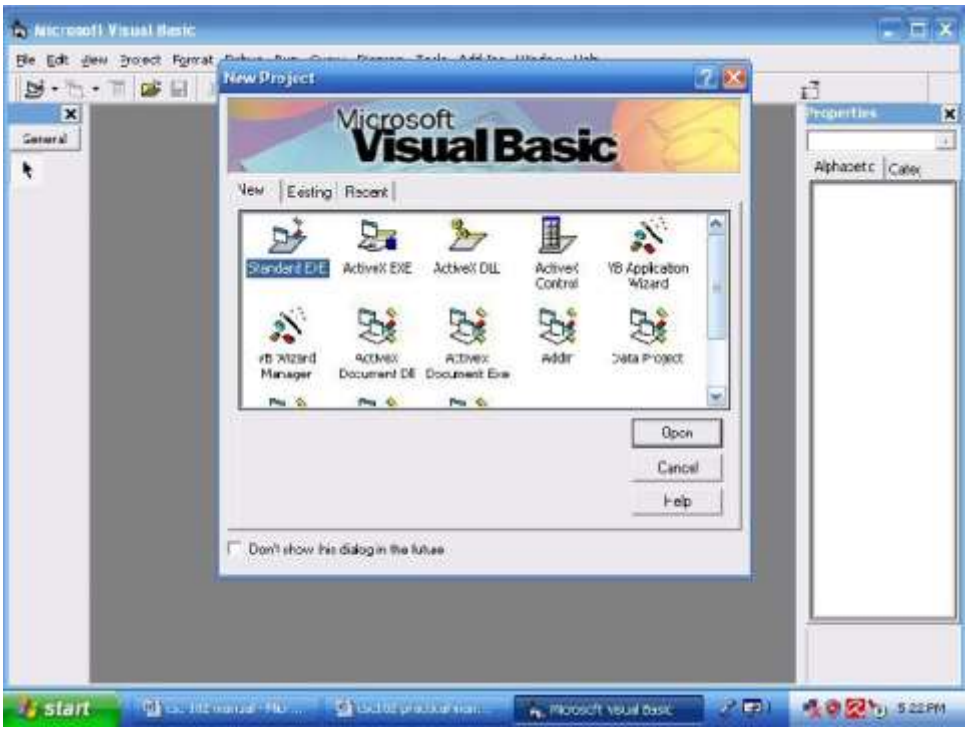


Fig.16

When you select the icon labelled VB Application Wizard on the new tab, the wizard begins its work. The interface type you select will determine how your application will process multiple windows. See Fig.17.

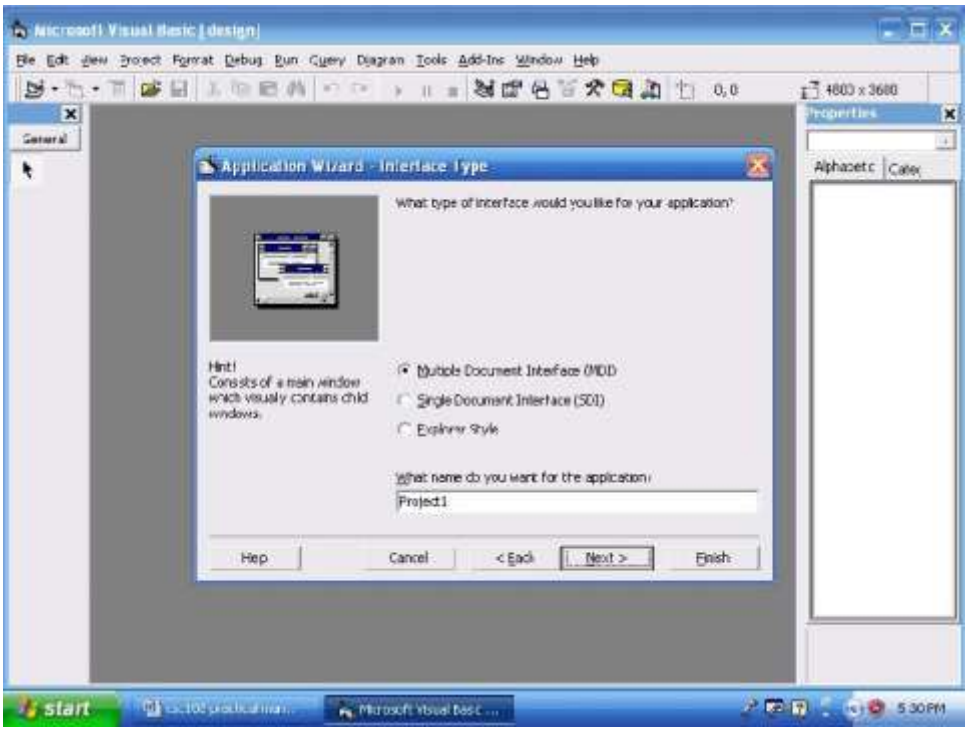


Fig. 17

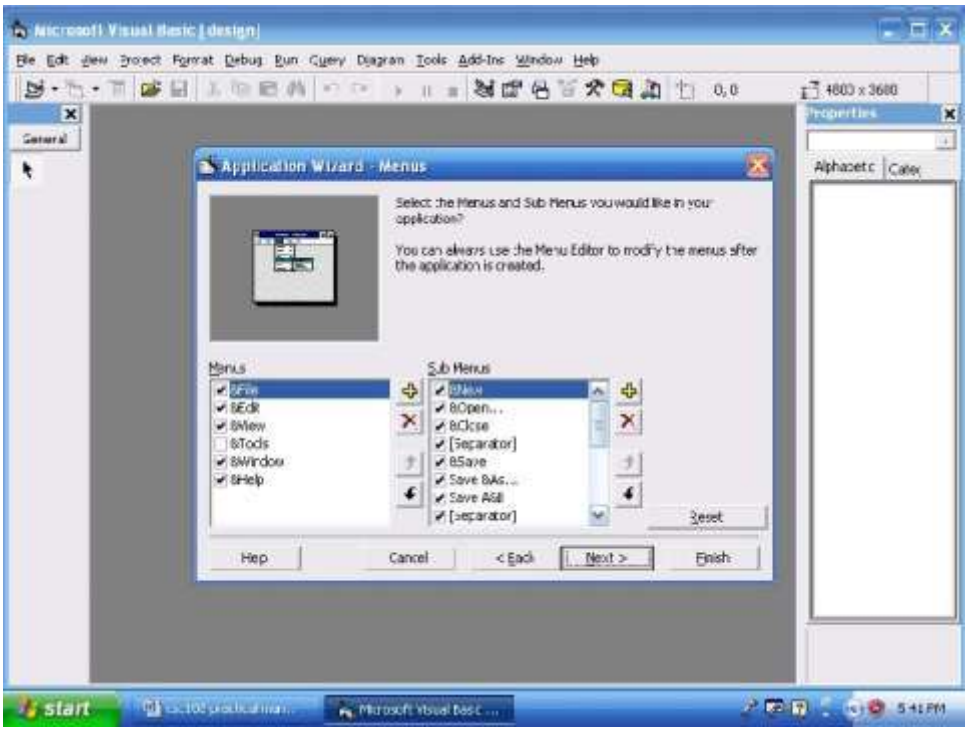


Fig.18

You can select the options you want your application's menu to contain as shown in Fig.18. The options are common Windows options found on most Windows programs. The ampersand (&) next to a letter in a menu name indicates the underscored accelerator key letter; in other words, &New indicates that New appears on the menu and that the user can select the option by pressing Alt+N.

The next wizard screen, shown in Fig.19, lets you select the toolbar buttons that your application will have. Click next to accept all the default toolbar settings.

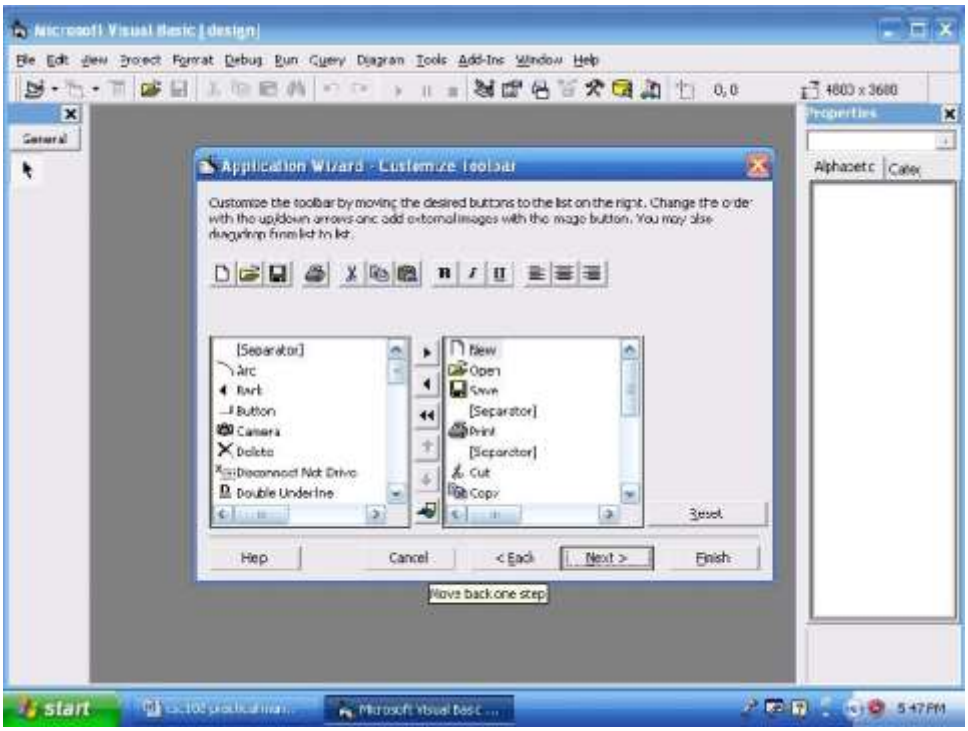


Fig. 19

The next wizard screen to appear is the Resource screen from which you can select to use resources in your program. The next one is the Internet Connectivity screen from which you can add an Internet interface to your program if you want one. The next screen gives the option of adding one of these standard screens to your application:

- **Splash screen** is an opening title screen that appears when your application first begins.
- **Login dialog** is a dialog box that asks for the user's ID and password as part of application security that you can add.
- **Options dialog** is a tabbed blank dialog box from which your users can specify attributes that you set up for the application.
- **About box** is a dialog box that appears when your users select Help, About from the application menu.

You can also select a form template from here. A form template is a model of a form that you can customise.

Click Next to get to the last screen and click the button labelled Finish to instruct Visual Basic to complete your initial application.

3.2 How to Create a Project from the New Project Window

The New Project Window appears when you first start Visual Basic or when you select File, New Project. You will always need **toolbars** in your project. Visual Basic has a total of four toolbars:

- **Debug.** This toolbar appears when you use the interactive debugging tools to trace and correct problems.
- **Edit.** This toolbar aids your editing of Visual Basic codes.
- **Form Editor.** This toolbar helps you adjust objects on forms.
- **Standard.** This toolbar is the default toolbar that appears beneath the menu bar.

You can display and hide these toolbars from the View, Toolbars menu.

3.3 Using the Toolbox

The Toolbox window differs from the toolbar. The toolbox is a collection of tools that act as a repository of **controls** you can place on a form. Fig.20 shows the most common collection of toolbox tools that you'll see.

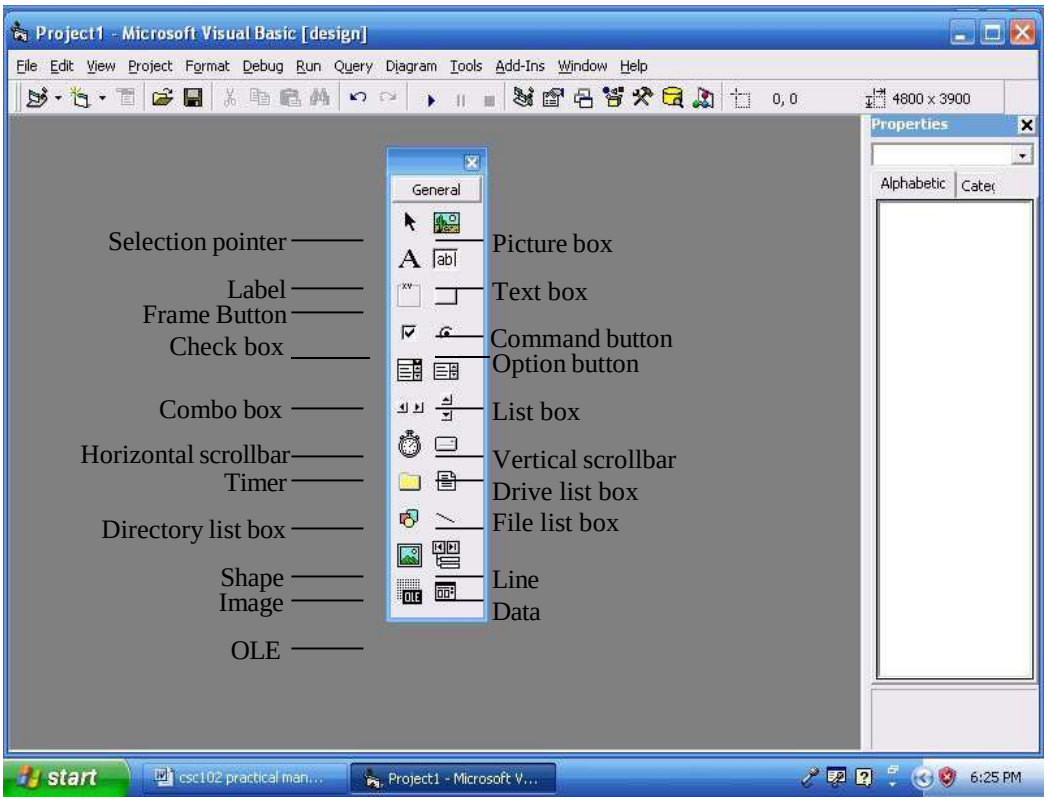


Fig.20

3.4 The Form Window

Most of your work goes on inside the Form window. You'll design all your application's forms, which are the background windows that your users see, in the central editing area where the Form window appears. You can resize the Form window to make the windows you create in your application as large or small as needed.

An application may contain multiple forms: you can display one or more of those forms in their own Form window editing areas. Activate a form by clicking a form by clicking anywhere within the window, or on the title bar.

3.4.1 The Form Layout Window

The Form Layout window is an interesting little window connected closely to the Form window, because the Form Layout window shows you a preview of the Form window's location.

VISUAL BASIC PROJECT WINDOW

1.1 The Project Window

The Project window helps you to manage your application's components. It lists its components in a tree-structured listing. Related objects appear together. You can expand or shrink the details by clicking the plus sign next to the object labelled *Forms*, so that a list of the current project's forms will appear.
The following kinds of objects can appear in the Project window:

- ✓ Projects
- ✓ Forms
- ✓ Modules
- ✓ Class modules
- ✓ User controls
- ✓ User documents
- ✓ Property pages

1.2 The Properties Window

A form can hold many controls. As you add controls to a form, you can select a control by clicking the control. When you select a control, the Properties window changes to list every property related to that control. When you add a control to a Visual Basic application, Visual Basic sets the control's initial property values. When you display the Properties window for a control, you can modify its values. You can do that by selecting the view option and then Properties window.

Fig.21shows a Properties window listing some of the properties for a Label control.

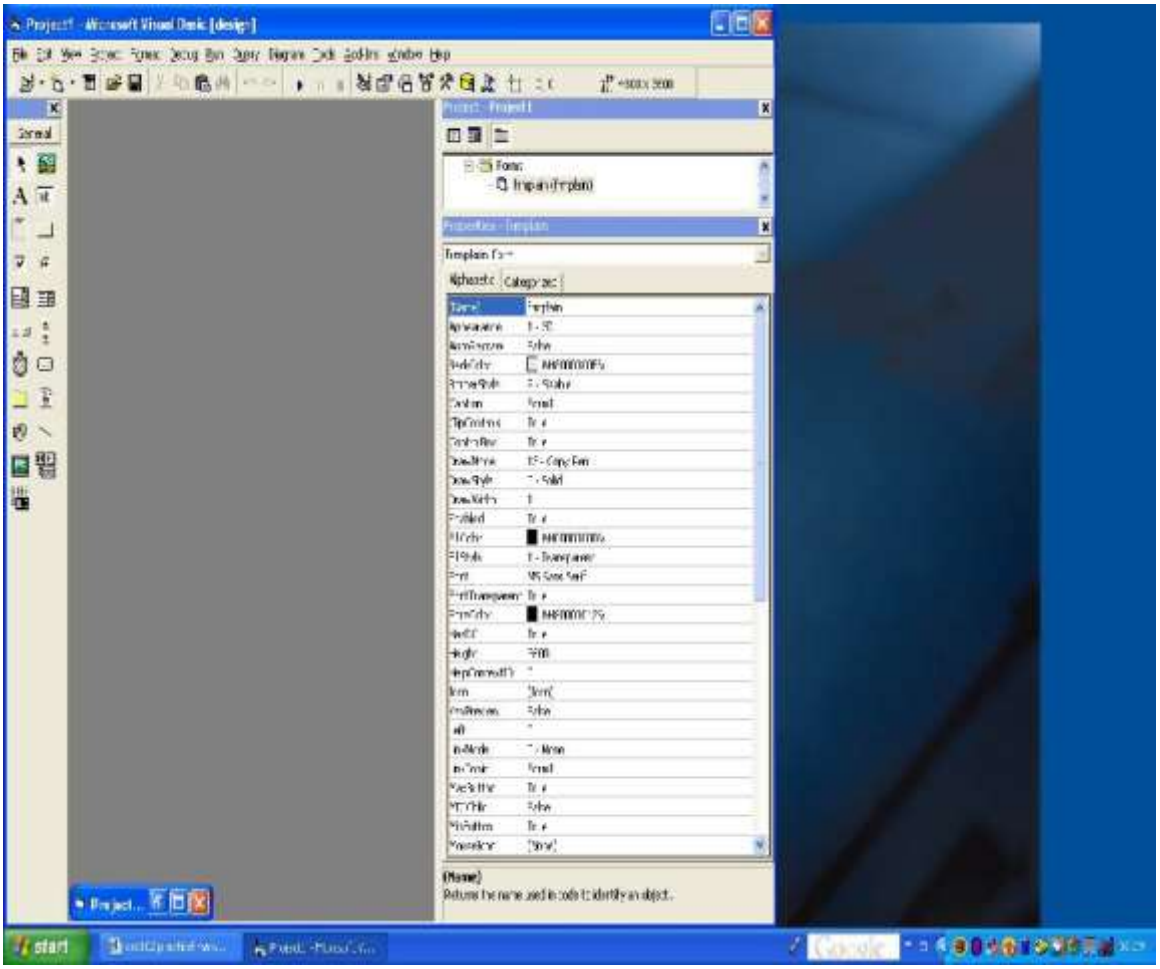


Fig.21

Example 1

Create an application with three controls, a label, a command button and an image control to look like what you have in Fig.22.

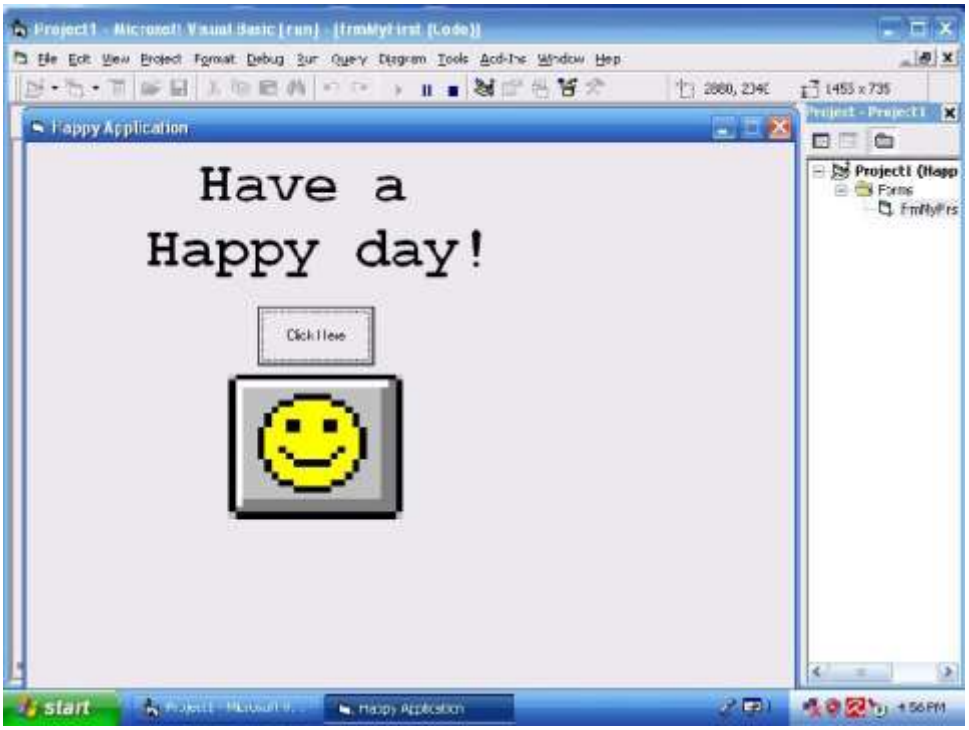


Fig. 22

Guide to the solution

To place a control on a form, click on the control's icon on the toolbox and move the crosshair mouse cursor to the form. As you drag the mouse, Visual Basic draws the control's outline on your form. When you have drawn the control at its proper location and size, release the mouse button to place the control at its proper location.

Assign the following property values to the application's forms and controls:

Control	**Property**	**Property value**
Form	Max Button	False
Label	Alignment	Centre
Label	Name	LblHappy
Label	Caption	Have a happy day!
Label	Font	Courier New
Label	Font style	Bold
Label	Size	36
Label	Left	1320

Label	Height	1695
Label	Top	120
Label	Width	4695
Image	Name	imgHappy
Image	Stretch	True
Command button	Name	cmdHappy
Command button	Caption	Click Here

While writing your application, you can run the application to see what you have done by pressing F5.

You need to add some codes to finalise the application. Double click the form somewhere on the grid inside the Form window to display the code window. Add the codes shown in Fig.23.

To return to the Form window, click the Project window's View Object button.

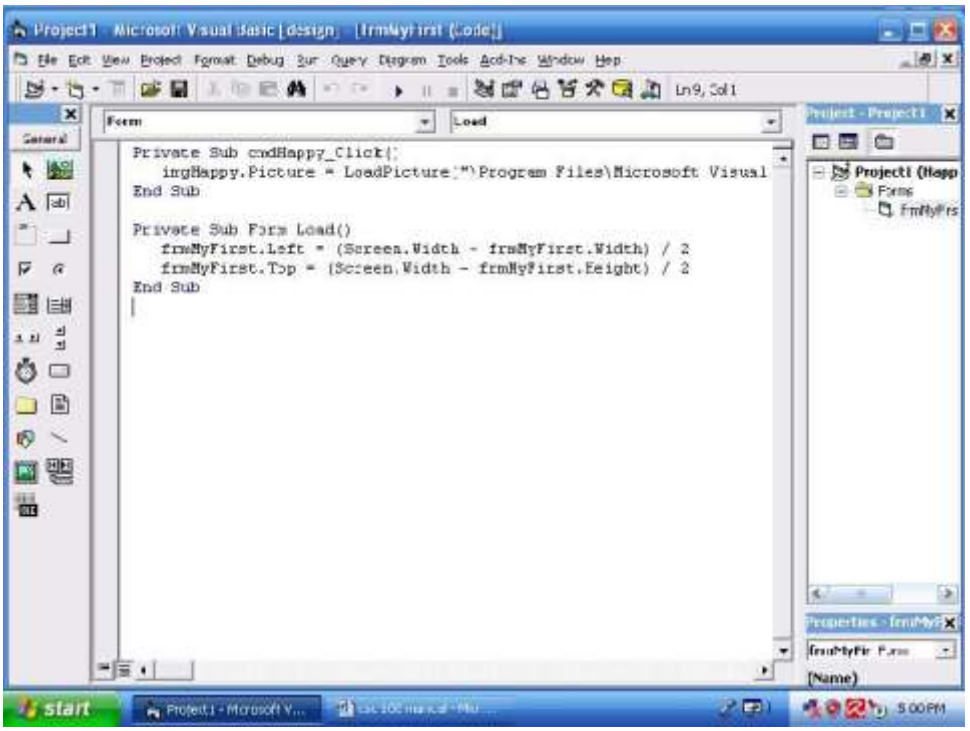


Fig.23

Run your program and click the command button. An image like that shown in Figure 13.2 appears. Save your project and click the Close window to terminate the program. To save, Select File, Save Project. The Save Project option saves every file inside your project as well as a project description file with the filename extension, VBP. Visual Basic asks first for the filename you want to assign to your form. Visual Basic then asks for a project for the project description file. Answer No if Visual Basic asks to add the project to the Source Safe library.

Example 2

Create an application to look like what is shown in Fig.24 to include a label, a textbox (where the secret characters will be entered), an image, and two command buttons.

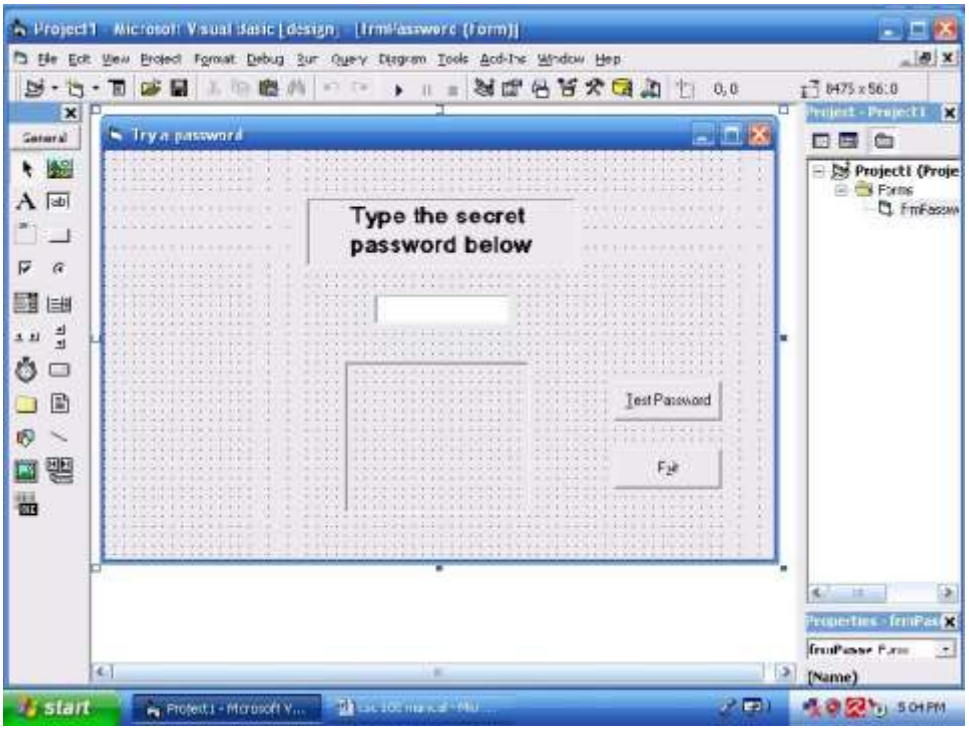


Fig. 24

Guide to the solution

Set these controls and properties on the form:

Control Property Name	Property Value
Form Name	frmPassword
Form Caption	Try a password
Form Height	5610
Form Width	8475
Image Name	imgPassword
Image Border Style	1-Fixed Single
Image Height	1890
Image Left	3000
Image Stretch	True
Image Top	2640
Image Width	2295
Label Name	lblPrompt
Label Border Style	1-Fixed Single
Label Caption	Type the secret password below
Label Font	MS Sans Serif
Label Font Size	14
Label Font Style	Bold

Label Height	855
Label Left	2520
Label Top	600
Label Width	3375
Text box Name	txtPassword
Text box Height	375
Text box Left	3360
Text box PasswordChar	*
Text box Text	(Leave blank by clearing the default value)
Text box Top	1600
Text box Width	1695
Command button Name	cmdTest
Command button Caption	&Test Password
Command button Left	6360
Command button Top	3000
Command button #2Name	cmdExit
Command button #2Caption	E&xit
Command button #2Left	6360
Command button #2Top	3720

Add the following code seen on the screen in Fig.25 to activate the password-based form:

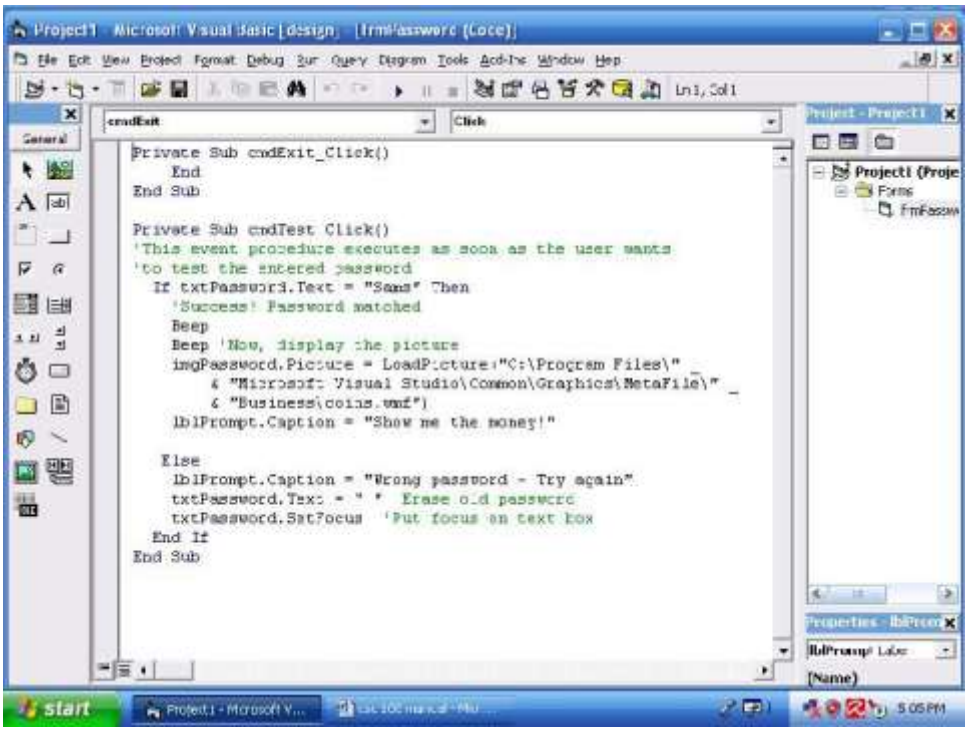


Fig. 25

After running the application, you have what is shown in Fig.26 below:

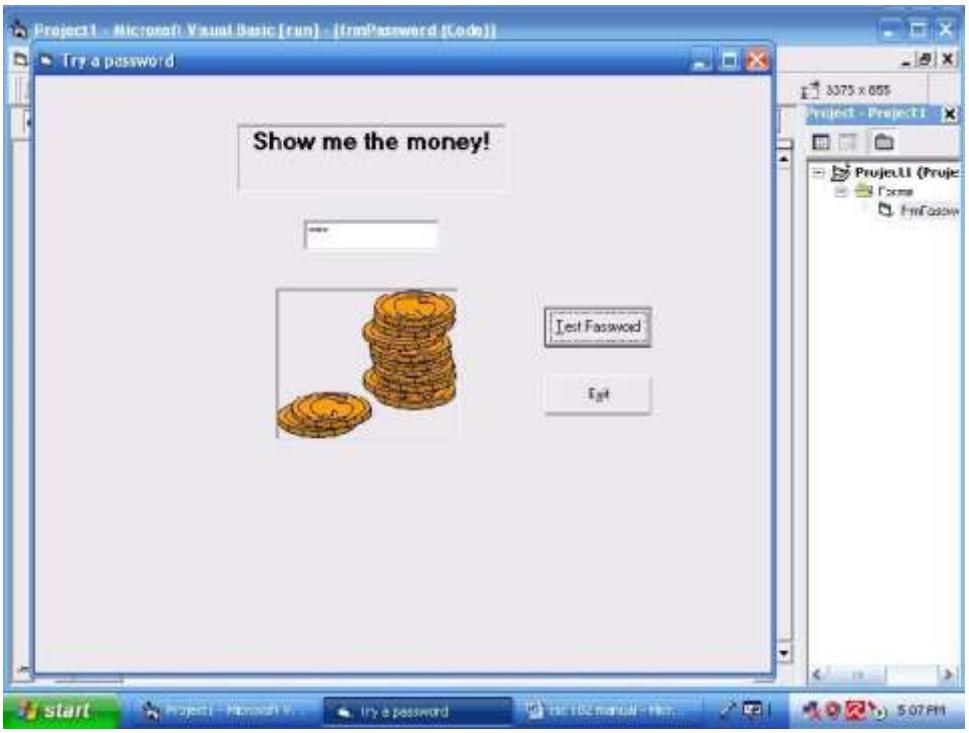


Fig.26

1.1 Creating Menu Applications

You can generate menus for your applications using the application wizard. After you click the menu options and submenus you want in your applications, the Application wizard generates the appropriate menu controls and places them in the generated application.

The Toolbox window does not contain any menu-creation tools. Instead, Microsoft offers a special menu tool called the Menu Editor, shown in Figure 14.1 that you use to create menus. From the Form window, you can press Ctrl+E to display the Menu Editor.

The Menu Editor helps you design menus for your applications. In a way, the Menu Editor acts as a Properties window for the menu bar because you will designate the names of the menu controls as well as the captions that the users see on the menus and other related information from within the Menu Editor.

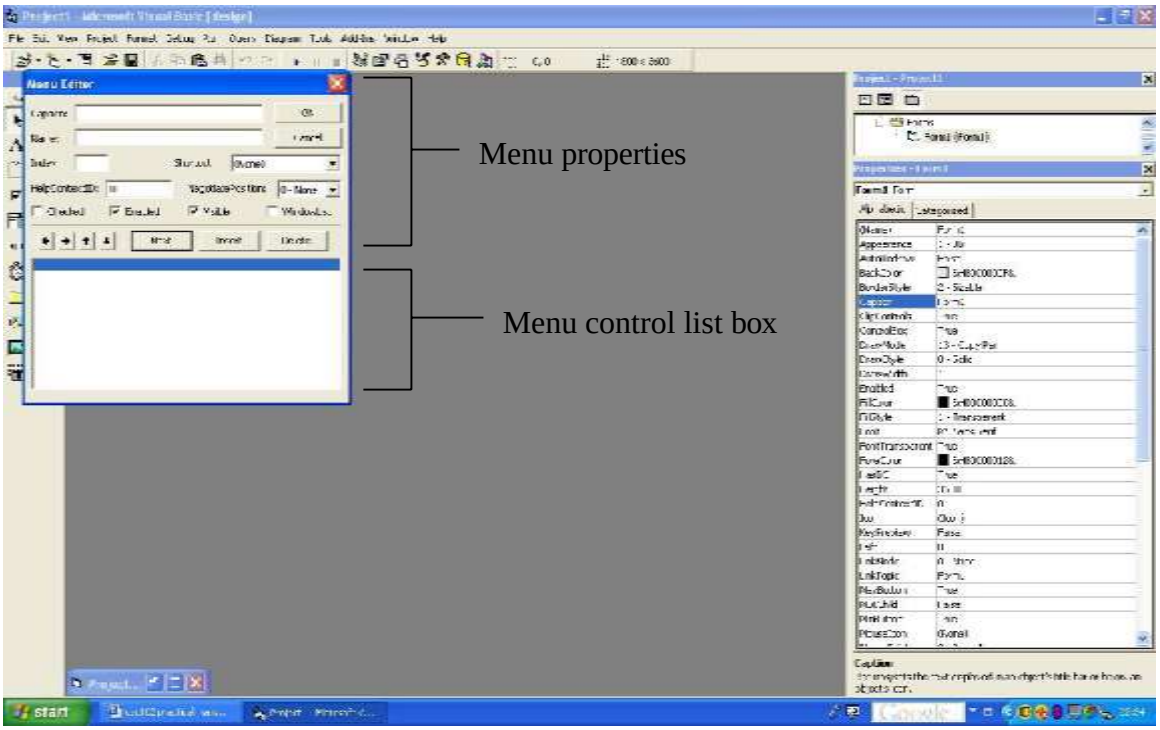


Fig.27

A menu bar offers a special kind of control that lets your select options and issue commands.

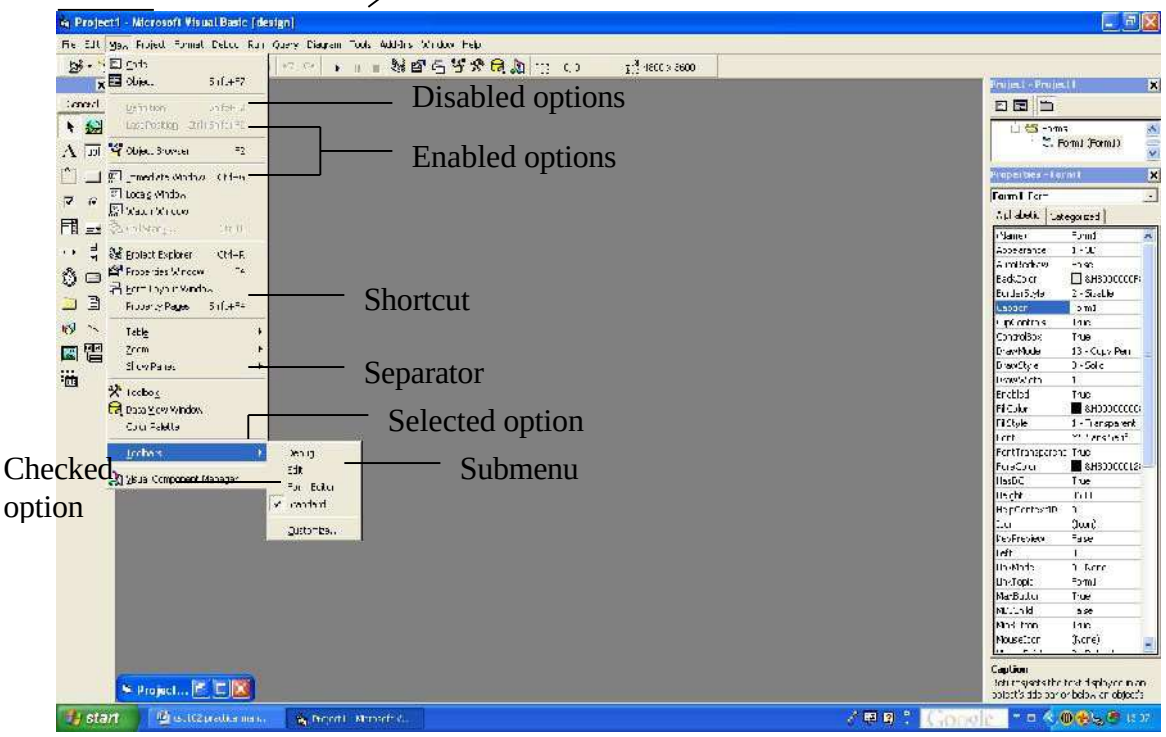


Fig.28

Fig. 28 displays the menu bar and its parts.

Practice creating menu applications with the following exercise.

SELF ASSESSMENT EXERCISE 1

Create an application with three menu options and a label. Your screen should look like what you have in Fig. 29.

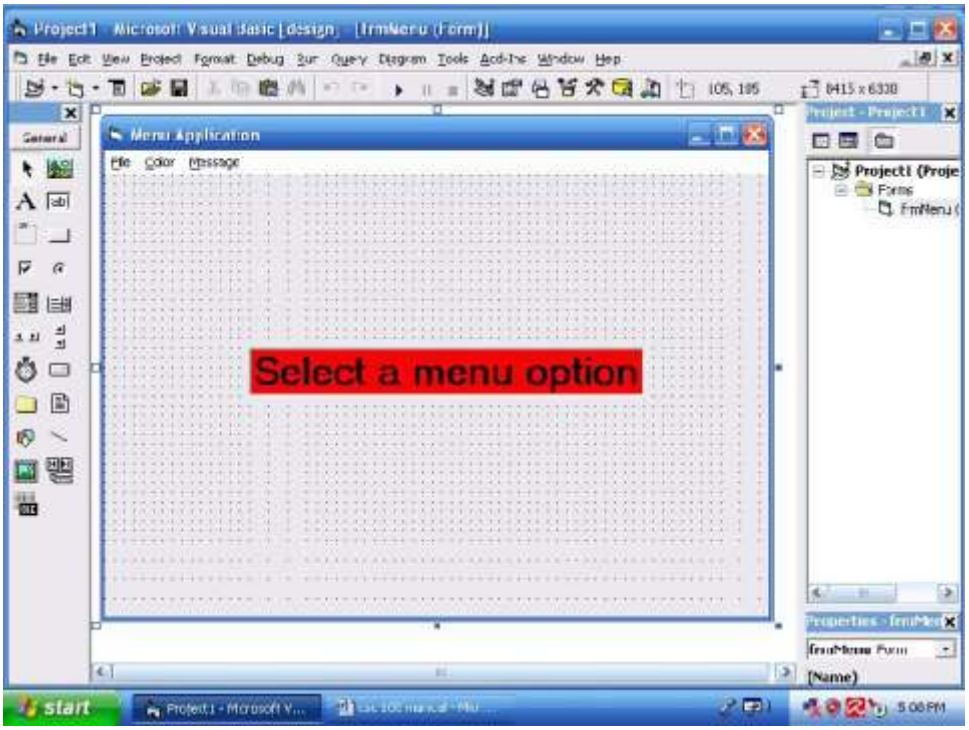


Fig.29

Guide to solution

To create the menu bar, click the form and press Ctrl+E to display the Menu Editor.

- Type &File for the Caption field. As with all other Visual Basic values, the ampersand indicates that the F will be the accelerator key for the menu selection. As you type the caption, Visual Basic displays the caption in the Menu control list box in the bottom half of the Menu Editor.
- Press Tab to move to the Name field. Tab and Shift+Tab shift the focus between the Menu Editor fields.
- Type *mnuFile* for the name of the first menu option.
- Leave all other fields alone and click the Next button to prepare for the remaining menu bar options. The Next button lets the Menu Editor know that you are through with the first option and want to enter another.
- Type &Color for the next menu bar caption and type *menuColor* for the name.
- Click Next to add the next item.

- Type &Message for the third and final menu bar caption and type menu Message for the caption. Your Menu Editor should look like the one in Fig.30.

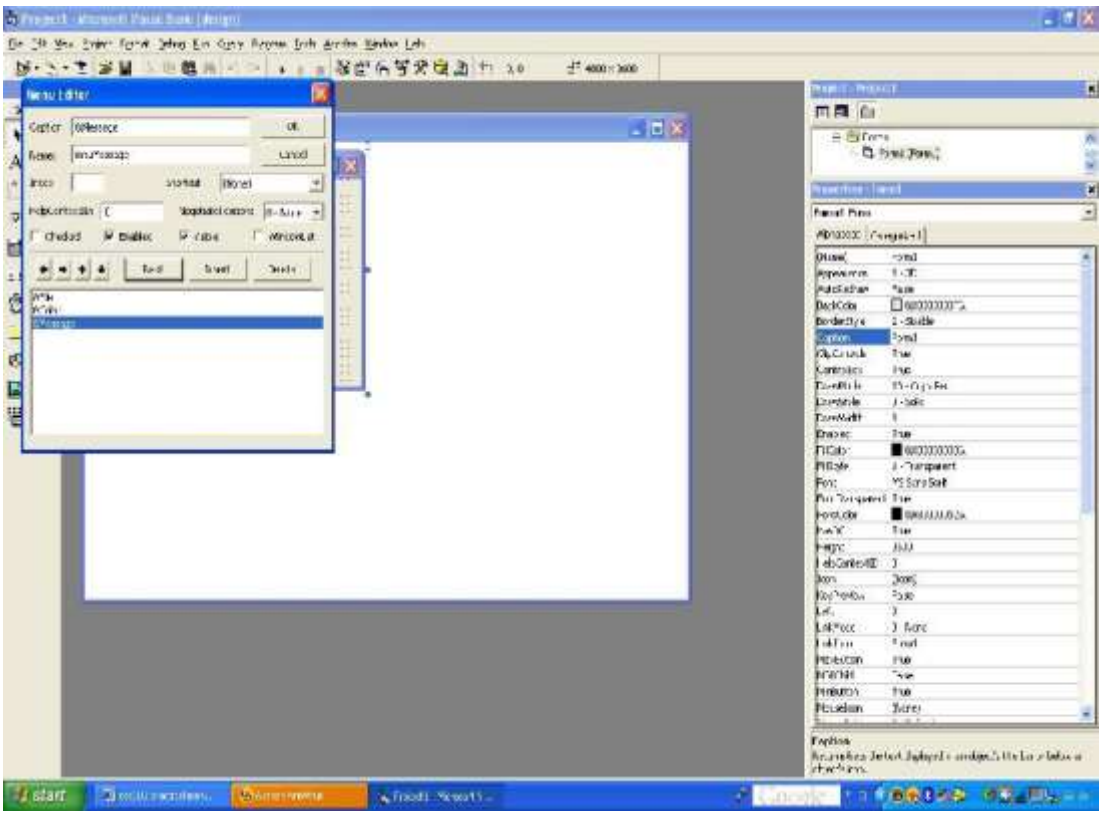


Fig.30

1.2 Adding a Pull-Down Menu

You can either create pull-down menus as you build the menu bar or add the menus later. If you create the complete menu bar first, however, as you've done in this exercise, you'll have to insert the menu options in their respective locations when you are ready to add them. The Menu Editor's Insert button lets you do just that.

SELF ASSESSMENT EXERCISE 2

.Add three checked options: Blue, Green, and Red, to the second menu, Colour. These colors will be mutually exclusive; the label will not be able to be all three colours at once, but only one colour at a time. Such colours make perfect candidates for checked menu options. Your application should like Figure 14.5 after running. Also, include a submenu to the Message menu. Let it contain the checked messages displayed in Figure 14.6 and include a separator bar as shown.

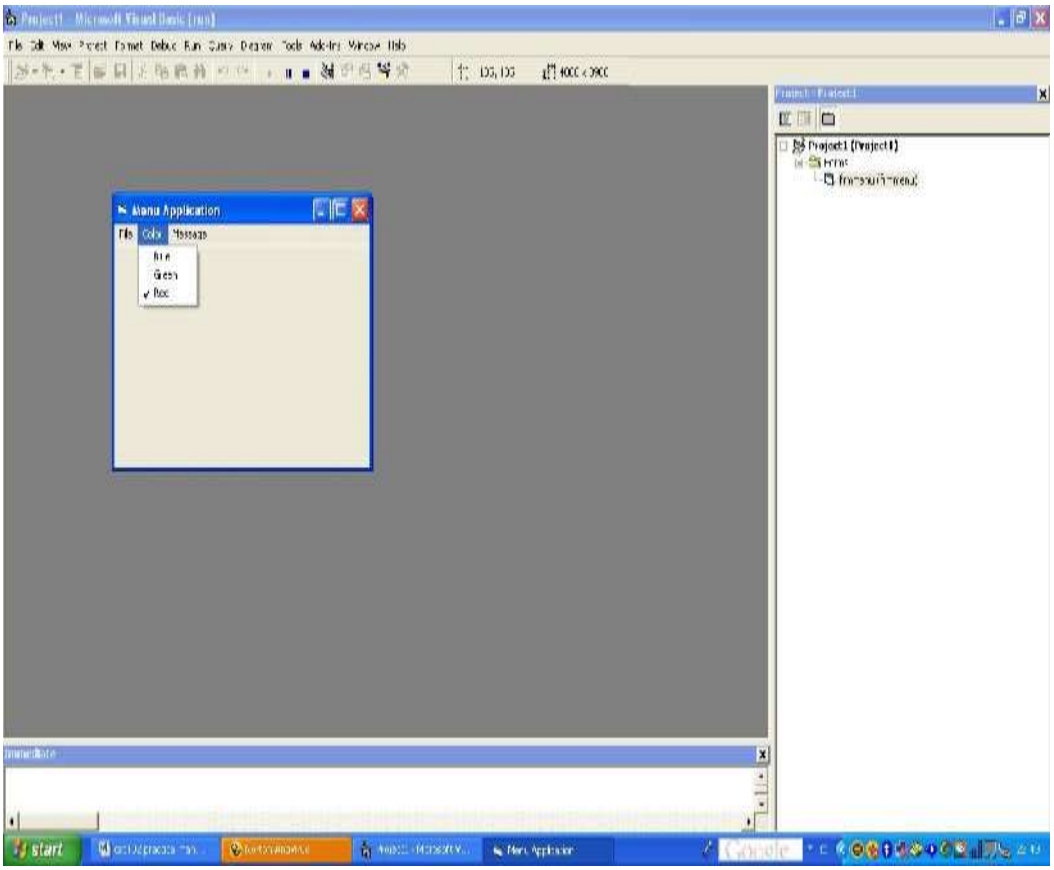

Menu Application
File
Color
Message
Blue
Green
Red

Fig.31

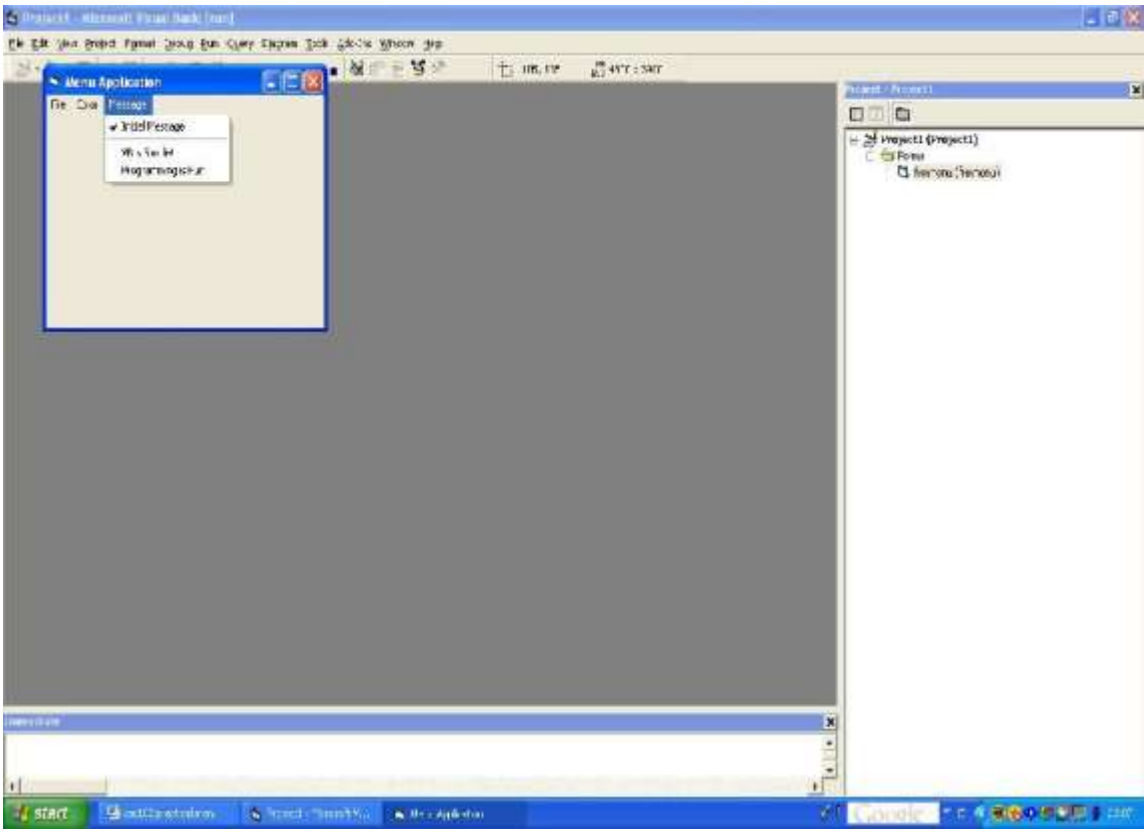

Menu Application
File
Color
Message

Fig.32

Follow these steps to do these:

- Open the Menu Editor
- Click the Message option in the Menu control list box to highlight that option.
- Click the Insert button and right arrow button three times to add three empty rows for the Color menu options.
- Highlight the first blank row where you'll add the Blue option.
- Type &Blue for the caption and mnuColorBlue in the Name field. When the user first runs the program, the Blue option will be unchecked to indicate that Blue is not currently selected.
- Click Next to enter the next option.
- Type &Green for the caption and mnuColorGreen in the Name field of the next option.
- Click Next to enter the next option.
- Type &Red for the caption and mnuColorRed for the name of the next option.
- The Red option is to be checked when the user first starts the program. Therefore, click the Checked field to place the check mark next to Red.
- Close the Menu Editor and run your application.
- To add the Message menu, display the Menu Editor and click the row beneath &Message in the lower half of the Menu Editor to prepare the Menu editor to receive the next option.
- Follow the steps as for the Color menu.
- To create the separator bar, after entering the first item, clicks Next and type a single hyphen (-) for the caption (all separator bars have this caption). Type mnuMessageSep1 as the separator bar's name.
- Fix the other options appropriately and run the application.

To finalise the menu with a code:

- Open the code window and type the code shown in Figures 33 &34. The code controls the label's colour and contents.

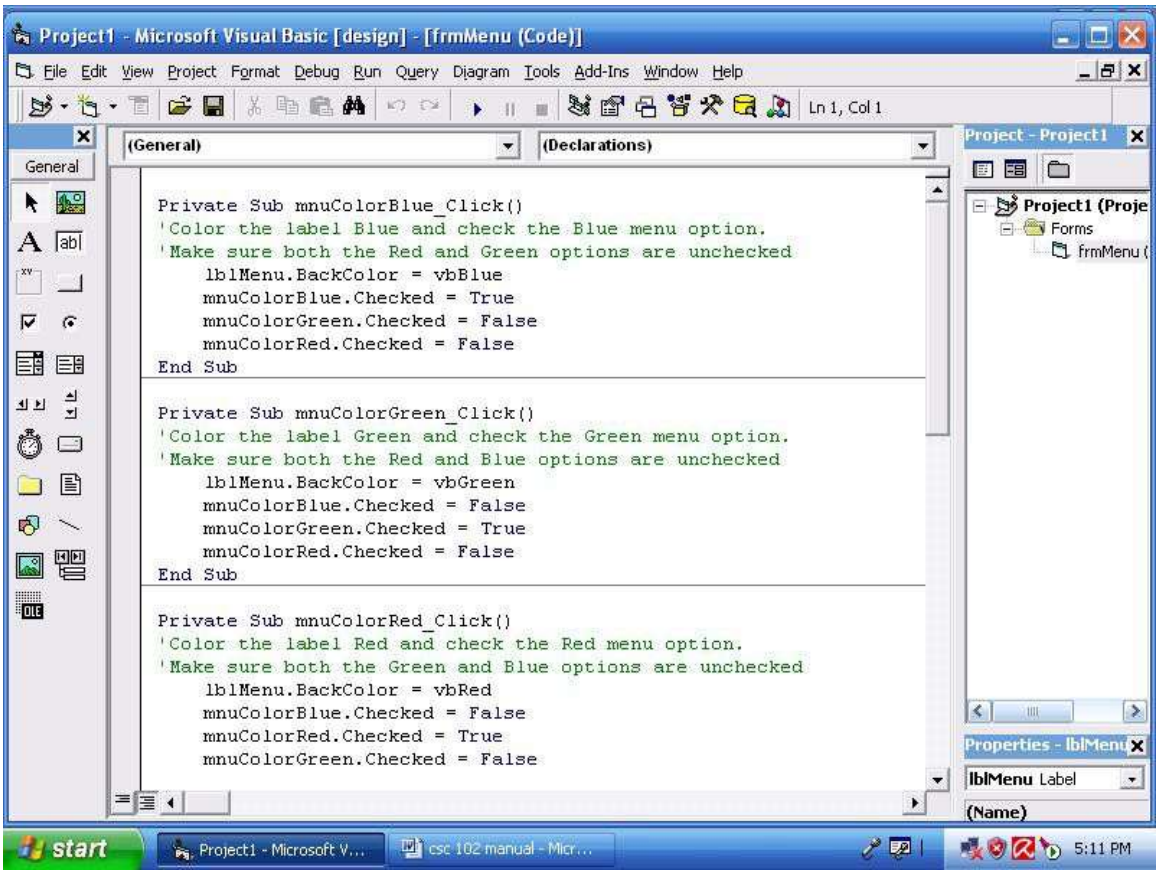

Project1 - Microsoft Visual Basic [design] - [frmMenu (Code)]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
Ln 1, Col 1
(General)
(Declarations)
Private Sub mnuColorBlue_Click()
'Color the label Blue and check the Blue menu option.
'Make sure both the Red and Green options are unchecked
lblMenu.BackColor = vbBlue
mnuColorBlue.Checked = True
mnuColorGreen.Checked = False
mnuColorRed.Checked = False
End Sub
Private Sub mnuColorGreen_Click()
'Color the label Green and check the Green menu option.
'Make sure both the Red and Blue options are unchecked
lblMenu.BackColor = vbGreen
mnuColorBlue.Checked = False
mnuColorGreen.Checked = True
mnuColorRed.Checked = False
End Sub
Private Sub mnuColorRed_Click()
'Color the label Red and check the Red menu option.
'Make sure both the Green and Blue options are unchecked
lblMenu.BackColor = vbRed
mnuColorBlue.Checked = False
mnuColorRed.Checked = True
mnuColorGreen.Checked = False
Project - Project1
Project1 (Proje
Forms
frmMenu (
Properties - lblMenu
lblMenu Label
(Name)
start
Project1 - Microsoft V...
csc 102 manual - Micr...
5:11 PM

Fig.33

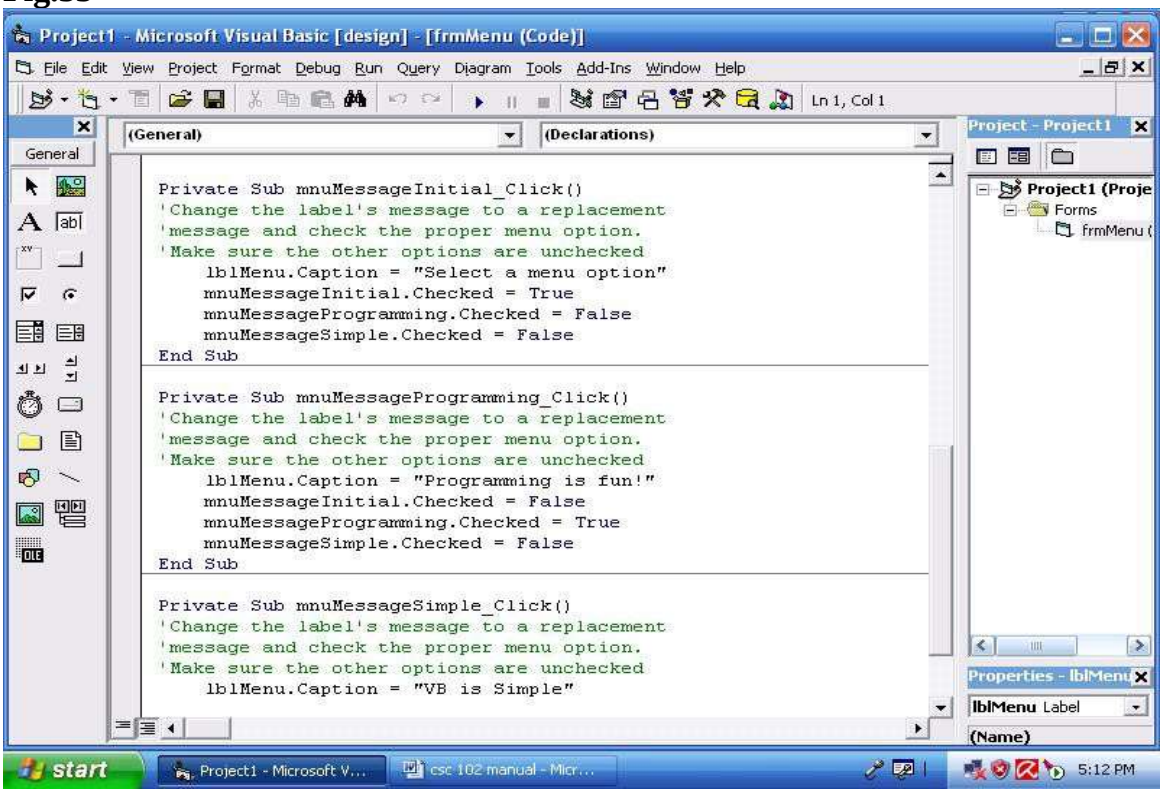

Project1 - Microsoft Visual Basic [design] - [frmMenu (Code)]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
Ln 1, Col 1
(General)
(Declarations)
Private Sub mnuMessageInitial_Click()
'Change the label's message to a replacement
'message and check the proper menu option.
'Make sure the other options are unchecked
lblMenu.Caption = "Select a menu option"
mnuMessageInitial.Checked = True
mnuMessageProgramming.Checked = False
mnuMessageSimple.Checked = False
End Sub
Private Sub mnuMessageProgramming_Click()
'Change the label's message to a replacement
'message and check the proper menu option.
'Make sure the other options are unchecked
lblMenu.Caption = "Programming is fun!"
mnuMessageInitial.Checked = False
mnuMessageProgramming.Checked = True
mnuMessageSimple.Checked = False
End Sub
Private Sub mnuMessageSimple_Click()
'Change the label's message to a replacement
'message and check the proper menu option.
'Make sure the other options are unchecked
lblMenu.Caption = "VB is Simple"
Project - Project1
Project1 (Proje
Forms
frmMenu (
Properties - lblMenu
lblMenu Label
(Name)
start
Project1 - Microsoft V...
csc 102 manual - Micr...
5:12 PM

Fig.34

When the application is run, the screens look likethose in Figures 35 & 36. For Figure 35, colour blue was selected with the second message "VB is Simple".

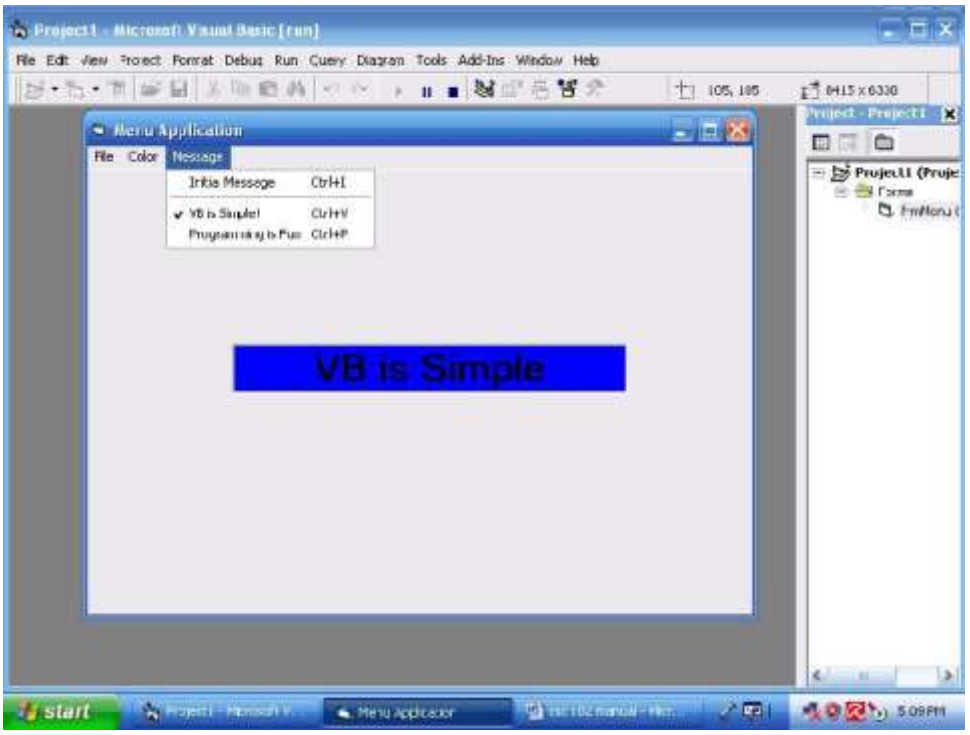


Fig.35

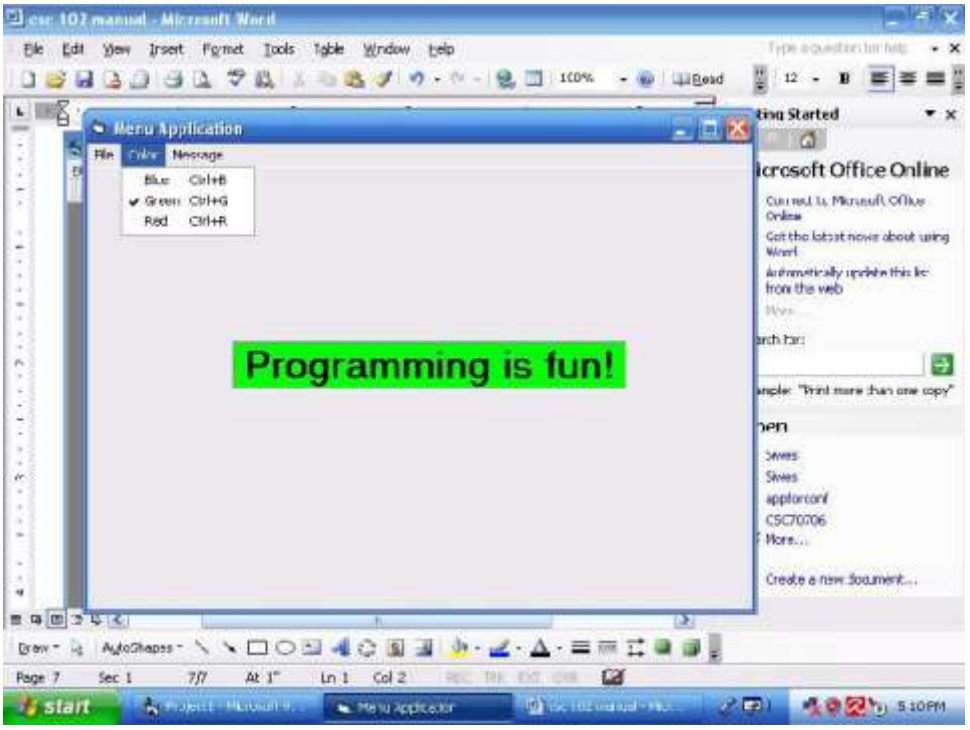


Fig.36

Analysing Visual Basic Data

Working inside the Code Window

The Code window contains several sections which include:

- The declarations section
- General-purpose procedures
- Event procedures

Figure 15.1 shows the declarations sections in the first set of codes. After the first wrapper line, there is an event procedure, followed by another one. General-purpose procedures can be meant to perform any kind of function like computing some data, while event procedures will be executed after the clicking of a mouse once or twice, the loading of a form, or any other event.

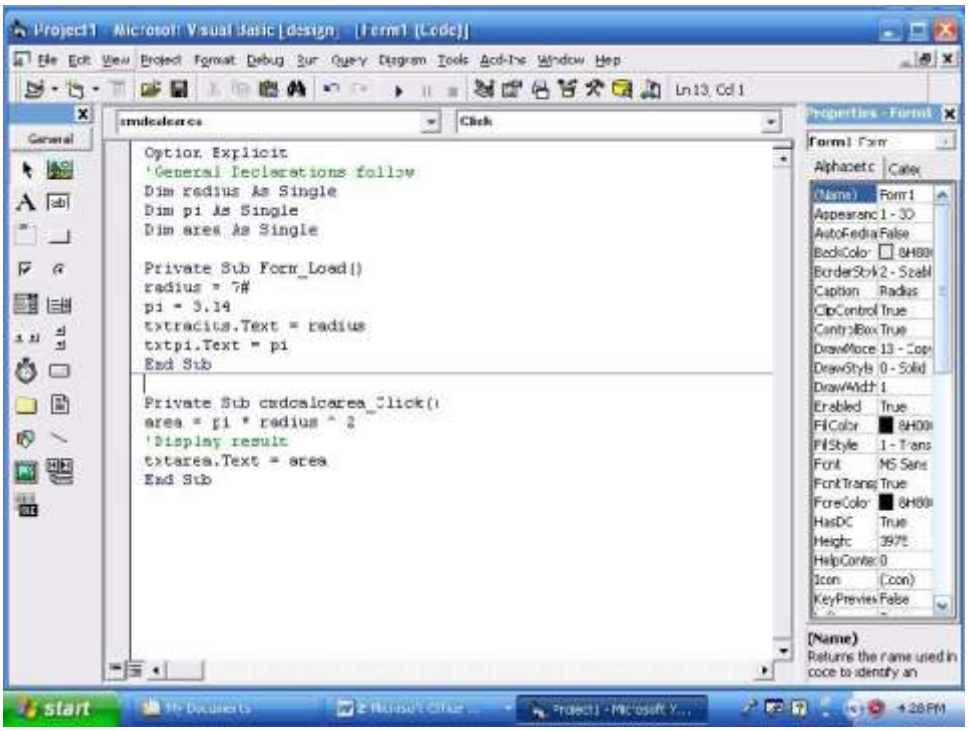


Fig.37

The output of the code displayed in Figure 37is shown in Figure 38.

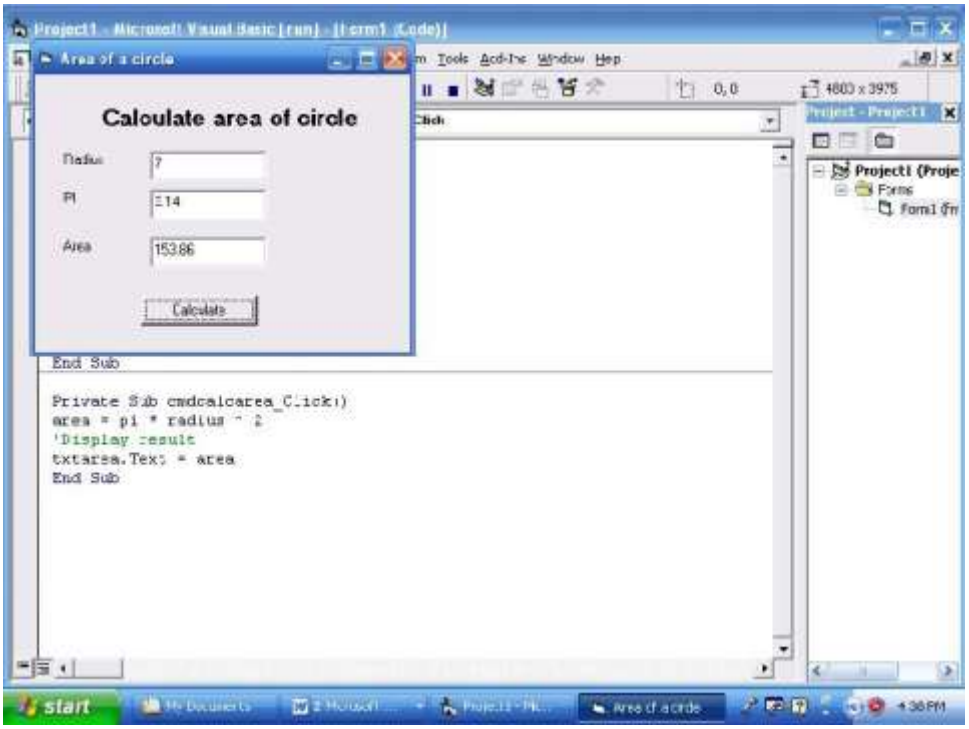


Fig.38

More of use of variables, together with the control of programs with conditional operators, logical operators and FOR DO loops, are described in the Exercises below:

SELF ASSESSMENT EXERCISE 1

Write and run a Visual Basic Program to solve the Quadratic Equation Problem, ax^2+bx+c.

Solution

The code that provides the solution to the problem using the IF-THEN statement, is shown in Figures 39 and 40. This contains statements explaining most of the basic concepts a beginning VB Programmer needs to know. The output of the program is displayed in Figure 41. As can be seen, key words are in blue, comments are in green, while the other codes are in black.

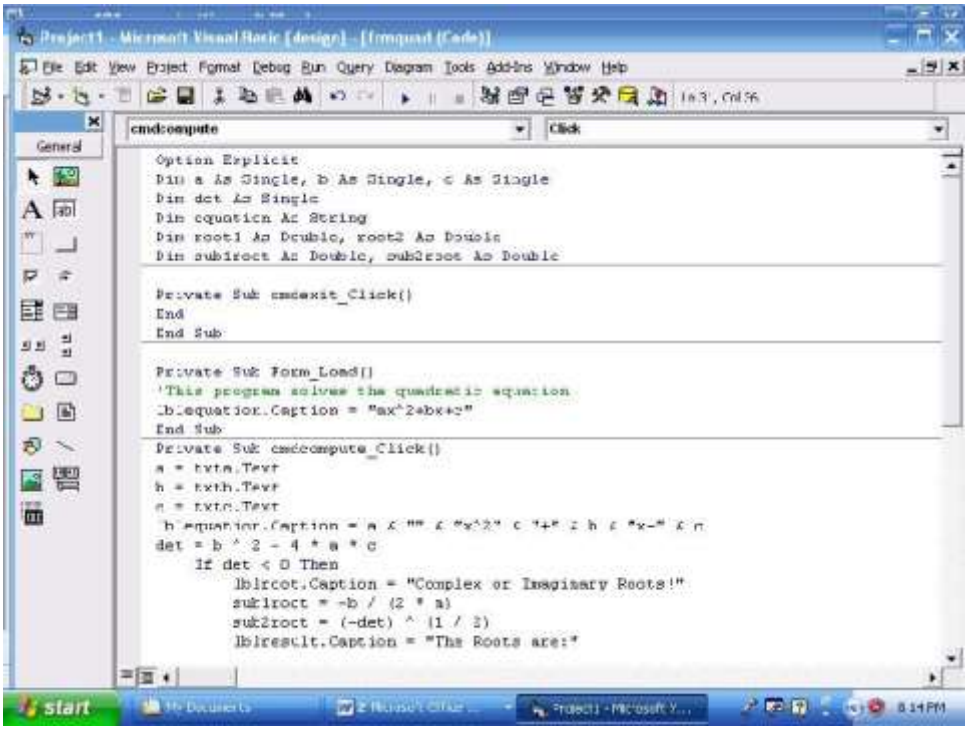


Fig.39

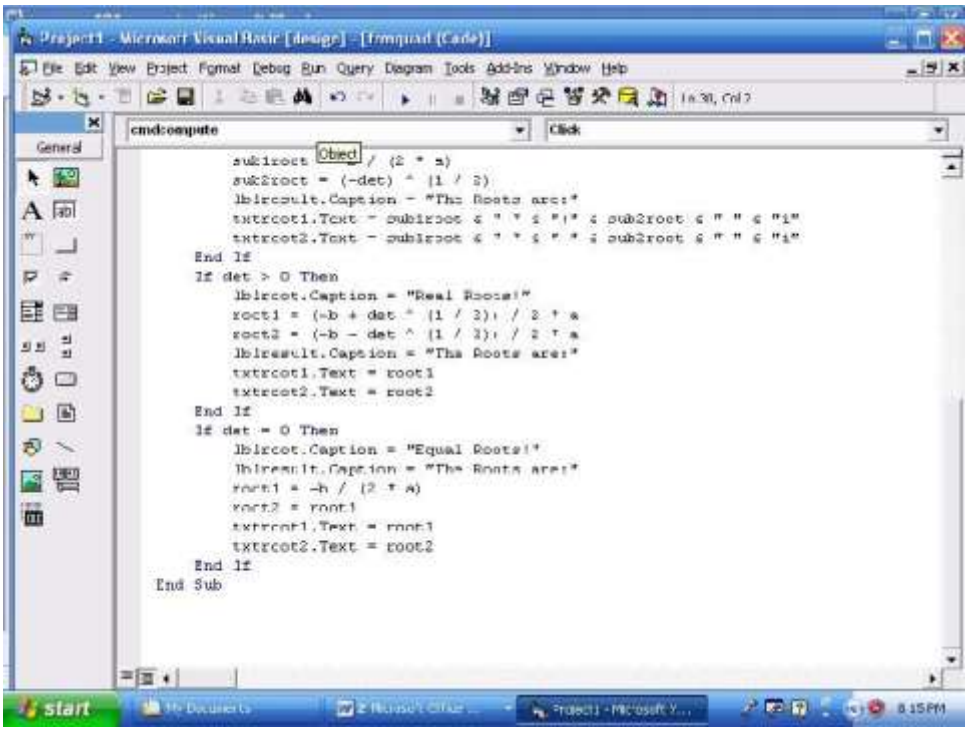


Fig.40

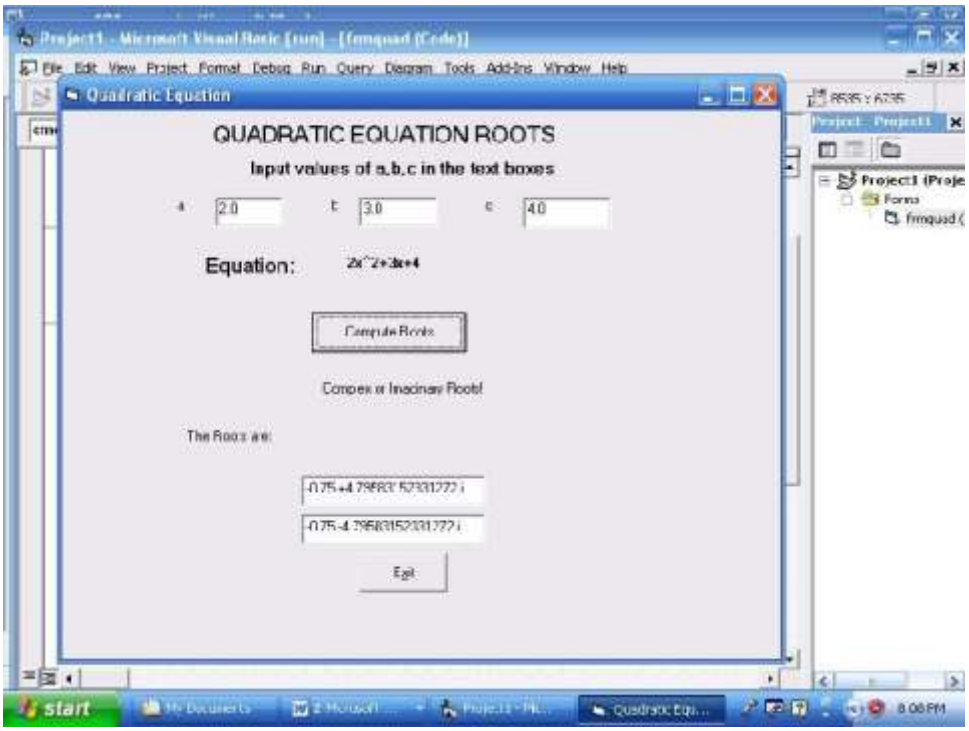


Fig.41

Solving the same problem using SELECT CASE statement, we have the codes displayed in Figure 42.

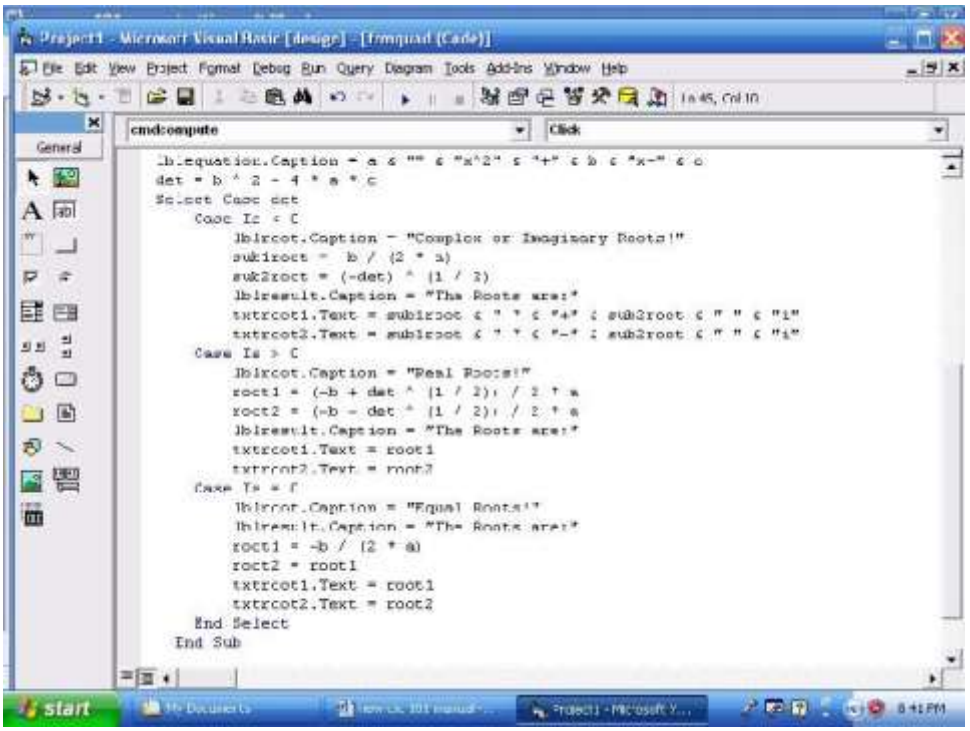


Fig.42

SELF ASSESSMENT EXERCISE 2

Write and Run a Visual Basic Payroll Program for 10 employees of a company. The Gross pay sums the Basic pay, Housing allowance and Professional allowance (where applicable). Workers' Grade levels range from 1 to 16. Housing allowance of workers is 30% of Basic pay for workers on levels 8-16 and 40% for levels 1-7 workers. Transport allowance is 20% of Basic pay for all workers. Hazard allowance is 15% of Basic pay for only levels 8-16 workers. The Net pay, which is the take home pay, is the Gross pay – Tax (10% of Gross pay). Design a form through which each worker's data can be entered (to look like the one displayed in Figure 43).

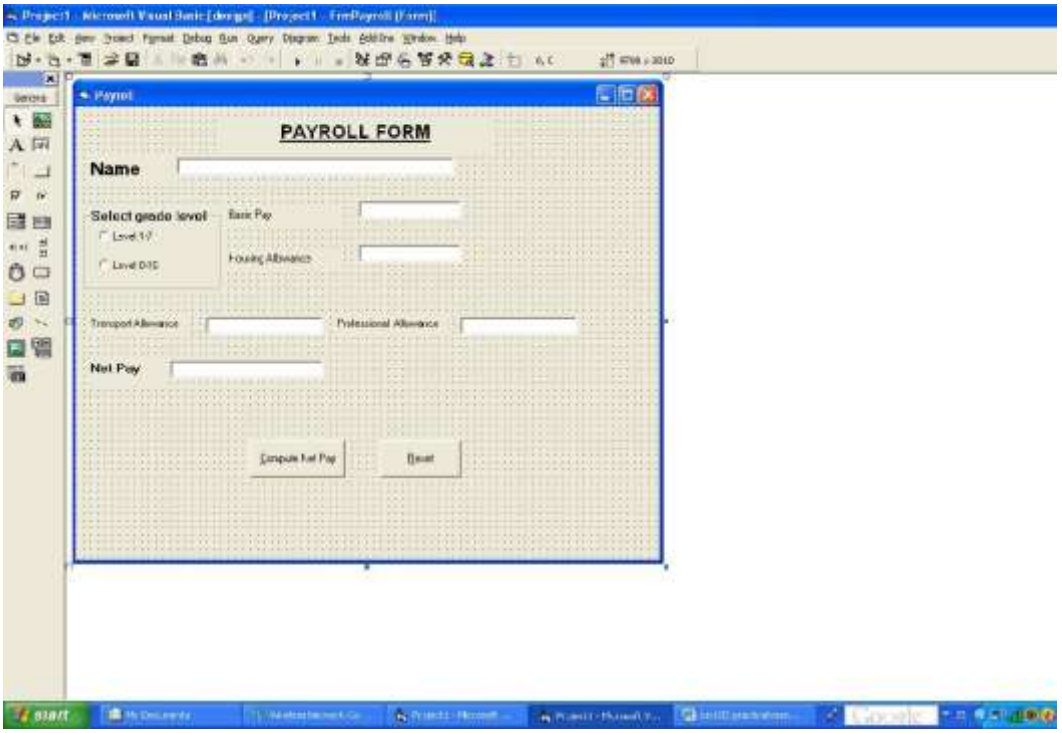

PAYROLL FORM
Name
Select grade level
Basic Pay
Housing Allowance
Transport Allowance
Professional Allowance
Net Pay

Fig.43

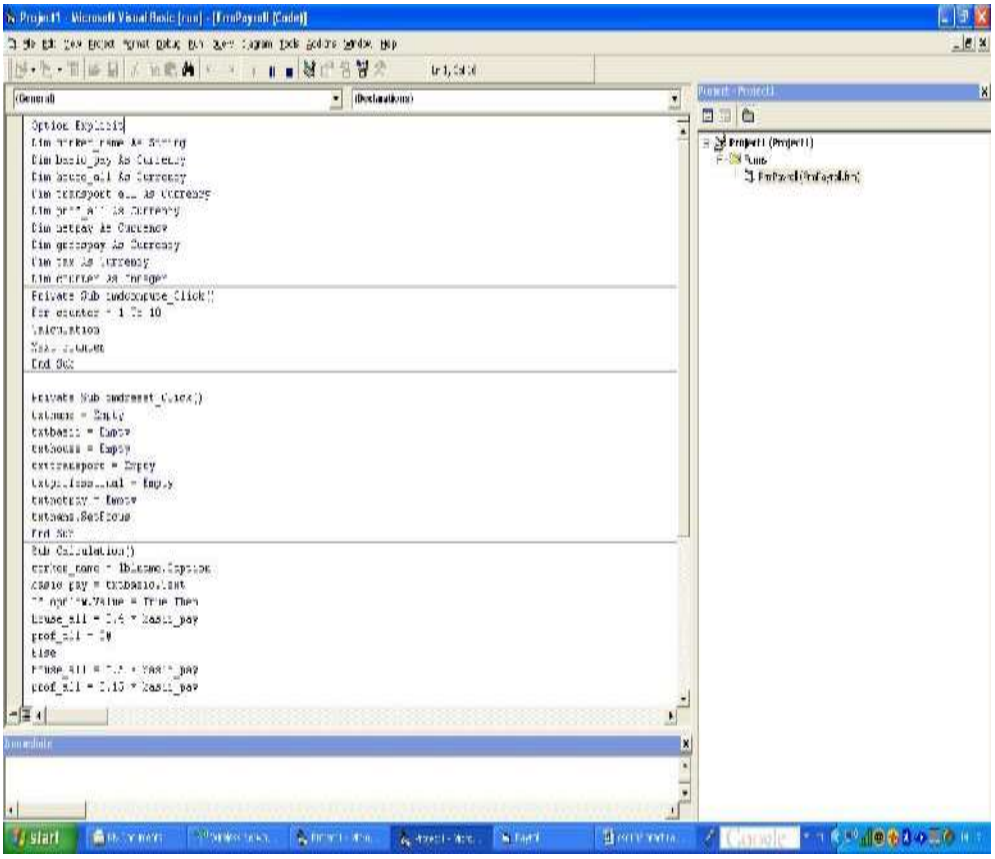


Fig.44

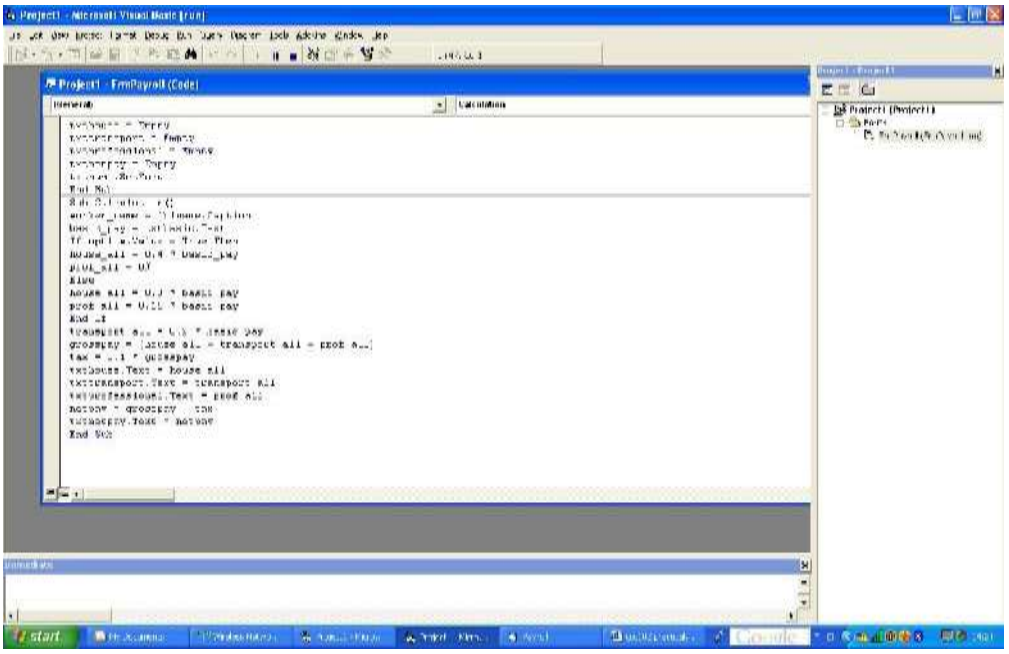


Fig.45

APPLICATION OF COMPUTERS IN EDUCATION

1.1 Application of Computers in Education

The field of education provides the most fascinating application of the computing system. This has consequently attracted considerable attention from educationists and policy makers since the late 1960s, when computers were introduced into the classroom. The entry of the computer into the classroom has now offered opportunities and possibilities for students to develop their potential with computer-aided instruction packages. A considerable number of fascinating and entertaining educational computer software packages are now available in almost every subject. These self-tutor instruction packages are well designed to enable the user to learn at their own time, speed and convenience. The role of the teacher here is that of a guide so that the student can think more logically and can gain meaningful experience in such structured situations, role playing and other well programmed exercises so that the student can have a better understanding of the interrelationships of variables to real life situations. Apart from the use of the computer as an instructional aid, it is also used in the execution of routine and administrative tasks such as the keeping of academic and administrative records on admissions, examinations, staffing and other routine functions.

The computer has also revolutionised the services rendered by libraries to readers. A computer based on-line public access catalogue system manages a search for materials using indices such as author's name, book title, subject and class mark. Readers using public terminals can go through a menu-driven programme to find specific books or periodicals, recall books on loans and also make requisitions for short loan items.

Furthermore, optical character reading devices are used to scan the bar codes on readers' library cards to offer a computer based issuing of books to readers. A computer based security system is used to maintain security services in libraries as well. In educational institutions such as nursery schools, primary schools, secondary schools, polytechnics, colleges of education and universities, the computer can be used for the following:

Computer aided self-t

- ✓ utored application packages.
- ✓ Computer instructional aids e.g. digital projectors.
- ✓ Microsoft PowerPoint application software for preparing slide shows, speeches, seminars, workshops, lectures etc.
- ✓ Computer simulated graded exercises and group work.
- ✓ Computer-aided laboratory experiments and investigations.
- ✓ Computer-aided software packages for special students e.g. the mentally/physically disabled (the blind, deaf etc), adults, KGs, teenagers etc.
- ✓ Distant learning programmes e.g. sandwich programmes, through the virtual library technology.
- ✓ Teleconferencing technology.
- ✓ Placing and sourcing of educational materials/resources e.g. papers, journals, newsletters, magazines, textbooks, and films, on the Internet.
- ✓ Computerisation of library services to make cataloguing indexing, retrieval, borrowing, return and other library services easier and faster.
- ✓ Keeping the records of students, teachers and teaching facilities.
- ✓ Estimating the teacher-student ratio with a view to assessing the adequacy of teaching and learning.
- ✓ Estimating the ratio of students to teachers and teachers to teaching facilities with a view to assessing the adequacy of teaching and learning.
- ✓ Timely generation of students' examination results.
- ✓ Automatic generation of lecture and examination time tables.
- ✓ Aiding students to learn basic theoretical concepts. There are currently, some computer aided learning software packages and hardware devices that are readily available in the market.