Program development

PROGRAM DEVELOPMENT.

Stages involved in the program development cycle.

The process of program development can be broken down into the following stages:

1. Problem recognition (Identification of the problem).
2. Problem definition.
3. Program design.
4. Program coding.
5. Program testing & debugging.
6. Program Implementation and maintenance.
7. Program documentation.

Problem recognition.

Problem recognition refers to the understanding and interpretation of a particular problem.

The programmer must know what problem he/she is trying to solve. He/she must also understand clearly the nature of the problem & the function of the program.

In order to understand a problem, look for the keywords such as compute, evaluate, compare, etc.

Usually, a programmer identifies problems in the environment and tries to solve them by writing a computer program.

There are 3 situations that cause the programmer to identify a problem that is worth solving:

1. Problems or undesirable situations that prevent an individual or organizations from achieving their purpose.
2. Opportunity to improve the current program.
3. A new directive given by the management requiring a change in the current system.

Sample problem: Develop a program that can be used to calculate/find the area of a circle. Use the equation $A = \pi * r^2$.

Problem definition (Problem Analysis).

In Problem definition, the programmer tries to define (determine) the:

1. Output expected from the program.
2. Inputs needed to generate the output information.
3. Processing activities (requirements), and
4. Kind of files which may be needed.

The programmer should write a narrative on what the program will do, and how it is meant to achieve the intended purpose. Within this narrative, he/she is required to determine what data is to be input & what information is to be output.

For example:

In calculating the area of any circle, the parameters needed to determine the area of any circle are:

1. **Input:**
    1. Pie (π) which is a constant.
    2. The radius of the circle.
2. **Process:** The formula for calculating area of a circle, which is π * radius * radius.
3. **Output:** The area of the circle (A).

At the end of the problem definition, the programmer is required to write a requirements report/document for the new program. This document will enable the programmer to come up with a program design that meets the needs at hand. Note. Problem definition should be done thoroughly to ensure user satisfaction, and to facilitate the subsequent stages in the program development cycle. A failure at this stage usually results in a system that will not work as intended, or that may not work at all.

Program design

Program design is the actual development of the program's process or problem solving logic called the Algorithm.

It involves identifying the processing tasks required to be carried out in order to solve the problem.

The design stage enables the programmer to come up with a model of the expected program (or a general framework (outline) of how to solve the problem, and where possible, break it into a sequence of small & simple steps.
The models show the flow of events throughout the entire program from the time data is input to the time the program gives out the expected information.

- The processing tasks must be in order & systematic. Therefore, the programmer identifies the processing tasks required, and the exact order in which they are to be carried out.

- The design process does not take account of the programming language to be used in the final product, since it only defines program logic.
- Program design provides for easy maintenance.

Note. It is important to design programs before entering them into the computer. The programmer should only attempt to covert a design into a program code after ensuring that it is logically correct. If possible, check the logical order on the desk.

Some programmers produce rough & ready solutions at a Keyboard, and continue to amend the programs until eventually the program appears to do what was expected. This is not recommended in programming because of the following reasons:

1. The final code may not be easy to follow, since it was just cobbled together.
2. Variable names & specific items of code may not be documented.
3. Programs produced by continuous amendments & changing of codes mostly lead to unforeseen side effects.
   E.g., there may not have been plan for testing the program or procedures, hence, the program may easily fail.
4. A programmer may be asked to modify the code at a later date. Without sufficient documentation, the programmer will be forced to trace through the program in order to gain an insight into how the program functions.

Modular programming

Many programs are non-monolithic (i.e., they are not usually made up of one large block of code). Instead, they are made up of several units called modules, that work together to form the whole program with each module performing a specific task. This approach makes a program flexible, easier to read, and carry out error correction.

Program coding

Program coding is the actual process of converting a design model into its equivalent program.

Coding requires the programmer to convert the design specification (algorithm) into actual computer instructions using a particular programming language.

For example;

The programmer may be required to write the program code either in Pascal, C++, Visual Basic or Java, and develop (invent) suitable identifiers, variable names, & their data types. However, remember that, at this stage the coding is still a Pencil & paper exercise.

The end result of this stage is a source program that can be translated into machine readable form for the computer to execute and solve the target problem.

Rules followed in coding a program.

1. Use the standard identifiers or reserved words.
2. Make the program more readable by using meaningful identifiers.
3. Don’t use similar variables.
4. Keep spellings as normal as possible.
5. Use comments to explain variables & procedures. This makes the program readable.

6. Avoid tricks – write the program using straightforward codes that people can readily understand.
7. Modularize your program.

Sample programs written in Pascal language.

Example 1:

Develop a program code that would be used to solve the equation of a straight line given by the expression: Y = mx + c

```
Program StraighLine (input, output);
VAR
y, m, x, c: INTEGER;

BEGIN
   Writeln (‘Input the value of M’);
Readln (M);

   Writeln (‘Input the value of X’);
Readln (X);

   Writeln (‘Input the value of C’);
Readln (C);

Y: = (m * x) +c;
   Writeln (‘The value of y is:’, Y);
END.
```

Explanation of the Pascal source code above

Program StraightLine (input, output); This is the program Header.
The word **“Program”** indicates the beginning of the program whose name is StraightLine.
The (input, output) statements shows that the program expects some input from the Keyboard and display the output on the Screen.
VAR is short form for Variable. A variable is a location for data in the computer memory.
This statement tells the computer that variables are about to be declared. When a variable is declared, the computer sets aside some memory space to store a value in the variable.
y, m, x, c: INTEGER; Four variables of type Integer have been declared. This means that, the memory spaces that will be set aside can only hold values that are whole numbers.
BEGIN The Begin statement marks the start of the program body. Statements in this section are executed by the computer. E.g., execution starts by asking the user to input the value of m.
Writeln (‘Input the value of M’); The Writeln statement displays whatever is between the inverted commas in the brackets. The statements will be sent to the screen exactly the way they appear in the brackets. This is because; the inverted commas are meant to make the output readable on the screen.
To display the value held in a variable on the screen, remove the inverted commas and write the name of the variable in the brackets, e.g., Writeln (y) will display the value held in the variable y.
Readln (M); The Read or Readln statement reads a value and stores it in a variable.

When the program is running, a Read/Readln statement in the code will displays blinking cursor that indicates to the user where to type the input.
Y: = (m * x) +c; Calculates the value of y. in Pascal, the symbol ': =' is called the Assignment statement.
The values on the right are calculated then the answer stored in the variable y which is on the left of the assignment symbol.
Writeln ('The value of y is:', Y); The Writeln displays the value held in the variable y on the screen.
Note. Y is not within the inverted commas.
END. The 'END.' statement shows the end of a program.

Example 2:

```
Program AreaCircle (input, output);
CONST
Pi = 3.142;

VAR
Radius, Area: REAL;

BEGIN
Writeln ('Enter the radius');

Readln (Radius);

Area: = Pi * Radius * Radius;
Writeln ('The Area is', Area);

END.
```

Explanation of Pascal source code

Program AreaCircle (input, output); The Header of the program.
The statements in ( ) shows that the user inputs data via Keyboard and the program display information on the Screen.
CONST
Pi = 3.142; A constant has been declared with a name Pi and value 3.142.
VAR
Radius, Area: REAL; Variables with fractional parts have been declared.
BEGIN Marks the beginning of the program body.
Writeln ('Enter the radius'); Displays on the screen the string between the inverted commas.
Readln (Radius); Displays a blinking cursor that tells the user that an input is needed before the program can continue.
Area: = Pi * Radius * Radius; Calculates the Area. An assignment statement (: =) has been used.
Writeln ('The Area is', Area); Displays the value stored in the variable Area.
END. Marks the end of the program.
Revision Questions.

1. State the rules followed in coding a program.

Program Testing and Debugging

After designing & coding, the program has to be tested to verify that it is correct, and any errors detected removed (debugged).

TESTING:

Testing is the process of running computer software to detect/find any errors (or bugs) in the program that might have gone unnoticed.

During program testing, the following details should be checked;

1. The reports generated by the system.
2. The files maintained in connection to the system's information requirements.
3. The input to the system.
4. The processing tasks.
5. The controls incorporated within the system.

Note. The testing process is a continuous process, and it ends only when the Programmer & the other personnel involved are satisfied that when operational, the program will meet the objectives and the growing demands of the organization.

Types of program errors

There are 5 main types of errors that can be encountered when testing a program. These are:

1. Syntax errors.
2. Run-time (Execution) errors.
3. Logical (arithmetic) errors.
4. Semantic errors.
5. Lexicon errors.

Syntax errors

Every programming language has a well-defined set of rules concerning formal spellings, punctuations, naming of variables, etc. The instructions are accepted only in a specified form & and must be obeyed by the programmer.

Syntax errors are therefore, programming errors/mistakes that occur if the grammatical rules of a particular language are not used correctly.

Examples:

1. Punctuation mistakes, i.e., if the programmer does not use the right punctuations & spaces needed by the translator program, e.g., omitting a comma or a semicolon.
2. Improper naming of variables.
3. Wrong spellings of user defined and reserved words.

Reserved words are those words that have a special meaning to the programming language, and should not be used by the programmer for anything else.

Syntax errors are committed by the programmer when developing, or transcribing the program, and can be detected by the language translators, such as the Compiler as it attempts to translate a program. Such errors must be corrected by the programmer before the program runs.

Logical (arithmetic) errors.

These are errors in the program logic.

Logical errors relate to the logic of processing followed in the program to get the desired results. E.g., they may occur as a result of misuse of logical operators.

Logical errors cannot be detected by the translator. The programmer will detect them when the program results are produced.

The program will run, but give the wrong output or stop during execution.

Run-time (Execution) errors.

These errors occur during program execution.

Run-time (execution) errors occur when the programmer introduces new features in the program, which are not part of the translator's standards.

For example; they may occur if:

1. The computer is asked to divide a number by zero.
2. The number generated as a result of an instruction is too large to fit in a memory location.
3. When you raise a number to a very big power that cannot be accommodated in the Register's structure of the computer.
4. In case of a closed loop in the program, leading to a set of instructions being executed repetitively for a long time.

Execution errors are not detected by the translator programs, but are detected by the computer during execution. Sometimes, execution errors may lead to premature end of a program.

To detect and eliminate Execution errors, a test run should be performed on the program after it has been translated.

Semantic errors.

These are meaning errors. They occur when the programmer develops statements, which are not projecting towards the desired goal. Such statements will create deviations from the desired objectives.

Semantic errors are not detected by the computer. The programmer detects them when the program results are produced.

Example;

```
a).    IF GP>=1500 OR 2200 THEN
TAX: = GP - (GP * 13%)

b).    IF GP>=1500 AND GP<= 2200 THEN
TAX: = GP - (GP * 13%)
```

In the 1st statement, if the selection is between 1500 & 2200, the computer will pick only 1500 & 2200, and the other values will not be touched.

In the 2nd statement, the computer will be able to pick all the values between 1500 & 2200 because of the 'AND' operator.

Lexicon errors.

These are the errors, which occur as a result of misusing Reserved words (words reserved for a particular language).

Revision Questions.

1. State the three types of errors that can be experienced in program testing, and how each can be detected.
2. Syntax errors can be detected by the help of translators while logical errors are detected differently. Explain FIVE methods which can be used to detect Logical errors.

DEBUGGING:

The term Bug is used to refer to an error in a computer program.

Most programming errors often remain undetected until an attempt is made to translate a program.

The most common errors include:-

- Improperly declared Constants and Variables.
- A reference to undeclared variable.
- Incorrect punctuation.

Debugging is therefore, the process of detecting, locating & correcting (removing, eliminating) all errors (mistakes or bugs) that may exist in a computer program.

TYPES OF TESTING (Methods of error detection)

For the program to be assumed as correct, several testing needs to be conducted by the programmer to ascertain/establish their validity.

There are several methods of testing a program for errors. These include:

1. Dry running (Desk checking).
2. Translator system checking.
3. Functional testing.
4. Use of Test data.
5. Use of debugging utilities.
6. Diagnostic procedures.
7. System test with actual data.

Dry Running (Desk checking):

Dry running is a method of checking a program for errors by making the corrections on a paper before entering it in the program editor.

It involves going through the program while still on paper verifying & validating its possible results. If the final results agree with the original test data used, the programmer can then type the program into the computer and translate it.

- Dry running helps the programmer to identify the program instructions, detect the most obvious syntax and logical errors, & the possible output.
- Dry running is much faster. This is because; it involves the use of human brain as the processor, which has got a well inbuilt common sense.

Translator system checking:

This is a type of testing, which involves the computer & the translator programs.

After entering the program, it is checked using a translator to detect any syntax errors. The translator can be a Compiler or an Interpreter, which goes through the set of instructions & produces a list of errors, or a program/statement listing which is free from errors.

Functional testing (White-box testing):

This type of testing is based upon examining the internal structure of a program & selecting test data, which give rise to the alternative cases of control flow.

Use of Test data.

The accuracy of a program can be tested by inputting a set of values referred to as Test data. The test data is designed to produce predictable output. There are 2 types of test data;

1. **Real data (live data):** - test data obtained from the real problem environment (practical applications).
2. **Dummy data: -** assumed test data.

The programmer invents simple test data, which he/she uses to carry out trial runs of the new program. At each run, the programmer enters various data variations including data with errors to test how the system will behave. For example, if the input required is of numeric type, the programmer may enter alphabetic characters. The programmer will then compare the output produced with the predicted (actual) output.

Notes.

- Where possible, the program should be tested using the same test data that was used for desk checking. More strict/rigid tests should be applied on the program in order to test the program to its limits.
- Only Logical errors & Semantic errors can be corrected by the programmer using test data.
- A good program should not crash due to incorrect data entry but should inform the user about the irregularity and request for the correct data to be entered.

Use of debugging utilities.

After the program has been entered in the program editor, debugging utilities which are built in the computer can be run during translation to detect any syntax errors in the program.
The errors are corrected and the debugging process is repeated again to find out more errors, before the program is executed.

Diagnostic procedures.

For complex programs, diagnostic procedures, such as Trace routines, may be used to find logical errors.
A Trace prints out the results at each processing step to enable errors to be detected quickly.

System Test with actual data.

This is whereby the new program is run in parallel with the existing system for a short time so that results can be compared and adjustments made. In such cases, the system test is made using actual data.

Review Questions.

1. Differentiate between Testing and Debugging.
2. What is Dry running?

Implementation and Maintenance.

IMPLEMENTATION

Implementation refers to the actual delivery, installation and putting of the new program into use.

The program is put into use after it is fully tested, well documented, and after training the staff who will be involved in the running of the new program.

Structured Walk Through:

It is an organized style of evaluating/reviewing a program by a team of other programmers, which then reports to the programming team.

REVIEW AND MAINTENANCE.

Once the program becomes operational, it should be maintained throughout its life, i.e., new routines should be added, obsolete routines removed, & the existing routines adjusted so that the program may adapt to enhanced functional environments.

The main objective of maintenance is to keep the system functioning at an acceptable level.
Program maintenance mainly involves: -

- Correcting errors that may be encountered after the program has been implemented or exposed to extensive use.
- Changing procedures.
- Hardware and software maintenance.
- Changing parameters and algorithms used to develop the original programs.
- Making any adjustments as new technology comes.

Note. Program maintenance runs parallel to the maintenance of the program documentation, i.e., any time maintenance is carried out on the program, the documentation should also be updated to convey the right image of the system.

Program documentation.

After writing, testing, and debugging a program, it must be documented. In other words, the programmer should describe all what he was doing during the program development stages.

Program documentation is the writing of supportive materials explaining how the program can be used by users, installed by operators, or modified by other programmers.

Note. All the program development activities (i.e., from the initial stage up to the complete program) should be documented/recorded in order to assist in the development of the program, future modification of the program, general maintenance, machine & software conversion at a later date, and program changeover.

Documentation can either be; Internal or External.

Internal documentation is the writing of non-executable lines (comments) in the source program that help other programmers to understand the code statements.

External documentation refers to reference materials such as user manuals printed as booklets.

Types of program documentation.

There are 3 target groups for any type of documentation:

1. User-oriented documentation.
   This enables the user to learn how to use the program as quickly as possible, and with little help from the program developer.
2. Operator-oriented documentation:
   This is meant for computer operators such as the technical staff. It is used to help them install & maintain the program.
3. Programmer-oriented documentation:
   This is a detailed documentation written for skilled programmers. It provides the necessary technical information to help in future modification of the program.

Some documents used in program documentation.

1. User guide/ manual.
   This is a manual provided for an end-user to enable him/her use or operate the program with minimal or no guidance.
   A User guide is used in user-oriented documentation.
2. Reference guide.
   It is used by someone who already knows how to use the program but needs to be reminded about a particular point or obtain more detailed information about a particular feature.
3. Quick Reference guide.
   This could be a single sheet or card small enough to fit into a pocket. It is used by the user to get help for the common tasks carried out within the program.
4. Technical manuals.
   They are intended for System analysts & Programmers. They assist in maintaining & modifying the program design and code.

Contents in a program document.

Documentation includes:

1. Title of the program.
2. Function of the program.
3. Language used.
4. Hardware & Software required to support the processing of the system.
5. File specifications (details of the data structures used, & details of how data files are to be organized, accessed, and kept secure).
6. Limitations of the program.
7. Format of the input & the output expected.
8. Design of the program using the design tools (i.e., detailed algorithms & procedures used).
9. A listing of the Source program and the program flowcharts.
10. A carefully devised set of Test data, and a table of expected results.
11. Detailed instructions on how to run the program.

Review Questions.

1. What is program designing?
2. (a). Define program documentation.
   (b). What does a program documentation contain?

3. Briefly explain how each of the following documents are useful in programming?
   (a). User manual / guide.
   (b). Reference guide.
   (c). Quick reference guide.
4. Program documentation is different from Implementation. Explain.
5. Outline and briefly explain the stages involved in program development.