KASNEB

DICT

LEVEL II

Programming concepts

Contents

TOPIC 1

INTRODUCTION

Programs

A computer program is a series of organised instructions that directs a computer to perform tasks. Without programs, computers are useless.

A program is like a recipe. It contains a list of variables (called ingredients) and a list of statements (called directions) that tell the computer what to do with the variables.

Programming

Programming is a creation of a set of commands or instructions which directs a computer in carrying out a task.

Programming languages

A programming language is a set of words, symbols and codes that enables humans to communicate with computers.

Examples of programming languages are:

- BASIC (Beginner's All Purpose Symbolic Instruction Code)
- Pascal
- C
- Smalltalk.

Types of programing languages

There are three main kinds of programming language:

- Machine language
- Assembly language
- High-level language

We just went over what **machine language** is - it's the language of machines, consisting of bits (1s and 0s) put together into chunks like **bytes**, a group of 8 bits, and lots of other larger sizes. It's highly unlikely you will ever have to write in machine language, but in the old days, we used to plot 1s and 0s on graph paper and then type them in, to make pictures appear on the computer screen. Very tedious!

.

TOPIC 4

LANGUAGE TRANSLATION PROGRAMS

A **translator** is a computer **program** that performs the **translation** of a **program** written in a given **programming language** into a functionally equivalent **program** in a different computer **language**, without losing the functional or logical structure of the original code (the "essence" of each **program**).

A computer language translator is a program that translates a set of code written in one programming language into a functional equivalent of the code in another programming language.
The different types of computer translators are

Assembler

An **assembler** translates assembly language into machine code. Assembly language consists of mnemonics for machine opcodes so assemblers perform a 1:1 translation from mnemonics to a direct instruction. For example:

`LDA #4` converts to `0001001000100100`

Conversely, one instruction in a high level language will translate to one or more instructions at machine level.

Advantages of using an Assembler:

- Very fast in translating assembly language to machine code as 1 to 1 relationship
- Assembly code is often very efficient (and therefore fast) because it is a low level language
- Assembly code is fairly easy to understand due to the use of English-like mnemonics

Disadvantages of using Assembler:

- Assembly language is written for a certain instruction set and/or processor

- Assembly tends to be optimised for the hardware it's designed for, meaning it is often incompatible with different hardware
- Lots of assembly code is needed to do relatively simple tasks, and complex programs require lots of programming time

Compiler

A **Compiler** is a computer program that **translates code** written in a high level language to a lower level language, object/machine code. The most common reason for translating source code is to create an executable program (converting from a high level language into machine language).

Advantages of using a compiler

- Source code is not included, therefore compiled code is more secure than interpreted code
- Tends to produce faster code than interpreting source code
- Produces an executable file, and therefore the program can be run without need of the source code

Disadvantages of using a compiler

- Object code needs to be produced before a final executable file, this can be a slow process
- The source code must be 100% correct for the executable file to be produced

Interpreter

An interpreter program executes other programs directly, running through program code and executing it line-by-line. As it analyses every line, an interpreter is slower than running compiled code but it can take less time to interpret program code than to compile and then run it — this is very useful when prototyping and testing code. Interpreters are written for multiple platforms, this means code written once can be run immediately on different systems without having to recompile for each. Examples of this include flash based web programs that will run on your PC, MAC, games console and Mobile phone.

Advantages of using an Interpreter

- Easier to debug(check errors) than a compiler
- Easier to create multi-platform code, as each different platform would have an interpreter to run the same code
- Useful for prototyping software and testing basic program logic

Disadvantages of using an Interpreter

- Source code is required for the program to be executed, and this source code can be read making it insecure

- Interpreters are generally slower than compiled programs due to the per-line translation method

What is a Software Defect, Bug and Debugging

What is a Software Defect / Bug?

A software bug is an (Error, Flaw, Failure, or Fault) In a computer program or system that produces an incorrect or unexpected result, or causes it to behave in unintended ways.

TOPIC 6

PROGRAMMING TOOLS

The **programming** is a solution of different problems of our real life. If efficient programming tools are used, problems are effectively solved. We code some instructions to instruct **computer** for problem solving purposes. The choice of tools depends on nature of problems.

There are many tools for programmers for programming. For instance, **algorithms**, **flowcharts**, **pseudocodes**, **data dictionary**, **decision table**, **data flow diagrams** etc are effective tools. Enough and adequate knowledge of programming tools are essential for programming (Software development).

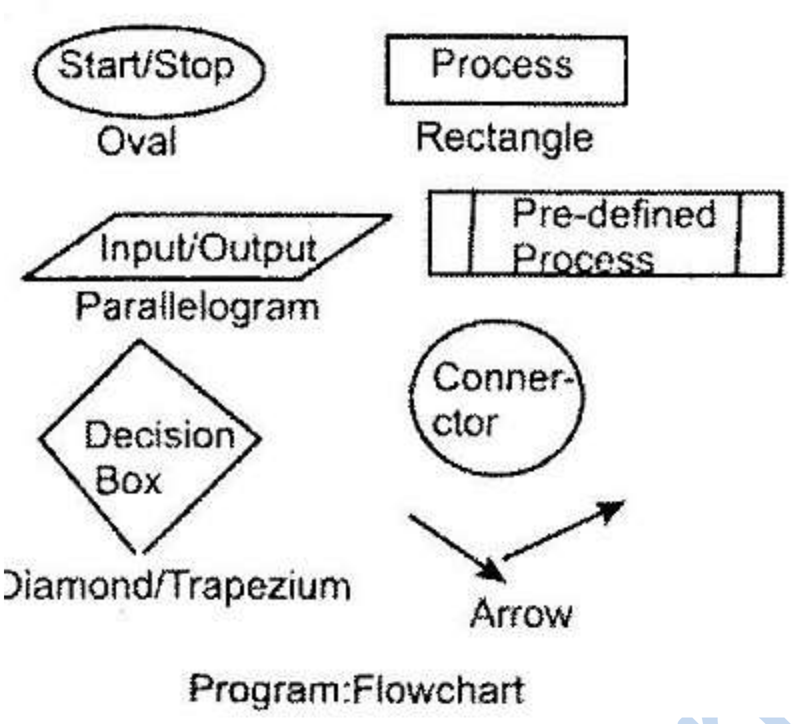
Flowchart

The pictorial presentation of program is called *Flowchart*. It is a tool and technique to find out solution of programming problems through some special symbols. It is a way to represent program using geometrical patterns. Prior to 1964, every manufactures use different types of symbols, there was no uniformity and standards of flowcharting. The Standard symbols were developed by American Standard National Institute (ANSI).

There are two types of Flow Chart:

(a) Program Flow Chart and

(b) System Flow Chart.



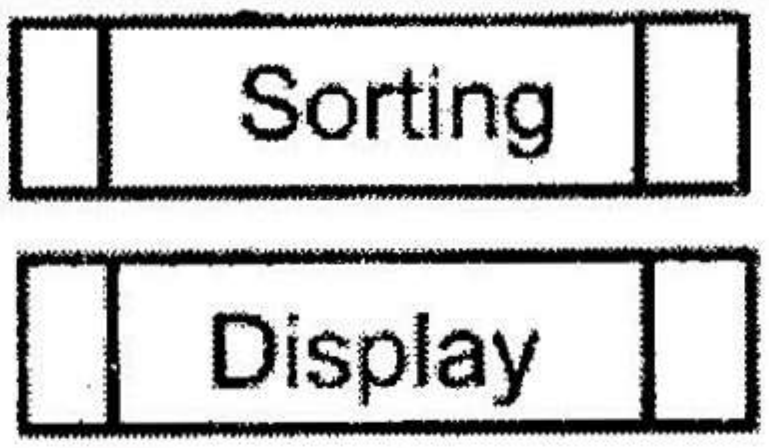
Flow Chart symbols

(i) **Start / Stop*:*** This oval is used to represent **START** and **STOP** of program.

(ii) **Input/Output**: Parallelogram is used to denote input and output of data. We take data through it and display result also using this symbol.

(iii) **Process:** Rectangle is used to denote process, formula, increment, and decrement and assigned value also.

(iv) **Pre-defined Process**: The predefined process is denoted by this symbol. *Example*: Sorting and Display is pre-defined process and represented as:



Pre-defined Process

(iv) **Decision box**: It is a symbol of decision. All type of decisions is written in it.

(v) **Connector:** It is used to link to segment of flowchart to complete as one.

(vi) **Data flow**: It is used to show data flow from one component to other component, one process to other and one symbol to other symbol.

The **John Von Neumann** used flowchart to solve problem in 1945 and explained its importance in program designing.

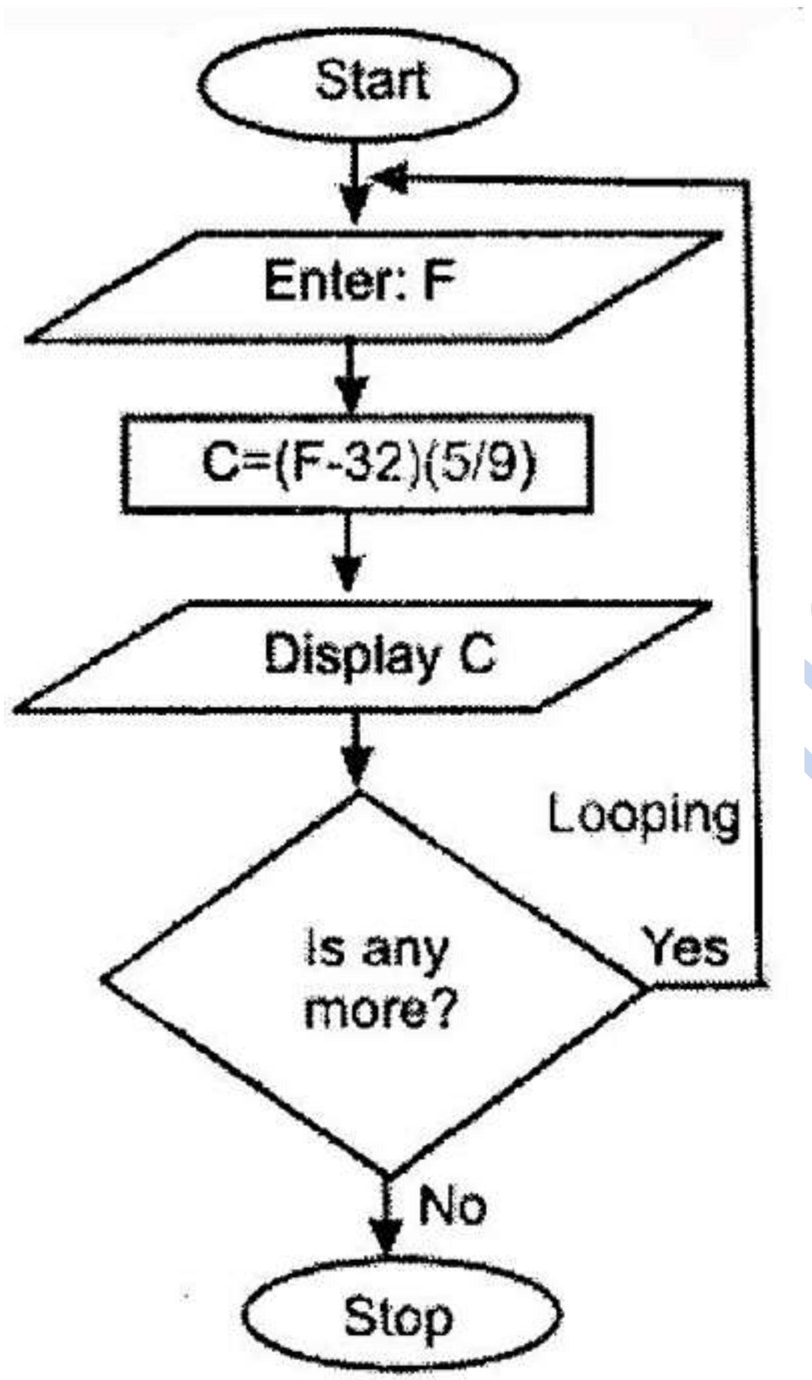
Illustration: In our everyday life, there is a sequence of works. These sequences of works are called routine. We all are tied with it. Suppose, we arise early in the morning at 5 0' clock, go to toilet, after coming from toilet, clean hands with soap and water, brush our teeth then take bath. Here, one sequence of work is formed.

In computer world, programming language is used to solve certain problems. The solution of problem is a sequence of processes. In one problem, there may be many processes, the flowchart help to arrange processes in definite order. Suppose, we have to convert Fahrenheit temperature into Celsius, it has following five steps:

- **Step-1**: Start
- **Step-2**: Enter Fahrenheit temperature (F)
- **Step-3**: Application of formula: C=(F-32)(5/9)
- **Step-4**: Displaying the result (C)
- **Step-5**: Stop

These types of processes involved in solution of problem are called *Algorithm*. It is a stepwise presentation of problem solution in simple English without using confusing words and phrases.

When we draw flowchart, it is only conversion of steps or algorithm in special symbols.



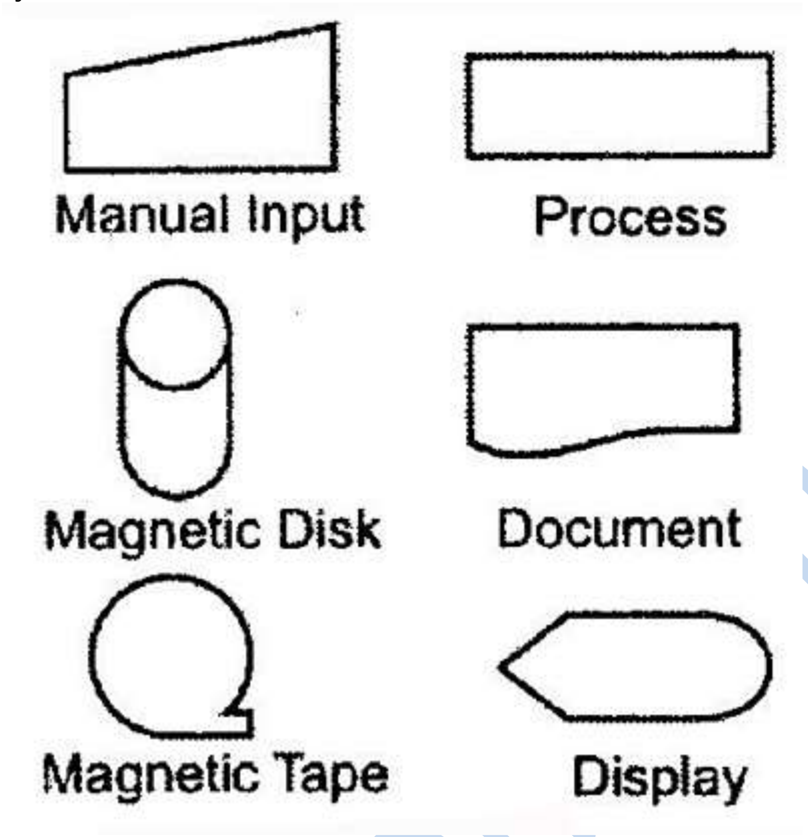
FLow Chart diagram

Flowchart is like as city map. Travelers or tourists search places of historical importance through city map. Buildings are built on the basis of building-design (map) drawn by Architect engineers.

Just like it, program is coded on the basis of flowchart. Flowchart is a language free concept, if it is prepared, any language can be used for program coding. In software development cycle, flowcharting is one essential phase.

System Flowchart:

System flowchart is a pictorial representation of procedure flows inside and outside of the systems. It is also called **Data Flow Chart** or **Procedure Chart.**



System Flowchart Symbols

(i) **Manual Input**: It is used to enter data manually. For example, keyboard is a manual input device.

(ii) **Process**: All type of processes is denoted by Rectangle.

(iii) **Magnetic Disk**: Mass storage device (Hard Disk).

(iv) **Magnetic Tape**: Magnetic tape is also storage device.

(v) **Display**: This is a symbol for online display, Example: Monitor (VDU).

(vi) **Document**: The print out document is denoted by this symbol.

TOPIC 7

VISUAL BASIC PROGRAMMING LANGUAGE

VISUAL BASIC is a high level programming language that evolved from the earlier DOS version called BASIC. **BASIC** means **B**eginners' **A**ll-purpose **S**ymbolic **I**nstruction **C**ode. The code looks a lot like English Language. Now, there are many versions of Visual Basic available in the market, the lastest being Visual Basic 2015 that is bundled with other programming languages such as C#. However, the most popular one and still widely used by many VB programmers is none other than Visual Basic 6. .

VISUAL BASIC is a VISUAL Programming Language because programming is done in a graphical environment. In VB6 , you just need to drag and drop any graphical object anywhere on the form and click on the object to enter the code window and start programming.

Visual programming environment

Software which allows the use of visual expressions (such as graphics, drawings, animation or icons) in the process of programming. These visual expressions may be used as graphical interfaces for textual programming languages. They may be used to form the syntax of new visual programming languages leading to new paradigms such as programming by demonstration or they may be used in graphical presentations of the behavior or structure of a program.

Integrated development environment (IDE)

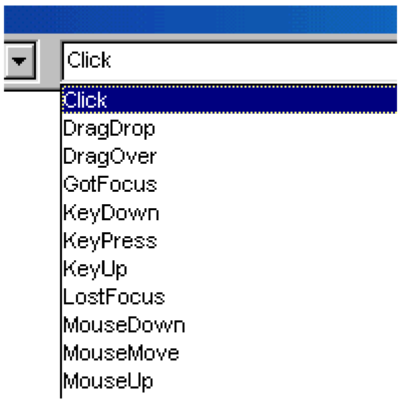
An integrated development environment (IDE) is a programming environment that has been packaged as an application program, typically consisting of a code editor, a compiler, a debugger, and a graphical user interface (GUI) builder.

(**I**ntegrated **D**evelopment **E**nvironment) A set of programming tools for writing applications (source code editor, compiler, debugger, etc.), all activated from a common user interface and menus. IDEs are necessary standard procedure for program development.

<u>Using the Visual Basic Development Environment</u>

In addition to the core user interface features that Visual Studio provides, the Visual Basic integrated development environment (IDE) adds some to help with productivity and ease of use.

Note: Visual Basic makes it easy to create **event-driven** programs. An event-driven program is one which responds to users' actions especially mouse clicks and mouse movement.



some user events associated with a command button

Visual Basic Graphical User Interfaces

Objects and Classes

An object is a combination of code and data that can be treated as a unit. An object can be a piece of an application, like a control or a form. An entire application can also be an object.

Each object in Visual Basic is defined by a class. A **class** describes the variables, properties, procedures, and events of an object. Objects are instances of classes; you can create as many objects you need once you have defined a class.

To understand the relationship between an object and its class, think of cookie cutters and cookies. The cookie cutter is the class. It defines the characteristics of each cookie, for example size and shape. The class is used to create objects. The objects are the cookies.

Two examples in Visual Basic might help illustrate the relationship between classes and objects.

- The controls on the Toolbox in Visual Basic represent classes. When you drag a control from the Toolbox onto a form, you are creating an object — an instance of a class.
- The form you work with at design time is a class. At run time, Visual Basic creates an instance of the form's class — that is, an object.

TOPIC 8

EMERGING ISSUES AND TRENDS

The Rise and Fall of Visual Basic, Great empires often fall from within.

The death knell for Visual Basic is premature, but it's true that VB has deviated from its original vision as an "Application Construction Kit" for the masses and has lost significant market share as a result.

Tim Anderson summed it up best: It sounds like perfection. Microsoft had perhaps the largest number of developers in the world hooked on a language which in turn was hooked to Windows. Yet Microsoft took this asset of incalculable value and apparently tossed it aside. Back in 2002, Microsoft announced that the language was to be replaced by something new, different and incompatible. That caused rumblings that continue today. Developers expressed emotions ranging from frustration to anger. They felt betrayed.

Much has been written lately about the fall of Visual Basic, triggered by an Evans Data survey indicating that VB use has dropped 35% in the past year, and other language surveys show VB falling behind its brother C# and market leader Java.

The problem is simply that when Visual Basic became VB.NET, it became a "real" programming language for trained developers, no longer the layman's "Application Construction Kit" of its original vision. As such, there's little to positively distinguish VB from the other .NET programming languages, especially the superior and more popular C#. The result is an expected drop in market share.

Perhaps next-generation Web development environments like Popfly and Silverlight will fill the gap left by VB. And there is a concerted effort including a web petition to convince Microsoft to support and upgrade the last "simple" version of Visual Basic, VB6. This support is unlikely, however, and VB's reign as "programming language for the masses" is over.

Humble Beginning

Alan Cooper is widely regarded as the father of Visual Basic. In 1987, Cooper was a director at Coactive Computing Corporation where he developed "Tripod," an improved shell/desktop for the fledgling Windows operating system. After initial testing, Cooper realized that "every user would need their own personal shell, configured to their own needs and skill levels." The idea of a "shell construction set" was born. There would be a palette of tools and controls, which users could drag & drop onto forms to create their custom shell.

Cooper began shopping the product around Silicon Valley seeking a publisher. There was little interest until March 1988 when Cooper showed a prototype to Microsoft CEO Bill Gates. Visionary that he is, the 32-year-old billionaire immediately saw Tripod's potential. Gates declared that Tripod was "cool" and would have significant impact across Microsoft's entire product line. In a few months the deal was done, Tripod became Microsoft's "Ruby," and Cooper assembled a team of engineers to deliver a commercial product.