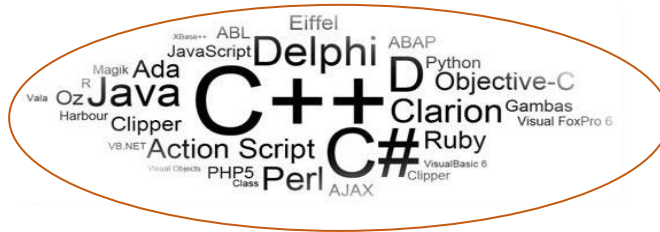

UNIT 1

Programming



Learning Outcomes:

By the end of the lesson, the students are expected to be able to use appropriate English to:

- identify and explain steps in programming.
 - identify symbols used in making a flowchart and their functions.
 - interpret a flowchart.
 - draw a flowchart and explain it.
 - identify and explain kinds of programming languages.
 - report screen messages using reported speech form.
-

1.1. Stages in Programming

Exercise 1: Look at the example of C program below. It tells the computer how to print the message “good morning”. Then, discuss with your partner what you think a programming is.

```
#include <stdio.h>
main ()
{
    Printf("good
morning\n");
}
```

Exercise 2: Match the words 1-5 with the definition (a-e).

- | | |
|-----------------|--|
| 1. flowchart | a. program instructions written in a particular computer |
| 2. source code | language |
| 3. compiler | b. the techniques of detecting and correcting errors which may |
| 4. machine code | occur in programs |
| 5. debugging | c. a diagram representing the successful logical steps of the |
| | program. |
| | d. a special program which converts the source program into |
| | machine code-the only language understood by the |
| | processor. |
| | e. the basic instructions understood by computers, consisting of |
| | 1s and 0s (binary code). |

Exercise 3: Listen to Andrea Finch, a software developer, talking to a group of students on a training course about how a program is written. You can also check the answer of exercise 2 above.

Exercise 4: Listen again and put these steps into the correct order.

- ☐ Write instructions in a programming language
- ☐ Prepare documentation
- ☐ Understand the problem and plan a solution
- ☐ Take a flowchart of the program
- ☐ Compile the program (to turn it into machine code)
- ☐ Test and debug the program

Exercise 5: Fill the missing words to complete the text. Use the words in the box.

errors program compiled debugging flowchart documentation language

Steps in Programming

To write a (1) _____ software engineers usually follow these steps. First, they try to understand the problem and define the purpose of the program. Next, they design a step-by-step plan of instructions. This usually takes the form of a (2) _____, a diagram that uses standardized symbols showing the logical relationship between the various parts of the program. These logical steps are then translated into instructions written in a high-level computer (3) _____ (PASCAL, COBOL, C++, etc.). These computer instructions are called the 'source code'. The program is then (4) _____, a process that converts the source code into machine code (binary code), the language that computers understand.

Testing program are then run to detect (5) _____ in the program. Errors are known as 'bugs', and the process of correcting these errors is called (6) _____. Engineers must find the origin of each error, then write the correct instruction, compile the program again, and conduct another series of tests. Debugging continues until the program runs smoothly.








Finally, software developers write detailed (7) _____ for the users. Manuals tell us how to use programs like word processors, databases, or web browsers.

(Taken from Infotech English for Computer Users Workbook, pp.50)

Exercise 6: Discuss and explain each step in your own words.

1.2. Flowcharting


Exercise 7: Programmers sometimes use flowchart when they are planning a program. These following symbols are used in making flowchart. Identify each and its function.



No.	Symbols	Names	Functions
1.			
2.			
3.			
4.			
5.			
6.			
7.			

Exercise 8: Read this text carefully and then do the exercises.

So far, we have dealt mainly with computers, but now it is imperative that we find out how a program is written. In all activities involving computers, it is necessary that the programmer is aware of what the machine is doing and what a program is supposed to do. As previously mentioned, flowcharting, one of the steps in programming, indicates the logical path the computer will follow in executing a program; it is a drawing very much like a road map. Flowcharting is not restricted to the preparation of programs in a particular language and should be done for each major problem before the writing of the program is attempted. If the finished program does not run as it should, the errors are more easily detected on the flowchart than in the maze of words, characters, and numbers that make up the computer program. In order to develop a flowchart successfully, a programmer should be aware of the sequence of steps needed to obtain a correct solution to a problem.


There are two ways of making a flowchart; the freehand version and the neater, more readable version. In the former version, the graphic outlines are simply jotted down as the steps of the program are worked out. This is quite satisfactory if the flowchart is not intended to be kept as a permanent record. However, if a permanent, neater and more readable flowchart is needed, the latter method whereby a template, a sheet of plastic with all the flowcharting symbols cut into it, is used.

The following symbol should be used for the purpose of uniformity. The first and last symbol is . This is the terminal symbol which indicates the beginning or the end of a program. The word 'START' must be inserted inside the figure if it is the beginning of the program and 'STOP' if it is the end of the program.

The figure in the form of a parallelogram  is used as an input/output symbol. It indicates that something is either brought to or taken from the program. The rectangular symbol  stands for processing and indicates a place in the program where action is taken. In a program, to indicate that a decision has to be made, the diamond -shaped symbol



is used. The decision is usually in the form of a question that must be answered by

either 'yes' or 'no'. Finally, the arrow  is used to show the flow or direction in which the different actions in the program are performed.

It should be noted that a flowchart is not a program, but only a step in the preparation of a program, and is used in determining how to set up and write the program. However, if the problem is not understood, neither the flowchart nor the program can be done correctly. It is possible for two programmers, working separately; to write programs to solve the same problem and come up with flowcharts and programs that may be altogether different.

After a program has been worked out, it is usually written down and kept with a copy of the flowchart along with detailed instructions for the use and interpretation of the program. This procedure is part of what is referred to as program documentation. If documentation isn't available, it is always possible to work backwards and make a flowchart from an application program. It may be necessary to create a new flowchart when the original one is missing, in order to understand the program for which it was a preparatory step.

Flowcharting is one of the first things a student programmer is taught, because a flowchart shows how a person thinks about a problem. In other words, it is through this that a new programmer reveals his or her logical and analytical ability, which is a must in programming.

Taken from English for Computer Science, pp. 197.

Decide whether these following statements are true (T) or false (F). Then make the necessary changes so that false statements become true.

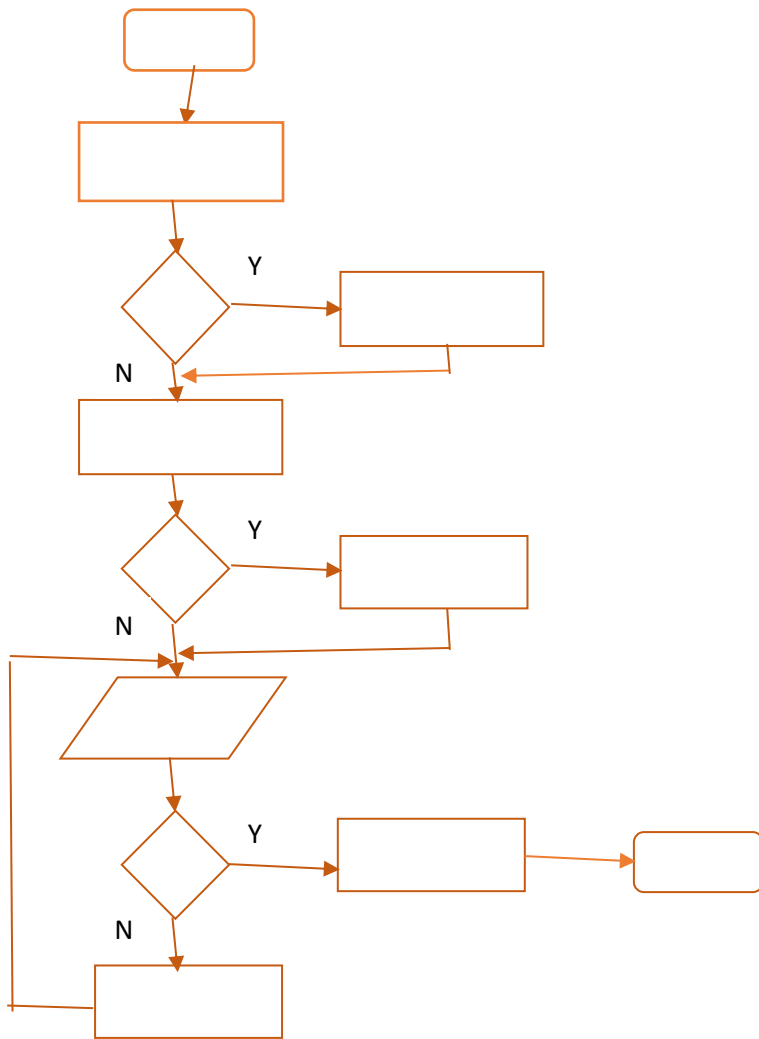
1. A good flowchart takes into account the steps which are necessary to solve the problem.
2. It is not possible to draw a flowchart without using a template.
3. There is only one possible flowchart for every problem.
4. Every programmer must learn flowcharting and realize its importance.
5. The method of flowcharting depends on the programming language being used.

-
6. Flowcharts show the logic one has to follow to solve a problem.
 7. Documenting a program is essential in explaining what the program is supposed to do.
 8. If the flowchart is correct, the program will work.
 9. Each symbol in flowcharting has a specific meaning.
 10. Flowcharts can show processes, but not decisions.

Exercise 9: Flowchart David's activities by completing the flowchart below.

David gets up in the morning, gets washed, and dressed. Before having breakfast, he checks to see if the newspaper has been delivered. If it has, he takes and puts it in the living room before sitting down to breakfast. After breakfast, he checks to make sure that he has completed all assigned homework. If there is still some to be done, he does it. Then he checks the clock, and if it is time to go, he leaves for the campus. If not, he reads the newspaper until it is time to go.

- | | |
|--|-----------------------|
| a. Read newspaper. | g. Time to go? |
| b. Take in and put newspaper in the living room. | h. Any homework? |
| c. Get up, wash, and dress. | i. Have breakfast. |
| d. Check time. | j. Complete homework. |
| e. Newspaper delivered? | k. Start. |
| f. Go to the campus. | l. Stop |



Exercise 10: Draw a flowchart for one of these activities. Then compare your completed flowchart with other students.

1. Using a payphone.
2. Buying a new computer/smartphone.
3. Preparing for an important exam.
4. Withdrawing money from ATM.
5. Re-registering a new semester.
6. Booking a ticket online.
7. Choosing a college or university.
8. Planning a vacation.

Exercise 11: Now, write the description of the flowchart you have made on Exercise 10 and present it in front of the class.

1.3. Programming Language

Exercise 12: Find 10 words about Programming.

F	H	C	G	Y	B	U	G	S	F	R
L	S	O	I	R	A	H	M	E	A	E
O	R	M	G	R	S	M	C	Z	D	P
W	L	P	X	Y	I	O	M	O	C	A
C	P	I	G	R	C	M	C	F	A	S
H	R	L	S	S	A	O	M	E	L	C
A	C	E	G	P	A	M	B	L	Z	A
R	D	R	G	R	A	Q	M	O	R	L
T	C	O	I	E	A	M	C	U	L	G
O	R	T	D	E	B	U	G	G	E	R
P	R	O	G	R	A	M	M	E	R	O

Exercise 13: Read the text carefully and answer the following questions.

Computer Languages

Unfortunately for us, computers can't understand spoken English or any other natural language. The only language they can understand directly is **machine code** which consists of 1s and 0s (binary code).

Machine code is too difficult to write. For this reason, we use symbolic language to communicate instructions to the computer. For example, **assembly languages** use abbreviations such as ADD, SUB, MPY to represent instructions. The program is then translated into machine code by a piece of software called an **assembler**. Machine code and assembly languages are called **low-level languages** because they are closer to the hardware. They are quite complex and restricted to particular machines. To make the programs easier to write and to overcome the problem of intercommunication between different types of computer, software developers designed **high-level languages**, which are closer to the English language. Here are some examples:

- **FORTRAN** was developed by IBM in 1954 and is still used for scientific and engineering applications.
- **COBOL** (Common Business Oriented Language) was developed in 1959 and is mainly used for business applications.
- **BASIC** was developed in the 1960s and was widely used in microcomputer programming because it was easy to learn. Visual BASIC is a modern version of the old BASIC language, used to build graphical elements such as buttons and windows in Window programs.
- **PASCAL** was created in 1971. It is used in universities to teach the fundamentals of programming.
- **C** was developed in the 1980s at AT&T, it is used to write system software, graphics and commercial applications. **C++** is a version of C which incorporates object – oriented programming: the programmer concentrates on particular things (a piece of text, a graphic or a table, etc.) and gives each object functions which can be altered without changing the entire program. For example, to add a new graphics format, the programmer needs to rework just the graphics object. This make programs easier to modify.
- **Java** was designed by Sun in 1995 to run on the Web Java applets provide animation and interactive features on web pages.

Programs written in high – level languages must be translated into machine code by a **compiler** or an **interpreter**. A compiler translates the source code into **object code** – that is, it converts the entire program into machine code in one go. On the other hand, an interpreter translates the source code line by line as the program is running.

It is important not to confuse **programming languages** with **markup languages**, used to create web documents. Markup languages use instructions, known as **markup tags**, to format and link text files. Some examples include:

- **HTML** which allows us to describe how information will be displayed on web pages.
- **XML** which stands for **EX**tensible **M**arkup **L**anguage. While HTML uses pre-defined tags, XML enables us to define our own tags; it is not limited by a fixed set of tags.
- **VoiceXML**, which makes Web content accessible via voice and phone. VoiceXML is used to create voice applications that run on the phone, whereas HTML is used to create visual applications (for example, web pages).

```
<xml>

<name> Andrea Finch </name>

<homework> write a paragraph describing
the C language </homework>

</xml>
```

In this XML example we have created two new tags: <name> and <homework>.

Taken from Infotech English for Computer Users (Esteras: 2011)

1. Do computers understand human languages? Why/Why not?
2. What is the function of *an assembler*?
3. How many high-level languages are mentioned? What are they?
4. Why did software developers design high-level languages?
5. What is the difference between *a compiler* and *intepreter*?
6. Why are HTML and VoiceXML called markup languages?

Exercise 14: Complete these sentences with a computer language from the text.

1. _____ allows us to create our own tags to describe our data better. We aren't constrained by a pre-defined set of tags the way we are with HTML.
2. IBM developed _____ in the 1950's. It was the first high-level language in data processing.
3. _____ applets are small programs that run automatically on web pages and let you watch animated characters, play games, etc.
4. _____ is the HTML of the voice web. Instead of using a web browser and a keyboard, you interact with a voice browser by listening to pre-recorded audio output and sending audio input through a telephone.
5. This language is widely used in the business community. For example, the statement ADD VAT to NET-PRICE could be used in a _____ program.

1.4. Grammar Study

Reporting Screen Messages

Study the examples of screen messages. Note how to report them. Pay attention to the italic words.

<i>Please enter the number.</i> <i>Type 999 to indicate end of data.</i>	<i>It requires you to enter a number.</i> <i>It tells you to type 999 to indicate the end of the data.</i>
<i>Do not attempt to log in.</i> <i>Printer is out of paper.</i>	<i>It tells you not to attempt to log on.</i> <i>It informs you that the printer is out of paper.</i>
<i>Do you want to exit?</i> <i>What is your password?</i> <i>How many copies do you want to copy?</i>	<i>It asks you if/whether you want to exit.</i> <i>It asks you what your password is.</i> <i>It asks you how many copies you want to copy.</i>

Exercise 15: Report each of these screen messages.

1. Make sure the printer is switched on before continuing.
2. Game mode is on.
3. Do you want to create a new document?
4. What is the captcha code?
5. Fill in your name in the box.
6. Please type the next number.
7. Enter your password.
8. Please choose from menu below.
9. Can't rename "Pictures" because a folder with that name already exists.
10. Exit?
11. Are you sure you want to copy the selected files?
12. Do you want to defrag the drive?
13. Mute story and posts?
14. If you unfollow this account, you'll have to request to follow again.
15. Click the subscribe button to follow us.