KEMENTERIAN PELAJARAN MALAYSIA

Self Access Learning Module

ICT Literacy for Secondary School Programme

PUSAT PERKEMBANAGN KURIKULUM
KEMENTERIAN PELAJARAN MALAYSIA

INTRODUCTION TO PROGRAMMING

Learning objectives:

At the end of this lesson, you should be able to:

1. define ***programming;*** and

2. state the main steps in programme development.

1.1 What is programming?

Programming language is a set of rules that provides a way of telling the computer what operation to perform. (according to Capron and Johnson,2004)

1.2 What programming languages are available?

Before looking at a specific programming language, we need to know the levels of programming languages.

Language is said to be "lower" or "higher" depending on how close they are to the language the computer itself uses or to the language people use (more English-like-high).

These programming languages are generally divided into five levels or generations:

i) Machine Language
ii) Assembly Language
iii) High-Level Language
iv) Very High-Level Language
v) Natural Language

The following are the descriptions of each level of programming language:

Level of language	Descriptions
Machine Language	-Ultimately the computer understands only binary number-strings of 0s and 1s. -Programs that are written in these 0s and 1s represent the "on" and "off" electrical states of computer -All other languages must be translated into machine language before executing instructions.

Assembly Language	-This programming language is considered very low level. -This language use mnemonic codes, abbreviations that are easy to remember such as: A for add, C for compare, MP for multiply and so on.
High- Level Language	-For this language, programmers no longer need to have detailed knowledge of computer hardware to produce a programme. - This language is closer to human language compare to the machine language.
Very High-Level Language	-Often known as fourth generation language(4 GLs) - This programming language is an improvement of high-level language.
Natural Language	- This programming language is also called the fifth generation language. - This language is similar to the “natural” spoken or written English Language. -The natural language translates human instructions into code that the computer can understand and execute.

1.3 Examples of Programming Languages

Examples of programming languages are like

a) BASIC

b) COBOL

c) PASCAL

d) C

e) FORTRAN

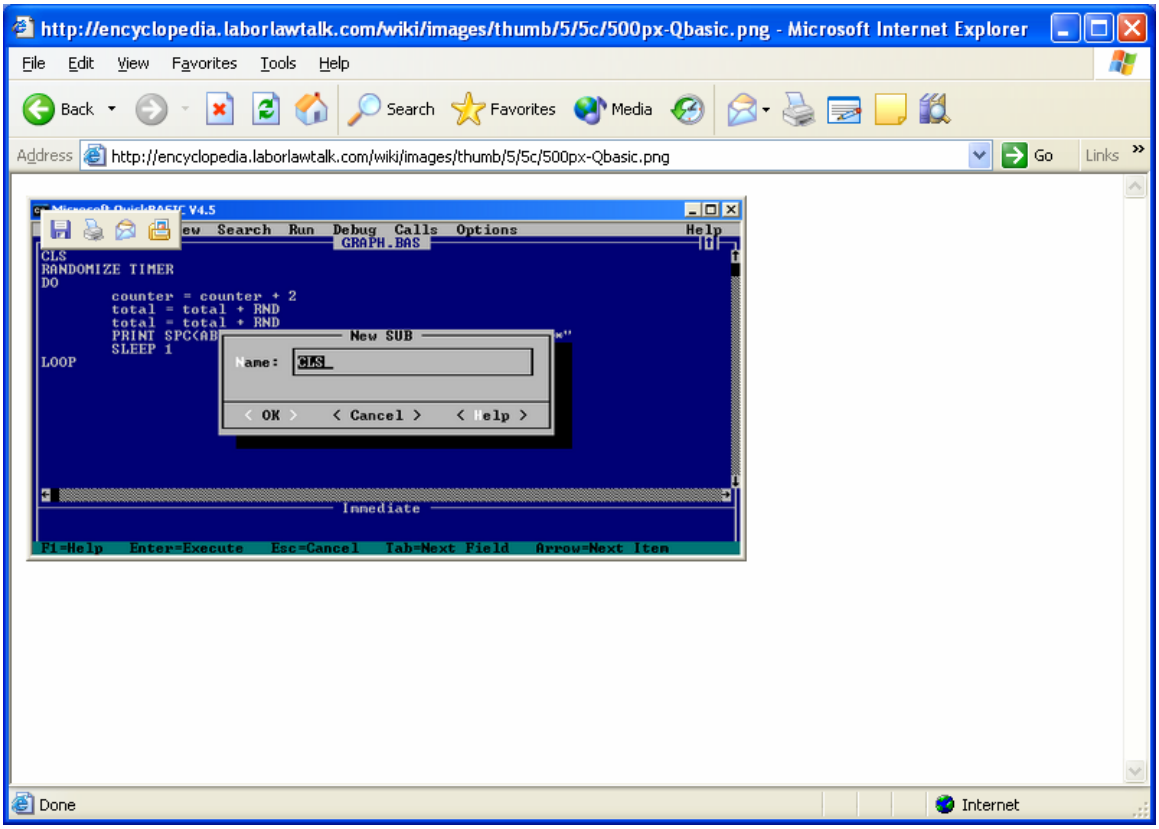
f) VISUAL BASIC

g) C++

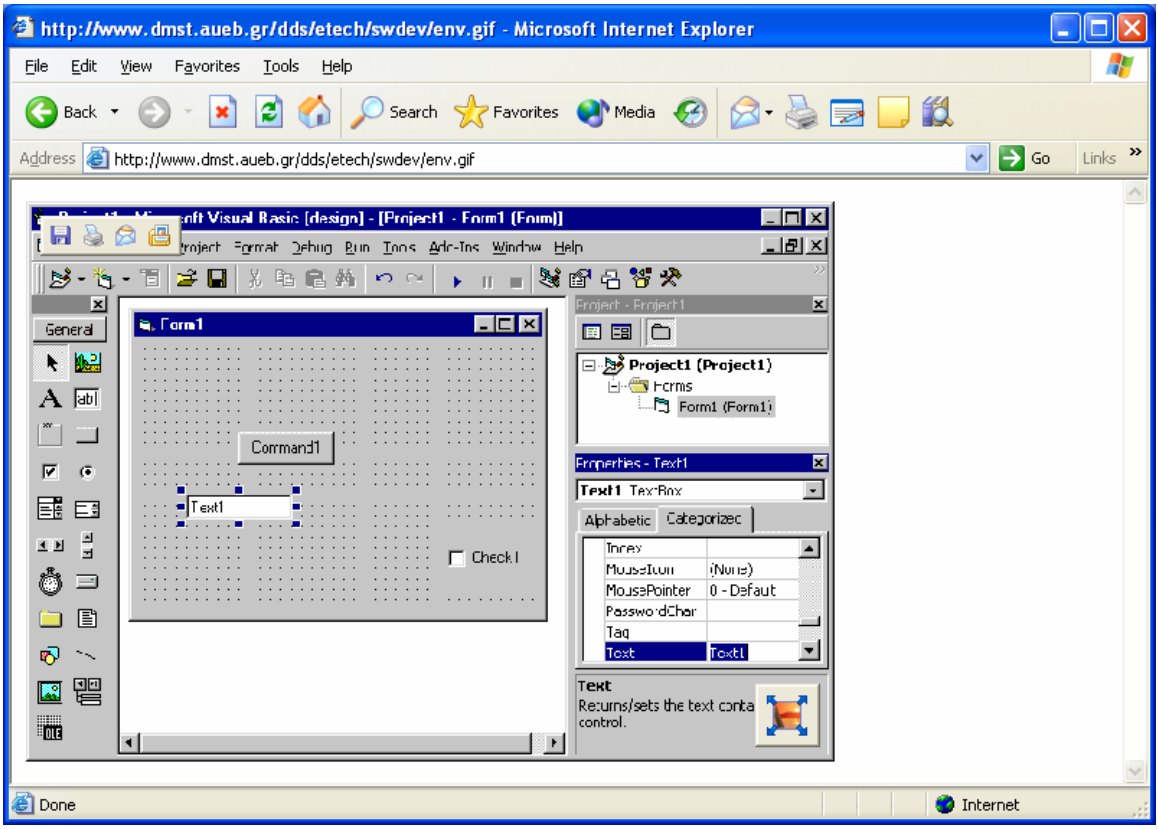
h) JAVA

Let's look at samples of each interface for the programming languages:

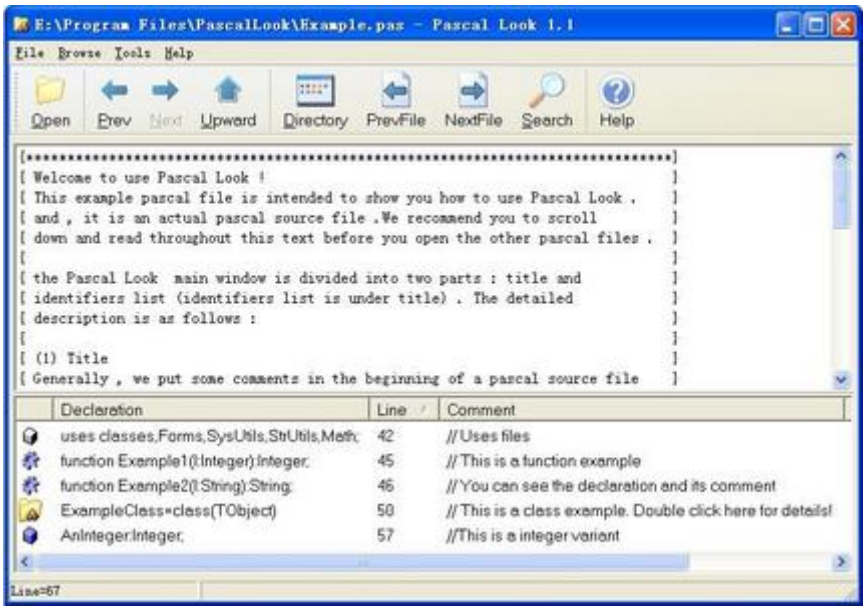
a) Example of C programming taken from http://encyclopedia.laborlawtalk.com/wiki/images/thumb/5/5c/500px-Qbasic.png) time 11.20/16.6.2006.



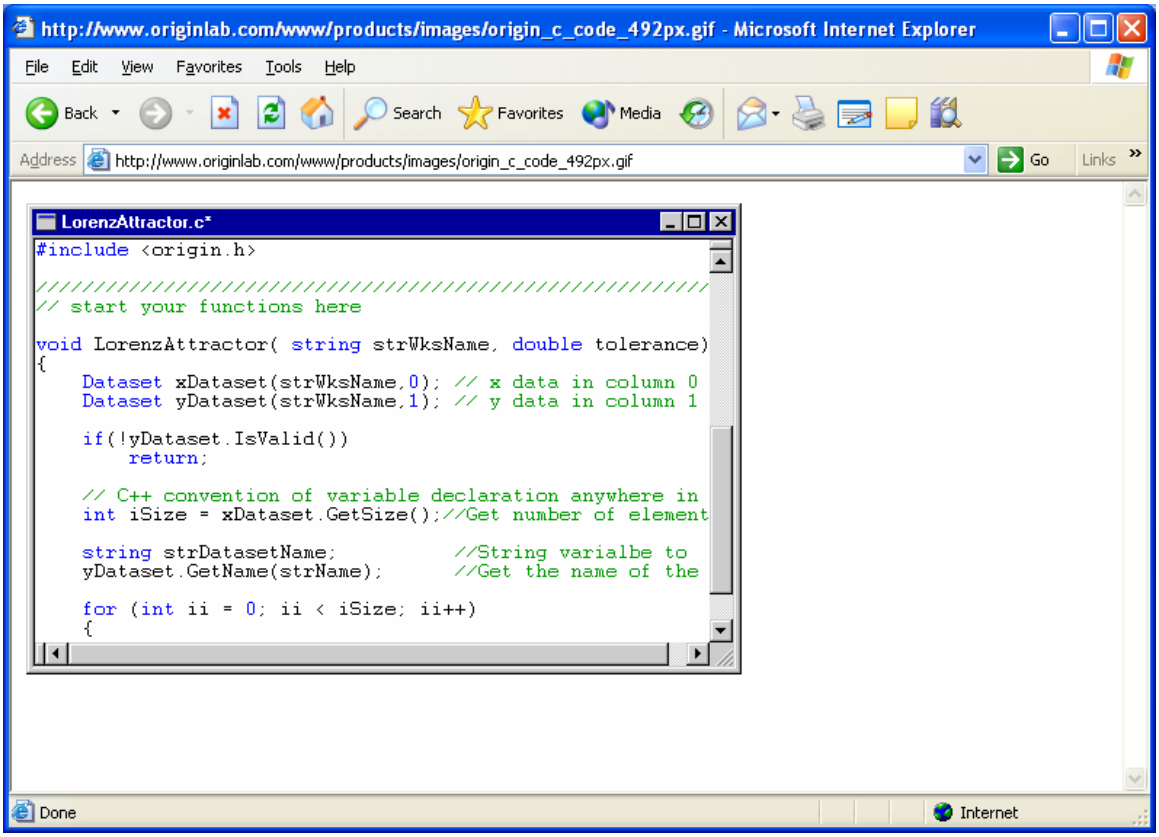
b) Example of Visual Basic taken from (www.dmst.aueb.gr/dds/etech/swdev/env.gif) 11.20/16.6.2006



c) Example of Pascal taken from (http://www.freedownloaddevelopment.com/delphi/pascal-look.html) time11.20/16.6.2006.



d) Example of C programming taken from (http://www.originlab.com/www/products/images/origin_c_code_492px.gif) 11.30/16.6.2006



e) Example of Fotran programming taken from
http://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading13.htm)
11:30/16.6.2006

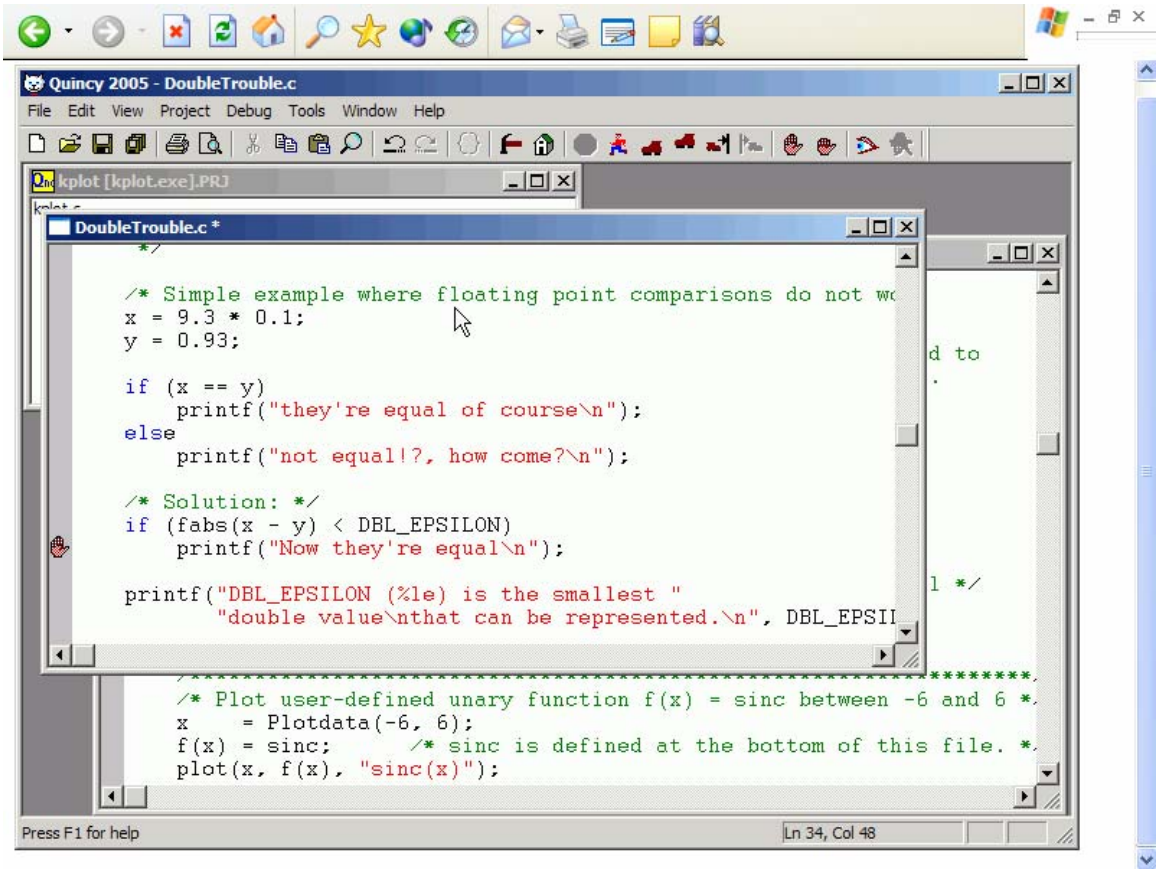
```
C       FORTRAN PROGRAM
C       AVERAGING INTEGERS ENTERED THROUGH THE KEYBOARD
        WRITE (6,10)
        SUM = 0
        COUNTER = 0
        WRITE (6,60)
        READ (5,40) NUMBER
    1   IF (NUMBER .EQ. 999) GOTO 2
        SUM = SUM + NUMBER
        COUNTER = COUNTER + 1
        WRITE (6,70)
        READ (5,40) NUMBER
        GO TO 1
    2   AVERAGE = SUM / COUNTER
        WRITE (6,80) AVERAGE
   10   FORMAT (1X, 'THIS PROGRAM WILL FIND THE AVERAGE OF ',
     +  'INTEGERS YOU ENTER ',/1X, 'THROUGH THE ',
     +   'KEYBOARD. TYPE 999 TO INDICATE END OF DATA.',/)
   40   FORMAT (I3)
   60   FORMAT (1X, 'PLEASE ENTER A NUMBER ')
   70   FORMAT (1X, 'PLEASE ENTER THE NEXT NUMBER ')
   80   FORMAT (1X, 'HE AVERAGE OF THE NUMBERS IS ',F6.2)
        STOP
        END
```

(a)

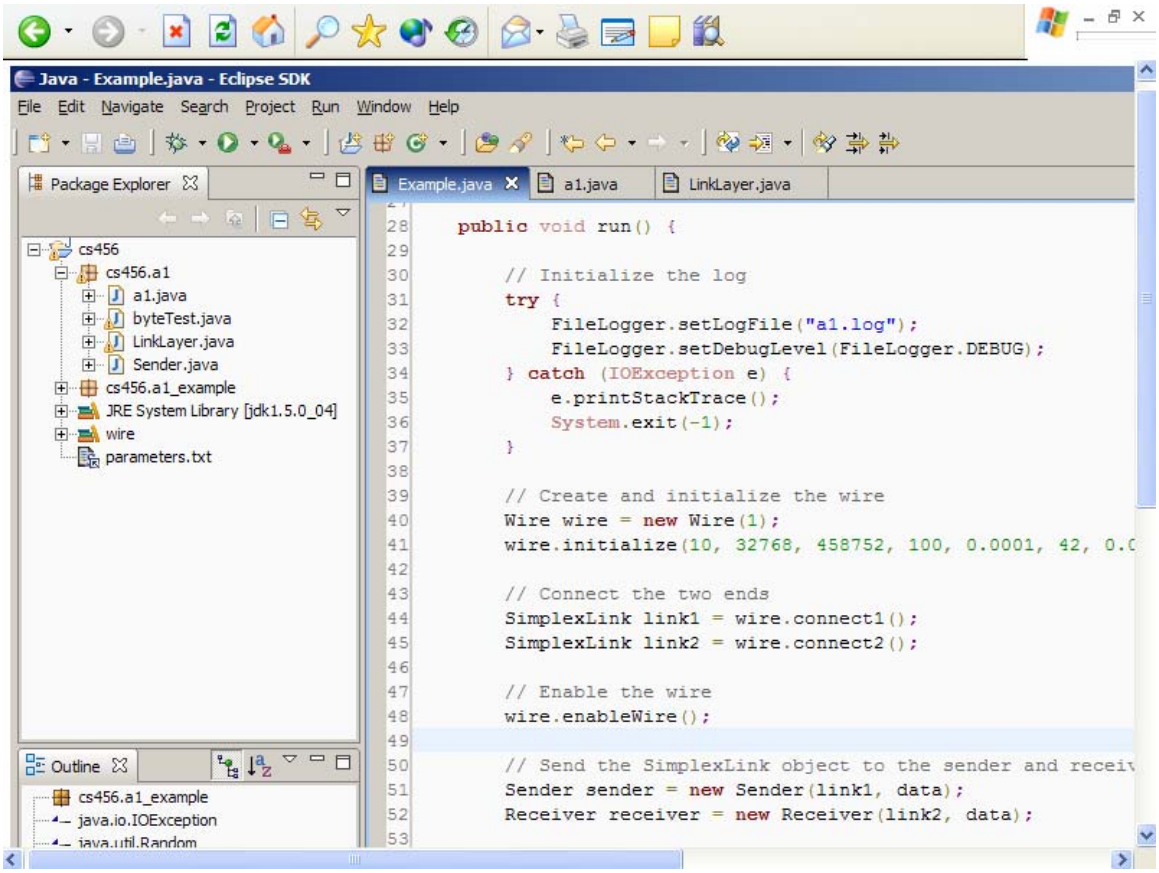
```
THIS PROGRAM WILL FIND THE AVERAGE OF INTEGERS YOU ENTER
THROUGH THE KEYBOARD.  TYPE 999 TO INDICATE END OF DATA.
PLEASE ENTER A NUMBER        6
PLEASE ENTER THE NEXT NUMBER       4
PLEASE ENTER THE NEXT NUMBER      11
PLEASE ENTER THE NEXT NUMBER     999
THE AVERAGE OF THE NUMBERS IS         7.00
```

(b)

f) Example of C++ programming taken from
(http://www.codecutter.net/tools/quincy/QscreenDump.gif)11:30/16.6.2006

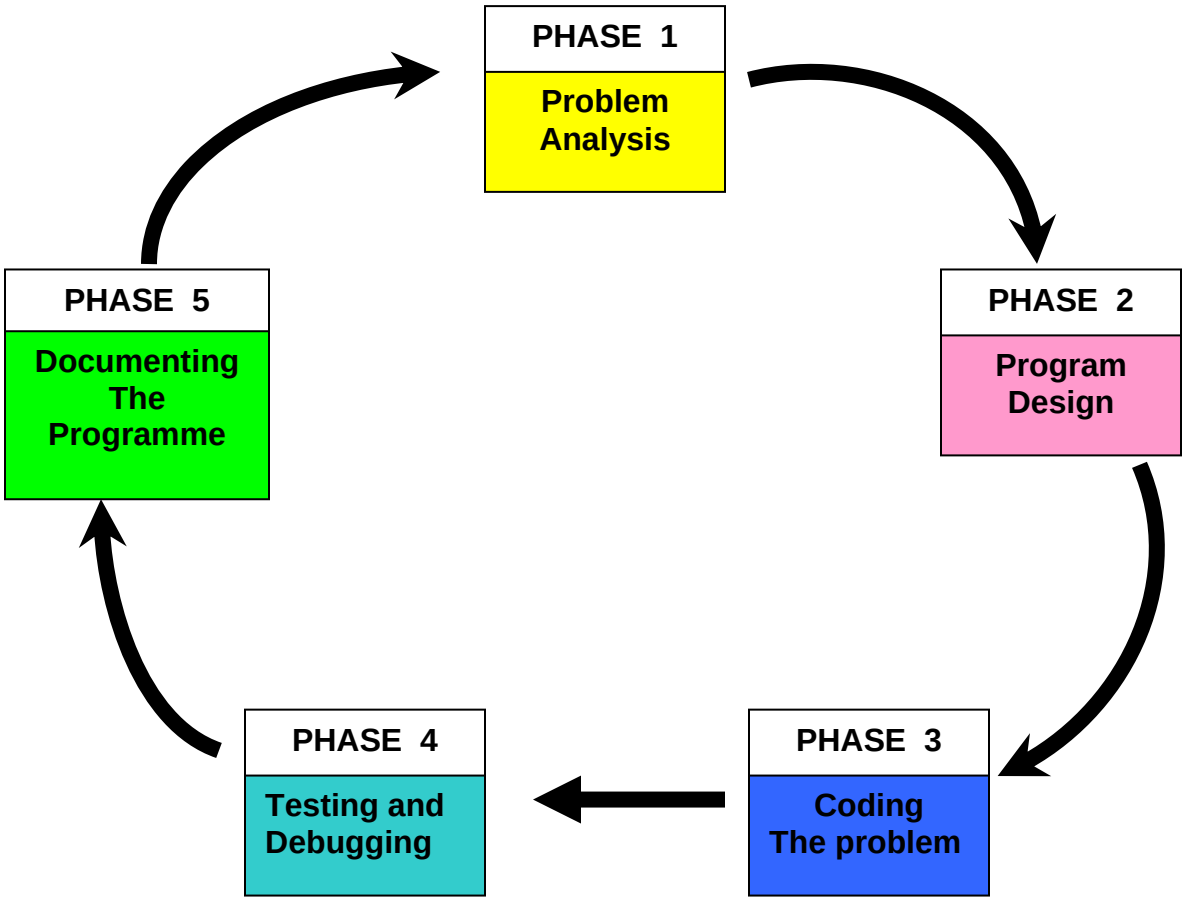


g) Example of Java programming taken from (http://ssrlibrary.ca/~megatron/images/screenshots/eclipse.png)11:30/16.6.2006



The Program Development Life Cycle

When we want to produce a program, we need to go through a few phases. There are five main phases in program development;



Let's look at the descriptions of each main phase in program development:

1. **Problem Analysis**

   What is problem analysis?

   You need to identify the problem before developing a program. Let's imagine that you are a programmer.

   As a programmer, you are contacted because your services are needed. You meet with users from the client organization to analyze the problem, or you meet with a systems analyst who outlines the project. Specifically, the task of defining the problem consists of identifying what it is you know (input-given data), and what it is you want to obtain (output-the result). Eventually, you produce a written agreement that, among other things, specifies the kind of input, processing, and output required. This is not a simple process.

2. Program Design

In this phase, you will do the following tasks:

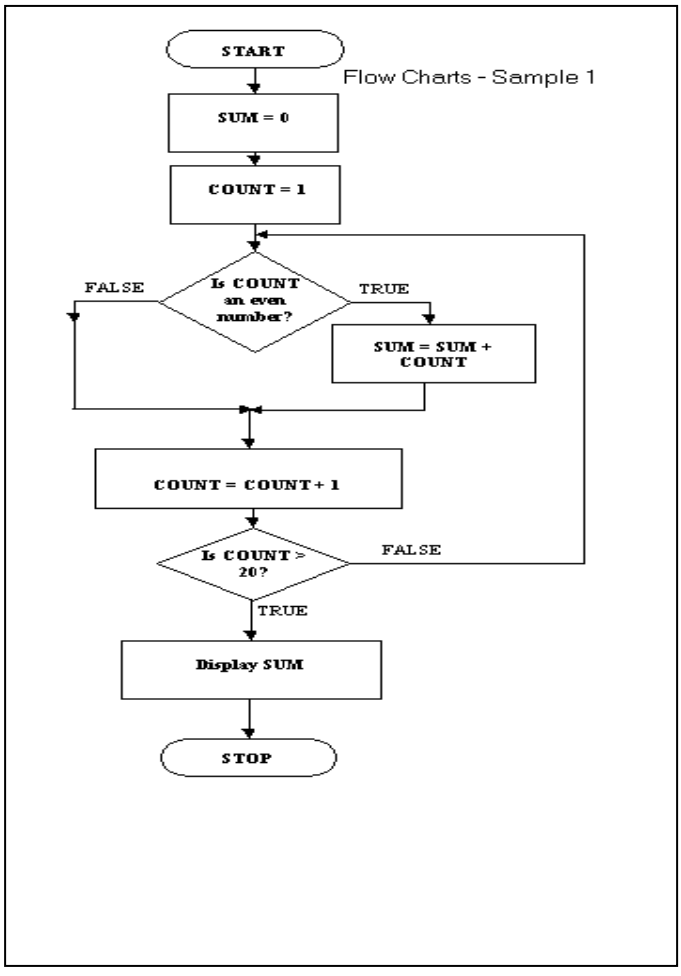
- Plan the solution to the problem (think about how will you solve the problem)
- Choose the interface (think about how will your program look like)

3. Coding

So now you are ready to write the code of the program that you have planned. You need to express your solution in a programming language.

The normal process you will go through is to translate the logic from the flowchart or pseudocode-or some other tool-to a programming language. The following are examples of both tools:

a) Example of a flow chart



b) Example of pseudo code (that reflects the flowchart shown)

```
sum = 0
count = 1
REPEAT
 IF count is even
THEN sum = sum
+ count
 count = count +
1
UNTIL count >
20
DISPLAY sum
```

As we have already noted, a programming language is a set of rules that provides a way of instructing the computer what operations to perform. There are many programming languages: BASIC, COBOL, Pascal, FORTRAN, and C are some examples. You may find yourself working with one or more of these. However in this module, we will either produce a program using Microsoft VB-express or Just BASIC v1.01.

Although programming languages operate grammatically, somewhat like the English language, they are much more precise. To get your program to work, you have to follow exactly the rules-the syntax-of the language you are using.

Of course, using the language correctly is no guarantee that your program will work, any more than speaking grammatically correct English means you know what you are talking about. The point is that correct use of the language is the required first step. Then your coded program must be keyed, probably using a terminal or personal computer, in a form the computer can understand.

One more note here: Programmers usually use a text editor, which is somewhat like a word processing program, to create a file that contains the program. However, as a beginner, you will probably want to write your program code on paper first.

4. Testing and Debugging

The fourth phase is to test and debug your program.

Why do you need to test?

The imperfections of the world are still with us, so most programmers get used to the idea that their newly written programs probably have a few errors. This is a bit discouraging at first, since programmers tend to be precise, careful, detail-oriented people who take pride in their work.

Still, there are many opportunities to introduce mistakes into programs, and you, just as those who have gone before you, will probably find several of them.

Why debug?

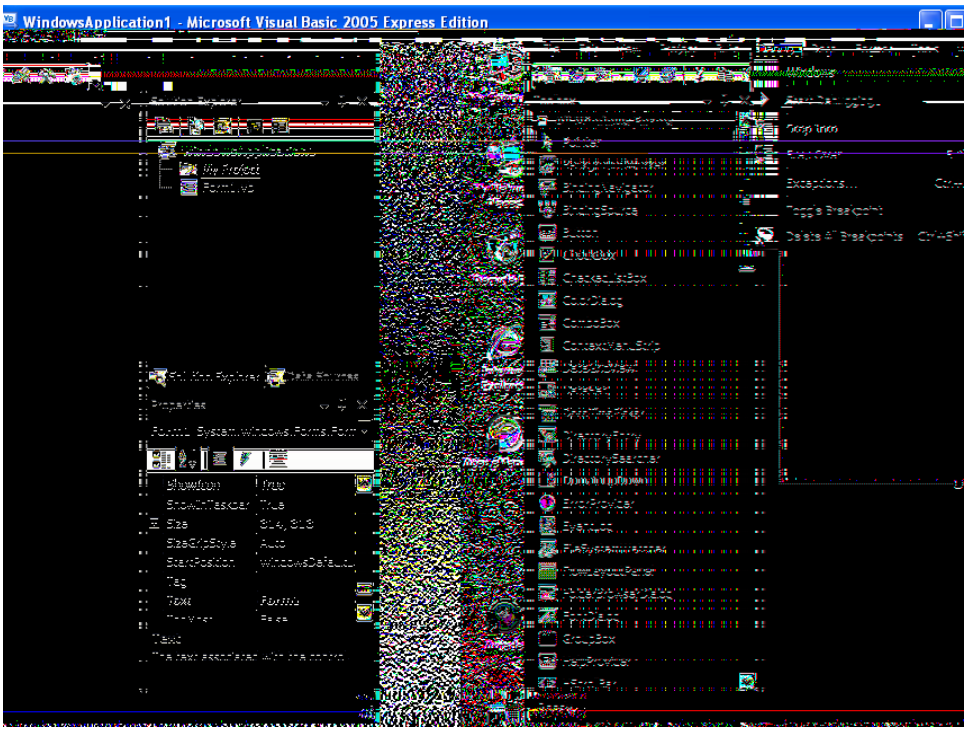
You need to debug to improve your program. Debug is a term used extensively in programming. “Debugging” means detecting, locating, and correcting bugs (mistakes), usually done by running the program.

These bugs are logic errors, such as telling a computer to repeat an operation but not telling it how to stop repeating. In this phase you run the program using test data that you devise. You must plan the test data carefully to make sure you test every part of the program.

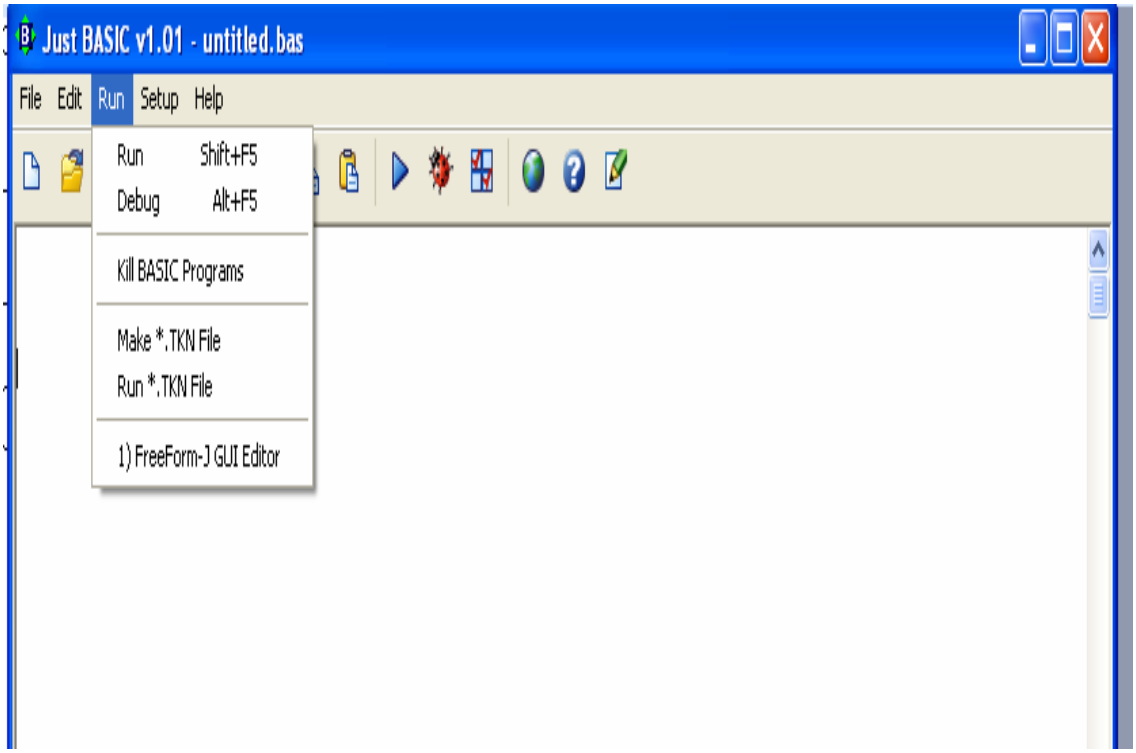
How do you debug?

You can carry out debugging by running the program. The following are examples of debugging for Visual Basic Express and Just BASIC.

a) debugging button in Visual Basic Express



b) debugging button in Just BASIC v1.01



5. Documentation

Documentation is important when programming. Documenting is an ongoing, necessary process, although, as many programmers are, you may be eager to pursue more exciting computer-centered activities.

What is documentation?

Documentation is a written detailed description of the programming cycle and specific facts about the program. Typical program documentation materials include the origin and nature of the problem, a brief narrative description of the program, logic tools such as flowcharts and pseudocode, data-record descriptions, program listings, and testing results.

Comments in the program itself are also considered an essential part of documentation. Many programmers document as they code. In a broader sense, program documentation can be part of the documentation for an entire system.

The wise programmer continues to document the program throughout its design, development, and testing. Documentation is needed to supplement human memory and to help organize program planning. Also, documentation is critical to communicate with others who have an interest in the program, especially other programmers who may be part of a programming team.

Written documentation is needed in computer industry so that those who come after you can make any necessary modifications in the program or track down any errors that you missed.