

## **CHAPTER 6: COMPUTER PROGRAM DEVELOPMENT**

**Unit of learning code:** ICT/CU/IT/CR/5/6

**Related Unit of Competency in Occupational Standard:** Develop A Computer Program

### **6.1 Introduction to the unit of learning**

This unit specifies competencies required to develop computer program. It involves identifying of programming concepts and approaches, identifying program development methodologies, identifying program design, identifying of programming languages, performing of basic structured programming, and performing basic internet programming.

### **6.2 Summary of Learning Outcomes**

1. Identify Programming concepts and approaches.
2. Identify program development methodologies.
3. Identify Program design
4. Identify computer programming languages.
5. Perform Basic structured Programming using C language.
6. Perform Basic Internet programming

#### **6.2.1 Learning Outcome 1: Identify Programming concepts and approaches.**

##### **6.2.1.1 Introduction to the Learning Outcome**

This learning outcome specifies the content of competencies required to develop a computer program. It entails the definition of various programming concepts i.e., program and programming, types of a Language translator, and types of programming approaches.

##### **6.2.1.2 Performance Standard**

- 6.2.1.2.1** Identification of program and programming is done.
- 6.2.1.2.2** Language translators are identified.
- 6.2.1.2.3** Description of programming approaches is done.

### 6.2.1.3 Information Sheet

#### Programming concepts

**Program**-A computer program is a set of coded instructions given to the computer and represents a logical solution to a problem. It directs a computer in performing various operations/tasks on the data supplied to it.

Computer programs may be written by the hardware manufacturers, Software houses, or a programmer to solve user problems on the computer.

**Programming** is the process of designing a set of instructions (computer programs) that can be used to perform a particular task or solve a specific problem.

It involves the use of special characters, signs, and symbols found in a particular programming language to create computer instructions.

The programming process is quite extensive. It includes analyzing an application, designing a solution, coding for the processor, testing to produce an operating program, and developing other procedures to make the system function.

The program created must specify in detail the logical steps to be taken & the method of processing the data input into the computer to carry out the specified task.

A computer program performs the following:

- Accepts data from outside the computer as its input.
- Carries out a set of processes on the data within the computer memory.
- Presents the results of this processing as its output, and
- Stores the data for future use.

**Source program (source code)**-Refers to the program statements that the programmer enters in the program editor window, and which have not yet been translated into machine-readable form. Source code is the code understood by the programmer and is usually written in a high-level language or Assembly language.

**Object code (object program)**- Refers to the program code that is in machine-readable (binary) form.

**Object code** is the code the computer can understand and is produced by a *Compiler* or *Assembler* after translating the Source program into a form that can be readily loaded into the computer.

## Language translators

A computer uses & stores information in binary form, and therefore, it cannot understand programs written in either high-level or assembly languages. This means that any program code written in Assembly language or high-level language must be translated into Machine language before the computer can recognize & run these programs.

A **Translator** is special system software used to convert the **Source codes** (program statements written in any of the computer programming languages) to their **Object codes** (computer language equivalents).

The Translators reside in the main memory of the computer, and use the program code of the high-level or Assembly language as input data, change the codes, and gives the output program in machine-readable code.

Also, translators check for & identify some types of errors that may be present in the program being translated, e.g., Syntax/grammatical errors. They will produce error messages if there is a mistake in the code.

### Types of language translators

#### i. Assembler

An *Assembler* translates programs written in Assembly language into machine language that the computer can understand and execute.

#### **Functions of an Assembler.**

- It checks whether the instructions are written are valid, and identifies any errors in the program.

The Assembler will display these errors as well as the complete source and object programs. If the program has no errors, the job control will let it run immediately, or save the object program so that it may run it later without translating it again.

- It assigns memory locations to the names the programmer uses.

E.g., the Assembler keeps a table of these names so that if an instruction refers to it, the Assembler can easily tell the location to which it was assigned.

- It generates the machine code equivalent to the Assembly instructions.

Usually, the Assembler generates a machine code only when no errors are detected. Some of the errors include;

- Typing mistakes.
- Using the wrong format for instruction.
- Specifying a memory location outside the range 0 – 2047.

## ii. **Interpreter**

- An interpreter is a common kind of language processor. Instead of producing the target program as a translation, an interpreter appears to directly execute the operations specified in the source program on inputs supplied by the user.
- In contrast, an interpreter reads a statement from the input, converts it to an intermediate code, executes it, then takes the next statement in sequence.
- If an error occurs, an interpreter stops execution and reports it.

## iii. **Compiler**

A *compiler* translates the entire/whole source program into object code at once and then executes it in machine language code. These machine code instructions can then be run on the computer to perform the particular task as specified in the high-level program.



- It checks for **Syntax errors** in a program (i.e., statements which do not conform to the grammatical rules of the language). If there are no syntax errors, it generates machine code equivalent to the given program.
  - It combines the program (machine) code generated with the appropriate subroutines from the library.
  - It produces a listing of the program, indicating errors if any.

#### **i. Linker**

The linker is a computer program that links and merges various object files to make an executable file. All these files might have been compiled by the separate assembler.

The major task of a linker is to search and locate referenced module/routines in a program and to determine the memory location where these codes will be loaded making the program instruction have an absolute reference.

#### **ii. Loader**

The loader is a part of the operating system and is responsible for loading executable files into memory and execute them.

It calculates the size of a program (instructions and data) and creates memory space for it. It initializes various registers to initiate execution.

### **Programming approaches**

- Procedural
- Event-driven
- Object-oriented
- Internet-based

#### **i. Procedural Programming**

Problem is broken down into procedures, or blocks of code that perform one task each. All procedures taken together form the whole program. It is suitable only for small programs that have a low level of complexity.

**Example** – For a calculator program that does addition, subtraction, multiplication, division, square root and comparison, each of these operations can be developed as separate procedures. In the main program, each procedure would be invoked based on the user’s choice.

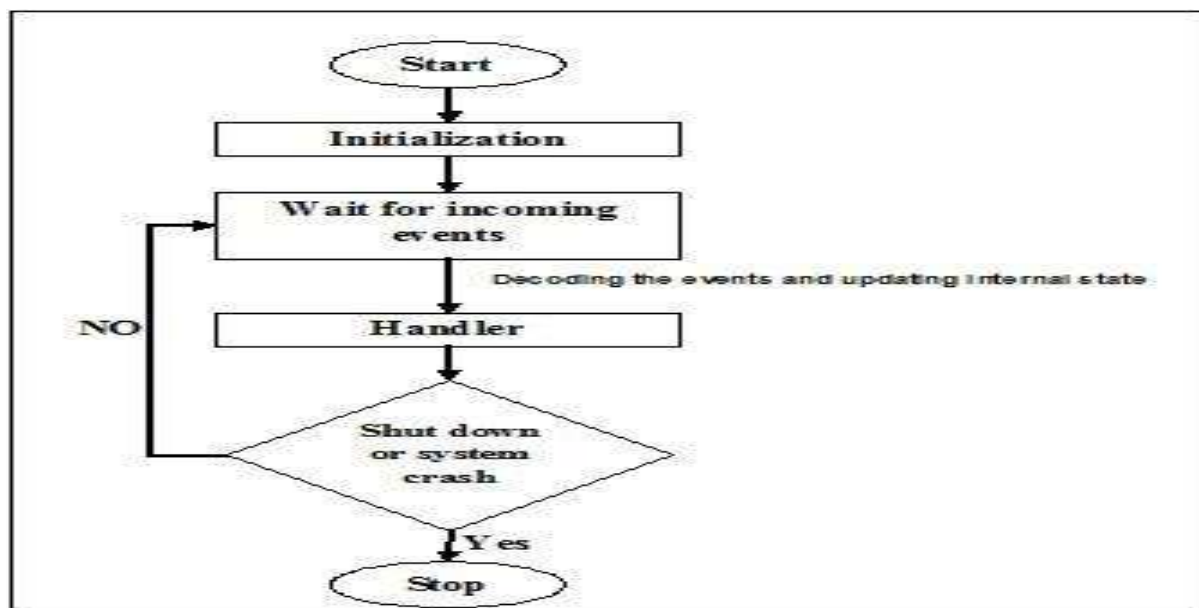
ii. Object-oriented Programming

Here the solution revolves around entities or objects that are part of the problem. The solution deals with how to store data related to the entities, how the entities behave and how they interact with each other to give a cohesive solution.

**Example** – If we have to develop a payroll management system, we will have entities like employees, salary structure, leave rules, etc. around which the solution must be built.

i. *Event-Driven programming*

Event-driven programming depends upon an event loop that is always listening for the new incoming events. The working of event-driven programming is dependent upon events. Once event loops, then events decide what to execute and in what order. –



## Figure 177 Event-driven programming

### *ii. Internet-based programming*

The internet-based programming approach is also called web programming. web programming is a set of web site development activities in WWW (World Wide Web) based on the public network Internet. In the simplest case, each web site includes static web pages, but usually is an aggregation of static and dynamic web pages. The later ones link under certain rules data sources (structured and non-structured DB) and program codes (scripts), realizing some functionality. This combination of web pages, scripts, and DB is called a web application. The web applications are distributed (on one or more servers), working under HTTP, and available by a browser or non-browser client application serving as a network interface. In detail, web programming refers to writing, markup, and coding, involved in Web site building (known as web development), which includes web content, web client and server scripting, and network security.

### **6.2.1.4 Learning Activities**

#### **Practical exercise**

ABC organization has recently employed you as a program developer expert. In preparation to develop a program to run some operations in the company, the manager in consultation with other ICT experts has requested you to lead a team in coming up with a detailed presentation on some of the existing programming approaches that can be utilized.

Special instructions;

- a. Using your computer, make a detailed PowerPoint presentation on the following programming approaches
  - i. Procedural
  - ii. Event-driven
  - iii. Object-oriented
  - iv. Internet-based
- b. Burn your presentation in a CD to be submitted to when required.



### 6.2.1.5 Self-Assessment

1. What is the meaning of the following terms:
  - i. Computer program.
  - ii. Programming.
  - iii. Programming language.
2. Regarding programming, distinguish between Source program and Object code.
3. What are the function(s) of Assemblers, Interpreters, and Compilers in a computer system?
4. What is the difference between a Compiler and an Interpreter?

### 6.2.1.6 Tools, Equipment, Supplies, and Materials

- Flow charts
- Data flow diagram
- Decision table
- Decision tree
- Web Authoring tools
- Notepad
- Computer
- Software
- Digital instructional material including DVDs and CDs

### 6.2.1.7 References

Blignaut, A. S., Hinostroza, J. E., Els, C. J., & Brun, M. (2010). ICT in education policy and practice in developing countries: South Africa and Chile compared through SITES 2006. *Computers & Education*, 55(4), 1552–1563.

*Computer Hardware*. (n.d.). Retrieved September 30, 2020, from <https://web.stanford.edu/class/cs101/hardware->

KIRIHATA, Y. (2009). *Information processing apparatus and method, computer-readable recording medium, and an external storage medium* (United States Patent No. US20090241114A1).

Lehmann, S., & Schweitzer, B. (2006). *Apparatus, system, and method for creating customized workflow documentation* (United States Patent No. US20060059423A1).

Nagar, T. (2019, December 3). *What is software and types of software with examples?* YourStory.Com. <https://yourstory.com/mystory/what-software-types-examples>

### 6.2.1.8 Model answers to self-assessment

1. What is the meaning of the following terms?
  - i. **Computer program**- *it is a set of coded instructions given to the computer and represents a logical solution to a problem. It directs a computer in performing various operations/tasks on the data supplied to it.*
  - ii. **Programming**- *is the process of designing a set of instructions (computer programs) that can be used to perform a particular task or solve a specific problem. It involves the use of special characters, signs, and symbols found in a particular programming language to create computer instructions.*
  - iii. **Programming language**- *is a set of symbols (a language) that a computer programmer uses to solve a given problem using a computer.*
2. Concerning programming, distinguish between Source program and Object code.
  - *A source program refers to the program statements that the programmer enters in the program editor window, and which have not yet been translated into machine-readable form while an object code refers to the program code that is in machine-readable (binary) form.*
3. What is the function(s) of Assemblers, Interpreters, and Compilers in a computer system?

#### **Functions of assembler**

- *It checks whether the instructions are written are valid, and identifies any errors in the program.*
- *It assigns memory locations to the names the programmer uses.*
- *It generates the machine code equivalent to the Assembly instructions.*

#### **Functions of interpreter**

- *It reads a statement from the input, converts it to an intermediate code, executes it, then takes the next statement in sequence.*
- *If an error occurs, an interpreter stops execution and reports it.*

### ***Functions of a compiler***

- *It identifies the proper order of processing, to execute the process as fast as possible & minimize the storage space required in memory.*
  - *It allocates space in memory for the storage locations defined in the program to be executed.*
  - *It reads each line of the source program & converts it into machine language.*
  - *It checks for **Syntax errors** in a program (i.e., statements which do not conform to the grammatical rules of the language). If there are no syntax errors, it generates machine code equivalent to the given program.*
  - *It combines the program (machine) code generated with the appropriate subroutines from the library.*
  - *It produces a listing of the program, indicating errors if any.*
4. What is the difference between a Compiler and an Interpreter in programming?
- Interpreter** translates just one statement of the program at a time into machine code. **Compiler** scans the entire program and translates the whole of it into machine code at once. An **interpreter** takes very less time to analyze the source code*

easyvet.com

## **6.2.2 Learning Outcome 2: Identify Program Development Methodologies**

### **6.2.2.1 Introduction to the learning outcome**

This learning outcome specifies the content of competencies required to develop a computer program. It entails the description of program specifications, program development cycle, various types of development methodologies and Styles of programming.

### **6.2.2.2 Performance Standard**

- 6.2.2.2.1** Description of program specifications is done.
- 6.2.2.2.2** Application of the program development cycle is done.
- 6.2.2.2.3** Types of development methodologies are identified.
- 6.2.2.2.4** Styles of programming are identified.

### **6.2.2.3 Information Sheet**

#### **Description of Program specifications**

Program specification is the first step in developing a computer program.

- Program specification is also called program definition or program analysis.
- It requires the programmer to follow five specific tasks:
  - a. Specifying program objectives
  - b. Specifying desired output
  - c. Determining the required input
  - d. Defining processing requirements
  - e. Documentation specification

#### ***Task 1 - specify objectives, What are program objectives?***

- Program objectives are the problems you are trying to solve.
- A clear statement should be written about the problem that needs a solution.
- This task defines the problem.

***Task 2 - specify output, How do you determine the desired output?***

- It is always best to specify outputs before inputs.
- a. You need to know what you want to get out of the computer.
- b. Then you can determine what will go into the computer.
- Sketch or write how the output will look when it is done.

***Task 3 - input data - How do you determine the required input?***

- The source and type of data must be known.
- The input must supply the program with data to produce the correct output.

***Task 4 - processing requirements - How do you determine processing requirements?***

- Processing that must take place to convert input data into output information must correspond with the problem definition determined in task 1.
- A step-by-step logical algorithm must be determined to process the input data to output information.

***Tasks 5- program specifications document - What is included?***

- Document the program objectives, desired outputs, needed inputs, and required processing.
- After these items are documented, then step 2, program design can commence.

**Program development methodologies**

**Types of development methodologies**

- Agile
- Crystal
- Rapid Application Development
- **Agile**

AGILE methodology is a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project. In the Agile model, both development and testing activities are concurrent, unlike the Waterfall model.

Click on the link below for further readings

[https://youtu.be/rvTejAg\\_fbY](https://youtu.be/rvTejAg_fbY)

- **Crystal method**

Crystal family is a collection of agile software development methodologies that can be used for different software projects depending upon size, complexity, criticality, and the number of people involved. It was developed by Alistair Cockburn in early 1990 while working at IBM. He interviewed different teams working on different projects to find the best practices followed by teams. He found that these teams did not follow the formal methodologies or not using specific technology for delivering successful software. However, they communicated frequently to discuss the project. On the other hand, delayed or failed project teams tried to follow formal methods with little team collaboration. This helped him to conclude that frequent communication among team members can improve the software success rate. According to Cockburn's philosophy "To the extent that you can replace the written documentation with face to face interaction, you can reduce the reliance on written 'promissory' notes and improve the likelihood of delivering the system. Crystal methods focus on people and communication among people rather than a process to frequently deliver working software.

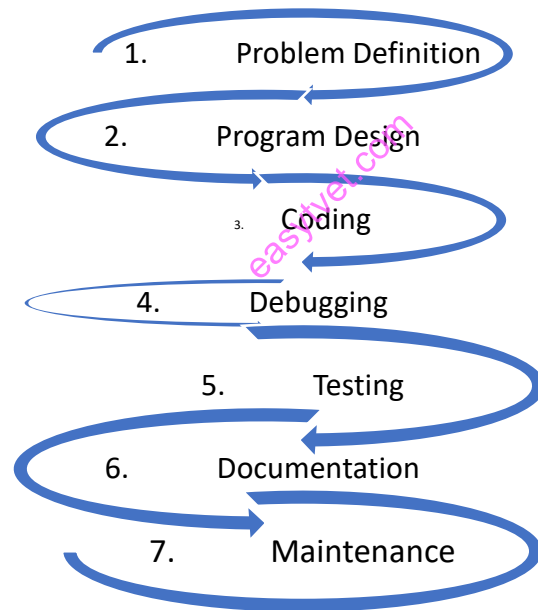
- **Rapid Application Development**

The rapid Application Development model is based on prototyping and iterative development with no specific planning involved. It focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using the iterative concept, reuse of the existing prototypes (components), continuous integration, and rapid delivery.

## **Program Development Life Cycle**

-The program development life cycle is a set of steps or phases that are used to develop a program in any programming language. Generally, the program development life cycle contains 7 phases, they are as follows

- i. Problem Definition
- ii. Program Design
- iii. Coding
- iv. Debugging
- v. Testing
- vi. Documentation
- vii. Maintenance



**Figure 178 program development life cycle(PDLC)**

- i. **Problem Definition:**



- The first step in the process of program development is the thorough understanding and identification of the problem for which the program or software is to be developed.
- In this step, the problem has to be defined formally.
- All the factors like Input/output, processing requirement, memory requirements, error handling, interfacing with other programs have to be taken into consideration in this stage.

## ii. Program Design:

- The next stage is the program design. The software developer makes use of tools like algorithms and flowcharts to develop the design of the program.
  - Algorithm
  - Flowchart

## iii. Coding:

- Once the design process is complete, the actual computer program is written, i.e. the instructions are written in a computer language.
- Coding is generally a very small part of the entire program development process and also a less time-consuming activity in reality.
- In this process all the syntax errors i.e. errors related to spelling, missing commas, undefined labels etc. are eliminated.
- For effective coding some of the guidelines which are applied are :
  - Use of meaningful names and labels of variables,
  - Simple and clear expressions,
  - Modularity with emphasis on making modules generalized,
  - Making use of comments and indenting the code properly,
  - Avoiding jumps in the program to transfer control.

## iv. Debugging:

- At this stage, the errors in the programs are detected and corrected.

- This stage of program development is an important process. Debugging is also known as program validation.
- Some common errors which might occur in the programs include:
  - Un initialization of variables.
  - Reversing of an order of operands.
  - Confusion of numbers and characters.
  - Inverting of conditions eg jumping on zero instead of on not zero.

v. Testing:

- The program is tested on several suitable test cases.
- A test plan of the program has to be done at the stage of the program design itself.
- This ensures a thorough understanding of the specifications.
- The most trivial and most special cases should be identified and tested.
- It is always useful to include the maximum and minimum values of all variables as test data.

vi. Documentation:

- Documentation is a very essential step in program development.
- Documentation helps the users and the people who maintain the software.

This ensures that future modifications if required can be done easily. Also, it is required during redesigning and maintenance.

vii. Maintenance:

- Updating and correction of the program for changed conditions and field experience are accounted for in maintenance.
- Maintenance becomes essential in the following situations:
  - Change in the specification,
  - Change in equipment,
  - Errors are found during the actual execution of the program.

## Styles of programming

### Types of styles programming

- Functional Programming
- Object-Oriented Programming
- Modular programming

### Functional Programming

Here the problem, or the desired solution, is broken down into functional units. Each unit performs its task and is self-sufficient. These units are then stitched together to form the complete solution.

**Example** – A payroll processing can have functional units like employee data maintenance, basic salary calculation, gross salary calculation, leave processing, loan repayment processing, etc.

### Object-Oriented Programming

Here the solution revolves around entities or objects that are part of the problem. The solution deals with how to store data related to the entities, how the entities behave, and how they interact with each other to give a cohesive solution.

**Example** – If we have to develop a payroll management system, we will have entities like employees, salary structure, leave rules, etc. around which the solution must be built.

### Modular programming

**Modular programming** is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality.

#### 6.2.2.4 Learning Activity

##### Practical activity

##### special instructions

Using your computer, develop a detailed document to be used by a newly employed person in your organization capturing the step by step process of Program Development Life Cycle

#### **6.2.2.5 Self-Assessment**

-You are provided with the following questions for self -assessment, attempt them and check your responses

1. What is PDLC?
2. Which are the Three major program development approaches used in PDLC?
3. What is the difference between Functional programming and Modular Programming?
4. Why is the Design phase important in Program Development Life Cycle?
5. Which are the five common errors that can occur in a program during program development?

#### **6.2.2.6 Tools, Equipment, Supplies, and Materials**

- Flow charts
- Data flow diagram
- Decision table
- Decision tree
- Web Authoring tools
- Notepad
- Computer
- Software
- Digital instructional material including DVDs and CDs

#### **6.2.2.7 References**

Blignaut, A. S., Hinostroza, J. E., Els, C. J., & Brun, M. (2010). ICT in education policy and practice in developing countries: South Africa and Chile compared through SITES 2006. *Computers & Education*, 55(4), 1552–1563.

*Computer Hardware.* (n.d.). Retrieved September 30, 2020, from <https://web.stanford.edu/class/cs101/hardware->

KIRIHATA, Y. (2009). *Information processing apparatus and method, computer-readable recording medium, and an external storage medium* (United States Patent No. US20090241114A1).

Lehmann, S., & Schweitzer, B. (2006). *Apparatus, system, and method for creating customized workflow documentation* (United States Patent No. US20060059423A1).

Nagar, T. (2019, December 3). *What is software and types of software with examples?* YourStory.Com. <https://yourstory.com/mystory/what-software-types-examples>

#### **6.2.2.7 Model answers to self-assessment**

**1. What is PDLC?**

*-PDLC is an abbreviation of Program Development Life Cycle. It is a set of steps or phases that are used to develop a program in any programming language.*

**2. Which are the Three major program development approaches used in PDLC**

- i. Agile*
- ii. Crystal method*
- iii. Rapid Application Development*

**3. What is the difference between Functional programming and Modular Programming**

*-In functional programming, the problem is broken down into functional units where each unit performs its task and is self-sufficient. These units are then stitched together to form the complete solution while in modular programming the functionality of a program is separated into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality.*

**4. Why is the Design phase important in Program Development Life Cycle?**

*-The requirements are specified in the form of a document. It is then converted into a logical structure that needs to be implemented in a specific programming language.*

*-The design phase is also helpful for specifying hardware & system requirements. It also allows defining complete system architecture. The output is designed to document that acts as an input for all the subsequent PDLC phases.*

5. Which Five common errors that can occur in a program during program development

- *Un initialization of variables.*
- *Reversing of an order of operands.*
- *Confusion of numbers and characters.*
- *Inverting of conditions eg jumping on zero instead of on not zero.*

*easytvvet.com*

## **6.2.3 Learning Outcome 3: Identify Program design**

### **6.2.3.1 Introduction to the learning outcome**

This learning outcome specifies the content of competencies required to design a computer program. It entails the description of Program design, various approaches for designing a program, and Program design tools.

### **6.2.3.2 Performance Standard**

6.2.3.2.1 Description of Program design is done.

6.2.3.2.2 Program design approaches are identified.

6.2.3.2.3 Program design tools are identified.

### **6.2.3.3 Information Sheet**

- **Program design**

The program design is the process that an organization uses to develop a program. It is most often an iterative process involving research, consultation, initial design, testing, and redesign. A program design is the plan of action that results from that process.

For further readings on Program design, click on the link below.

<https://youtu.be/aXPIxUafbM8>

- **Program design approaches**

- Top – Down
- Bottom-Up
- Data-Driven

### ***Top-Down Design Model:***

Top-down programming starts by implementing the most general modules and works toward implementing those that provide specific functionality. In the top-down model, an overview of the system is formulated without going into detail for any part of it. Each part of it then refined into more details, defining it in yet more details until the entire specification is detailed enough to validate the model. If we glance at a haul as a full, it's going to appear not possible as a result of it's so complicated. For example: Writing a University system program, writing a word processor. Complicated issues may be resolved victimization high down style, conjointly referred to as Stepwise refinement where,

1. We break the problem into parts,
2. Then break the parts into parts soon and now each of the parts will be easy to do.

#### ***Bottom-Up Design Model:***

In this design, individual parts of the system are specified in detail. The parts are linked to form larger components, which are in turn linked until a complete system is formed. Bottom-up programming implements the modules that provide specific functionality first and then integrates them by implementing the more general modules

#### ***Data-driven approach***

When doing *data-driven programming*, one clearly distinguishes code from the data structures on which it acts and designs both so that one can make changes to the logic of the program by editing not the code but the data structure.

Data-driven programming is sometimes confused with object-orientation, another style in which data organization is supposed to be central. There are at least two differences. One is that in data-driven programming, the data is not merely the state of some object, but defines the control flow of the program. Where the primary concern in OO is encapsulation, the primary concern in data-driven programming is writing as little fixed code as possible. Unix has a stronger tradition of data-driven programming than OO.

Programming data-driven style is also sometimes confused with writing state machines. It is, in fact, possible to express the logic of a state machine as a table or data structure, but hand-coded state machines are usually rigid blocks of code that are far harder to modify than a table. An important rule when doing any kind of code generation or data-driven programming is this: always push problems upstream. Don't hack the generated code or any intermediate



representations by hand — instead, think of a way to improve or replace your translation tool. Otherwise, you're likely to find that hand-patching bits that should have been generated correctly by the machine will have turned into an infinite time sink.

## **Program Design Tools**

Computers are problem-solving devices and it is an electronic programmable machine that requires a set of instructions to perform tasks. To facilitate a computer to solve problems effectively, clear and concise instructions must be provided to it. A programmer must provide clear instruction to perform a task and he must have a plan to solve a particular problem using programming tools like algorithm, flowchart, and pseudocode.

### **I. Algorithm**

An algorithm is the stepwise logical instructions written in any human-understandable language to solve a particular problem in a finite amount of time. It is written in simple English language.

**Steps used to develop the algorithm are:**

- i. The problem has to be understood by the programmer.
- ii. The expected output has to be identified.
- iii. The logic that will produce the required output from the input has to be developed.
- iv. The algorithm should be tested for accuracy for a given set of input data.
- v. The steps are repeated till the desired result is produced.

### **Characteristics of an Algorithm**

- All the instructions of the algorithm should be simple.
- The logic of each step must be clear.
- There should be a finite number of steps for solving problems.

### **Example of an algorithm**

An algorithm for adding two-digit numbers is:

1. add the tens

2. add the ones
3. add the numbers from steps 1 and 2

So to add 15 and 32 using that algorithm:

1. add 10 and 30 to get 40
2. add 5 and 2 to get 7
3. add 40 and 7 to get 47

## II. Flowchart



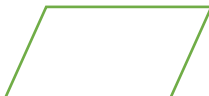
The flowchart is a pictorial representation of a stepwise solution to a problem. It helps the programmer in developing the program logic and serves as documentation for future reference.

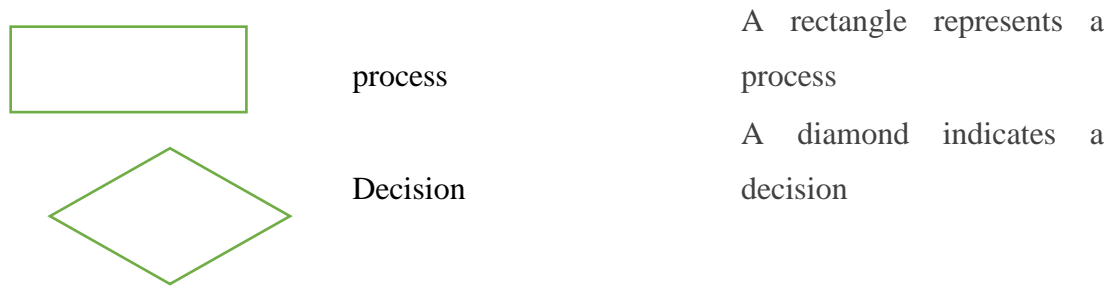
It uses different boxes linked by the arrows. The process of drawing a flowchart is flowcharting.

### Rules for developing a flowchart.

- Analyze the input, process, storage, and output information.
- Use standard symbols and arrowhead to the direction of flow of data and instructions.
- Use simple words that can be easily understood by other programmers.
- There should be a list of activities inside each symbol.

**Table 42: flowchart symbols**

Symbol	Name	Function
	Start/End	An oval represents the start or endpoint
	Arrows	An arrow is a connector that shows a relationship between representative shapes
	Input/output	A parallelogram represents input or output



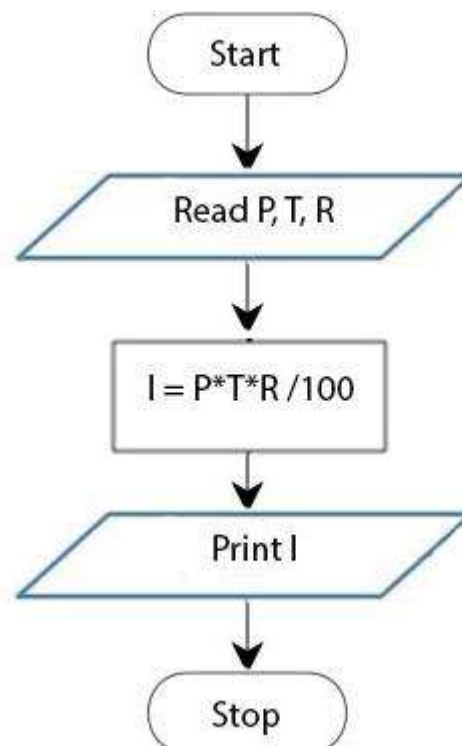
### Advantages

- The flowchart helps programmers to explain the logic of a program to others easily.
- The flowchart provides documentation support.
- A flowchart helps detect and remove bugs in a program in a systematic manner.
- With the reference of the Flowchart, the programmer can write a program and correspond to the program.

### Disadvantages

- Flowcharts are time-consuming and laborious to draw with proper symbols.
- The complicated logic of a program is not easy to represent

**Figure 179 Flowchart example**



### III. Pseudocode

A pseudocode is a method of documenting a program logic in which English-like statements are used to describe the processing steps.

These are structured English-like phrases that indicate the program steps to be followed to solve a given problem.

The term “Code” usually refers to a computer program. This implies that, some of the words used in pseudocode may be drawn from a certain programming language and then mixed with English to form structured statements that are easily understood by non-programmers, and also make a lot of sense to programmers. However, pseudocodes are not executable by a computer.

#### **Guidelines for designing good pseudocode.**

1. The statements must be short, clear, and readable.
2. The statements must not have more than one meaning (i.e., should not be ambiguous).
3. The pseudocode lines should be clearly outlined and indented.
4. Pseudocode must have a Begin and an end. i.e., pseudocode should show clearly the start and stop of executable statements and the control structures.
5. The input, output & processing statements should be clearly stated using keywords such as PRINT, READ, INPUT, etc

### **Decision Tables**

A Decision table represents conditions and the respective actions to be taken to address them, in a structured tabular format.

It is a powerful tool to debug and prevent errors. It helps group similar information into a single table and then by combining tables it delivers easy and convenient decision-making.

### **Creating a Decision Table**

To create the decision table, the developer must follow the basic four steps:

- Identify all possible conditions to be addressed.
- Determine actions for all identified conditions.
- Create Maximum possible rules.

For your further readings on Decision tables,click on the link below.

<https://youtu.be/A5-w3mof-3I>

#### **6.2.3.4 Learning Activities**

##### **Practical activity**

Using your computer,prepare detailed document to be used by a newly employed person in your organization capturing the following program design tools;

- i. Pseudocodes.
- ii. Flowcharts.
- iii. Decision Tables.
- iv. Decision Trees

#### **6.2.3.5 Self-Assessment**

1. .What is the difference between Top-Down and Bottom-Up program design approaches?
2. What is the difference between pseudocode and flowchart program design tools
3. What are some of the disadvantages of a flowchart?
4. What are the rules of developing a flowchart
5. Write a structured algorithm that would prompt the user to enter the Length and Width of a rectangle, calculate the Area and Perimeter, then display the result.

### 6.2.3.6 Tools, Equipment, Supplies and Materials

- Flow charts
- Data flow diagram
- Decision table
- Decision tree
- Web Authoring tools
- Notepad
- Computer
- Software
- Digital instructional material including DVDs and CDs

### 6.2.3.7 References

Blignaut, A. S., Hinojosa, J. E., Els, C. J., & Brun, M. (2010). ICT in education policy and practice in developing countries: South Africa and Chile compared through SITES 2006. *Computers & Education*, 55(4), 1552–1563.

*Computer Hardware*. (n.d.). Retrieved September 30, 2020, from <https://web.stanford.edu/class/cs101/hardware->

KIRIHATA, Y. (2009). *Information processing apparatus and method, computer-readable recording medium, and an external storage medium* (United States Patent No. US20090241114A1).

Lehmann, S., & Schweitzer, B. (2006). *Apparatus, system, and method for creating customized workflow documentation* (United States Patent No. US20060059423A1).

Nagar, T. (2019, December 3). *What is software and types of software with examples?* YourStory.Com. <https://yourstory.com/mystory/what-software-types-examples>

### 6.2.3.8 Model answers to self assessment

1. What is the difference between Top-Down and Bottom-Up program design approaches?
  - *Top-down programming starts by implementing the most general modules and works toward implementing those that provide specific functionality while in Bottom-*

*up approach, individual parts of the system are specified in detail. The parts are linked to form larger components, which are in turn linked until a complete system is formed*

2. Give the difference between pseudocode and flowchart program design tools

*-pseudocode is a method of documenting a program logic in which English-like statements are used to describe the processing steps while flowchart while is a diagram that demonstrates the logical sequence of events that must be performed to solve a problem.*

3. What are some of the disadvantages of a flowchart?

- *Flowcharts are complex, clumsy & become unclear, especially when the program logic is complex.*
- *If changes are to be made, the flowchart may require complete re-drawing.*
- *No uniform practice is followed for drawing flowcharts as it is used as an aid to the program.*
- *Sometimes, it becomes difficult to establish the link between various conditions, and the actions to be taken upon a particular condition.*
- *Reproduction of flowcharts is usually a problem, since the flowchart symbols cannot be typed.*

4. State the rules of developing a flowchart

- *A flowchart must have a Start & an end.*
- *It should have only one entry (starting point) & one exit point (i.e., ensure that the flowchart has a logical start and finish).*
- *It should be clear, neat & easy to follow.*
- *The logical flow should be clearly shown using arrows.*
- *Use the correct symbol at each stage in the flowchart.*
- *Make comparison instructions simple, i.e., capable of YES/NO answers.*
- *Avoid overlapping the lines used to show the flow of logic as this can create confusion in the flowchart.*
- *The flowchart should not be open to more than one interpretation.*
- *Where necessary, use Connectors to reduce the number of flow lines.*
- *Ensure that the flowchart is logically correct & complete.*

5. Write a structured algorithm that would prompt the user to enter the Length and Width of a rectangle, calculate the Area and Perimeter, then display the result.

Step 1: Write down the Pseudocode.

START

PRINT "Enter Length and Width"

READ L, W

Area =  $L * W$

Perimeter =  $2 (L + W)$

PRINT Area, Perimeter

STOP

Step 2: Design a flowchart for the program.

easytvet.com

## **6.2.4 Learning Outcome 4: Identify Computer Programming Languages**

### **6.2.4.1 Introduction to the learning outcome**

This learning outcome specifies the content of competencies required to design a computer program. It entails information on generations of programming languages, factors for choosing a programming language and basic tools for program development.

### **6.2.4.2 Performance Standard**

6.2.4.2.1 Generations of programming languages are Identified.

6.2.4.2.2 Factors for choosing a programming language are determined.

6.2.4.2.3 Basic tools for program development are identified.



### 6.2.4.3 Information Sheet

- **Programming Languages**

A programming language is a set of words, symbols, and codes that enables a programmer to communicate a solution algorithm to the computer. Just as humans understand a variety of spoken languages (English, Spanish, French, and so on), computers recognize a variety of programming languages.

Programming languages are classified into 2 major categories:

- I. Low-level programming languages.
- II. High-level programming languages.

Each programming language has its own grammatical (syntax) rules, which must be obeyed in order to write valid programs, just as a natural language has its own rules for forming sentences.

#### **I. Low-level languages**

These are the basic programming languages, which can easily be understood by the computer directly, or which require little effort to be translated into computer understandable form.

They include:

- Machine languages.
- Assembly languages.

#### *Features of low-level languages*

- They are machine hardware-oriented.
- They are not portable, i.e., a program written for one computer cannot be installed and used on another computer of a different family.
- They use Mnemonic codes.
- They frequently use symbolic addresses.

## **Machine languages (1st Generation computer languages)**

-Machine language is written using machine codes (binary digits) that consist of 0's & 1's.

The computer can readily understand Machine code instructions without any translation.

A programmer is required to write his program in strings of 0's & 1's, calculate & allocate the core memory locations for his data and/or instructions.

## **Assembly language (2nd Generation computer languages).**

Assembly languages were developed to speed up programming (i.e., to overcome the difficulties of understanding and using machine languages).

The vocabulary of Assembly languages is close to that of machine language, and its instructions are symbolic representations of the machine language instructions.

- ◆ Assembly language programs are easier to understand, use & modify compared to Machine language programs.
- ◆ Assembly language programs have fewer error chances.

To write program statements in Assembly language, the programmer uses a set of predefined symbols (operation codes) called Mnemonic codes.

## **Advantages of Low-level languages**

- The CPU can easily understand machine language without translation.
- They have closer control over the hardware, are highly efficient & allow direct control of each operation.
- They are therefore suitable for writing Operating system software & Game programs, which require fast & efficient use of the CPU time.
- The program instructions can be executed by the hardware (processor) much faster. This is because; complex instructions are already broken down into smaller simpler ones.
- They require less memory space.
- They are stable, i.e., they do not crash once written.

## **Disadvantages of Low-level languages**

- Very few computer programs are written in a machine or Assembly language because of the following reasons;
- Low-level languages are difficult to learn, understand, and write programs in them.
- Low-level language programs are difficult to debug (remove errors from).
- The programs are very long; hence, writing a program in a low-level language is usually tedious & time-consuming.
- The programs are difficult to develop, maintain, and are also prone to errors (i.e., it requires highly trained experts to develop and maintain the programs).
- Low-level languages are machine-dependent (specific), hence non-portable. This implies that they are designed for a specific machine & specific processor, and therefore, cannot be transferred between machines with different hardware specifications.

## **II. High-level programming languages**

High-level languages were developed to solve (overcome) the problems encountered in low-level programming languages.

The grammar of High-level languages is very close to the vocabulary of the natural languages used by human beings. Hence; they can be read and understood easily even by people who are not experts in programming.

Most high-level languages are general-purpose & problem-oriented. They allow the programmer to concentrate on the functional details of a program rather than the details of the hardware on which the program will run.

High-level language programs are machine-independent, (i.e., they do not depend on a particular machine, and can run in any family of computers provided the relevant translator software is installed).

### **Advantages of High-Level Languages:**

- It is close to the human being
- It is easy to understand
- It consists of an English language like structure
- It does not depend upon the machine
- It is easy to modify

The programs written in high-level languages is called source code Example of high-level languages are BASIC, PASCAL, C/C++, etc.

### **Disadvantages of High-level languages.**

- High-level languages are not machine-oriented; hence, they do not use the CPU and hardware facilities efficiently.
- The languages are machine-independent and cannot be used in programming the hardware directly.
- Each high-level language statement converts into several machine code instructions. This means that, they use more storage space, and it also takes more time to run the program.
- Their program statements are too general; hence, they execute slowly than their machine code program equivalents.
- They have to be interpreted or compiled to machine-readable form before the computer can execute them.
- The languages cannot be used on very small computers.
- The source program written in a high-level language needs a Compiler, which is loaded into the main memory of the computer, and thus occupies much of memory space. This greatly reduces the memory available for a source program.

### **III. Fourth-generation languages (4gl's).**

4GLs make programming even easier than the 3GLs because; they present the

programmer with more programming tools, such as command buttons, forms, text boxes, etc. The programmer simply selects graphical objects called controls on the screen and then uses them to create designs on a form by dragging a mouse pointer.

The languages also use application generators (which are in the background) to generate the necessary program codes; hence, the programmer is freed from the tedious work of writing the code.

4GLs are used to enquire & access the data stored in database systems; hence, they are described as the Query Languages such as Structured Query Languages (SQL), Report Generators, Application Generators, Decision-support languages & Graphics languages

*Examples of 4GLs are:*

- Visual Basic
- Delphi Pascal
- Visual COBOL (Object COBOL)
- Access Basic

*Advantages of fourth-generation languages.*

- They are user-based, and therefore, easy to learn & understand.
- Their grammar is very close to the natural English language.
- They use menus & prompts to guide a non-specialist to retrieve data easily.
- Very little training is required to develop & use 4GL programs.
- They provide features for the formatting of input, processing, & instant reporting.

#### **IV. Object-oriented programming Language**

Object-Oriented Programming uses objects. An Object is a representation of a software entity.

such as a user-defined window or variable.

In OOP the data & procedures that operate on data are combined into one object. Several objects can be linked together to form a complete program.

Each object has specific data values that are unique to it called state & a set of the things it can accomplish called functions or behavior. Therefore, programs send

messages to an object to perform a procedure that is already embedded in it. The process of having data & functions that operate on the data within an object is called **encapsulation**.

OOP is greatly applied in the development of GUI operating systems & application programs.

OOP enables rapid program development. Every object has properties such as colour, size, data source, etc, which can be set easily without much effort. Besides, every object has events associated with it that can be used to trigger certain actions, e.g. remove the window from the screen by clicking the 'Close' button.

*Examples of Object-oriented programming languages are: -*

- Simula
- C++
- Smalltalk
- Java
- Python

Click on the link below for further reading on Object-Oriented Programming.

<https://youtu.be/SiBw7os-zI>

## **Visual Basic programming Language**

Visual Basic is a programming language and development environment created by Microsoft. It is an extension of the BASIC programming language that combines BASIC functions and commands with visual controls. Visual Basic provides a graphical user interface GUI that allows the developer to drag and drop objects into the program as well as manually write program code.

- Visual Basic, also referred to as "VB," is designed to make software development easy and efficient, while still being powerful enough to create advanced programs. For example, the Visual Basic language is designed to be "human-readable," which means the source code can be understood without requiring lots of comments. The Visual Basic program also includes features like "IntelliSense" and "Code Snippets," which automatically generate code for visual objects added by the programmer. Another feature, called "AutoCorrect," can debug

the code while the program is running.

-Programs created with Visual Basic can be designed to run on Windows, on the Web, within Office applications, or on mobile devices. Visual Studio, the most comprehensive VB development environment, or IDE can be used to create programs for all these mediums. Visual Studio .NET provides development tools to create programs based on the .NET framework, such as ASP.NET applications, which are often deployed on the Web.

- **Factors to consider when choosing a programming language.**

The following factors should be considered when choosing a Programming language to use in solving a problem:

- i. *The availability of the relevant translator.* Translators help in converting the Source codes (program statements written in any of the computer programming languages) to their Object codes (computer language equivalents).
- ii. *Ease of learning and use.* The programming language chosen should be the one that is easy for users to learn and use
- iii. *Purpose of the program,* i.e., application areas such as education, business, science, etc.
- iv. *Execution time:* - Applications that require a quick response are best programmed in machine code or assembly language. High-level languages are not suitable for such an application because they take longer to be translated & executed.
- v. *Development time:* - Development time is the time a programmer takes to write and run a program. High-level languages are easy to read, understand and develop; hence, they require less development time. Machine code & Assembly languages are relatively difficult to read, understand and develop; hence, they are time-consuming.
- vi. *Popularity:* - The language selected should be suitable and/or successful in the market concerning the problems to be solved.
- vii. *Documentation:* - It should have accompanying documentation (descriptions) on how to use the language or maintain the programs written in the language.
- viii. *Maintenance:* - Programs are developed to solve specific problems, and the problems keep on changing; hence, the programs are also changed to perform the new functions. Program maintenance is the activity of incorporating more routines into the program,

modifying the existing routines, or removing the obsolete routines to make the program adapt to a functionally enhanced environment. The maintenance is made easier if the language used is easy to read and understand.

- **Program development tools**

some of the program development tools include.

- Pseudocode
- flow charts
- Data flow diagrams

***Pseudocodes.***

A pseudocode is a method of documenting a program logic in which English-like statements are used to describe the processing steps.

These are structured English-like phrases that indicate the program steps to be followed to solve a given problem.

The term “Code” usually refers to a computer program. This implies that some of the words used in pseudocode may be drawn from a certain programming language and then mixed with English to form structured statements that are easily understood by non-programmers, and also make a lot of sense to programmers. However, pseudocodes are not executable by a computer.

***Guidelines for designing good pseudocode.***

1. The statements must be short, clear, and readable.
2. The statements must not have more than one meaning (i.e., should not be ambiguous).
3. The pseudocode lines should be clearly outlined and indented.
4. A pseudocode must have a Begin and an end.  
i.e., pseudocode should show clearly the start and stop of executable statements and the control structures.
5. The input, output & processing statements should be clearly stated using keywords such as PRINT, READ, INPUT, etc.

**Flowcharts.**



A Flowchart is a diagram that demonstrates the logical sequence of events that must be performed to solve a problem.

It is a diagrammatic or pictorial representation of a program's algorithm.

*Types of Flowcharts.*

There are 2 common types of Flowcharts:

1). **System flowchart.**

A System flowchart is a graphical model that illustrates each basic step of a data processing system.

-It illustrates (in summary) the sequence of events in a system, showing the department or function responsible for each event.

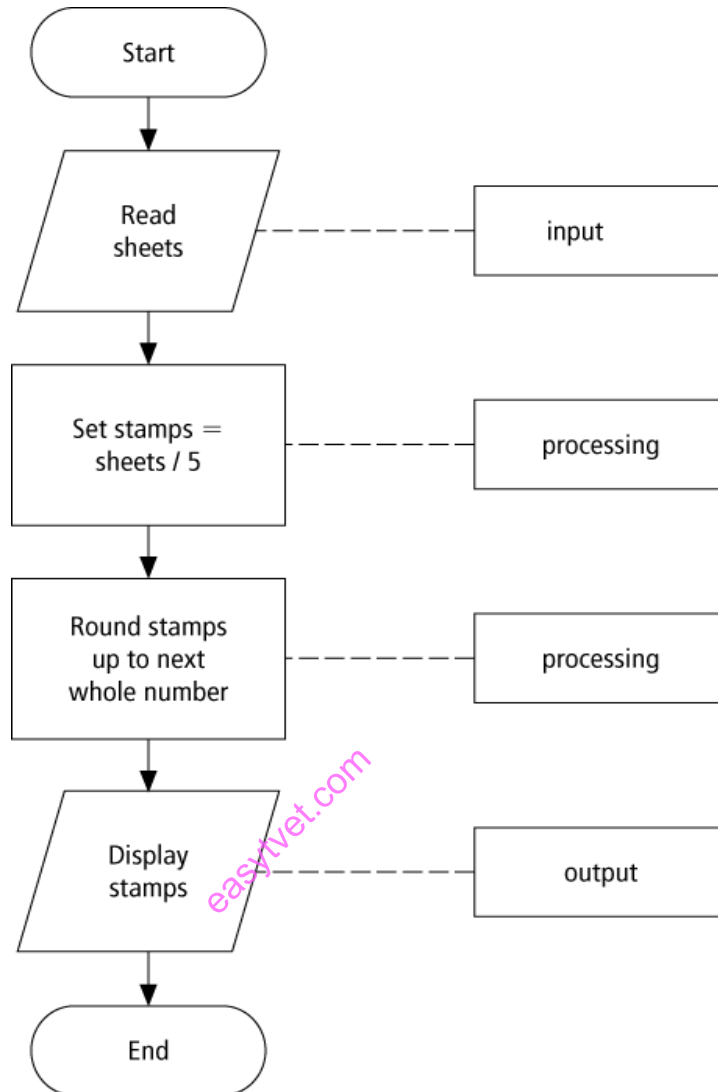
2). **Program flowchart.**

This is a diagram that describes, in sequence, all the operations required to process data in a computer program.

-A program flowchart graphically represents the types of instructions contained in a computer program as well as their sequence & logic.

*Guidelines for drawing a program flowchart.*

- A flowchart must have a Start & an end.
- It should have only one entry (starting point) & one exit point (i.e., ensure that the flowchart has a logical start and finish).
- It should be clear, neat & easy to follow.
- The logical flow should be clearly shown using arrows.
- Use the correct symbol at each stage in the flowchart.
- Make comparison instructions simple, i.e., capable of YES/NO answers.
- Avoid overlapping the lines used to show the flow of logic as this can create confusion in the flowchart.
- The flowchart should not be open to more than one interpretation.
- Where necessary, use Connectors to reduce the number of flow lines.
- . Ensure that the flowchart is logically correct & complete.



**Figure 180:Example of a flow chart**

### **Data flow diagrams**

A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various subprocesses the data

moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships.

Data flow diagrams visually represent systems and processes that would be hard to describe in a chunk of text. You can use these diagrams to map out an existing system and make it better or to plan out a new system for implementation. Visualizing each element makes it easy to identify inefficiencies and produce the best possible system.

#### **6.2.4.4 Learning Activities**

##### **Practical activity**

In your institution computer laboratory, perform the following;

- i. You need to calculate the area of the rectangle using its length and width. Draw a flowchart and write pseudocode that will accept the length and width and calculate and print the area of the rectangle. (area = length \* width)
- ii. Net salaries of employees are paid after the calculation of their deductions and allowances. The following rates are used for calculating these allowances and deductions. Allowances Deductions HRA 15% of Basic SS 7% of Basic DA 10% of Basic Levy 1% of Basic EA 5% of Basic TA 12% of Basic To calculate the Gross salary the following formulas are used: Gross salary = Basic salary+ allowances Net salary = Gross salary - deductions Write pseudocode that will prompt the user to enter the name and basic salary of an employee and output their name, basic salary, allowances, deductions, gross salary and net salary with appropriate labels.

#### **6.2.4.5 Self-Assessment**

You are provided with the following questions for self -assessment, attempt them and check your responses.

- a. What is a Programming language?
- b. What is meant by 'Machine language'?
- c. Show the difference between Machine language and Assembly language.

- d. Give two advantages & three disadvantages of Machine language programming.
- e. Give the features/characteristics of high-level programming languages.
- f. List 8 factors that need to be considered when selecting a programming language.
- g. Write pseudocode that will accept 25 integers and displays the number of positive and negative numbers.
- h. Write a pseudocode algorithm that will create a conversion table to convert degrees Celsius to degree Fahrenheit. Prompt the user to enter the temperature in degree Celsius and display the temperature in Fahrenheit.  
(Fahrenheit = 32+ (9\*Celcius/5))

#### 6.2.4.6 Tools, Equipment, Supplies, and Materials

- Flow charts
- Data flow diagram
- Decision table
- Decision tree
- Web Authoring tools
- Notepad
- Computer
- Software
- Digital instructional material including DVDs and CDs

#### 6.2.4.7 References

Blignaut, A. S., Hinostroza, J. E., Els, C. J., & Brun, M. (2010). ICT in education policy and practice in developing countries: South Africa and Chile compared through SITES 2006. *Computers & Education*, 55(4), 1552–1563.

*Computer Hardware*. (n.d.). Retrieved September 30, 2020, from <https://web.stanford.edu/class/cs101/hardware->

KIRIHATA, Y. (2009). *Information processing apparatus and method, computer-readable recording medium, and an external storage medium* (United States Patent No. US20090241114A1).

Lehmann, S., & Schweitzer, B. (2006). *Apparatus, system, and method for creating customized workflow documentation* (United States Patent No. US20060059423A1).

Nagar, T. (2019, December 3). *What is software and types of software with examples?* YourStory.Com. <https://yourstory.com/mystory/what-software-types-examples>

#### 6.2.4.8 Model answers to self-assessment

- a. What is a Programming language?
  - *A programming language is a set of words, symbols, and codes that enables a programmer to communicate a solution algorithm to the computer*
- b. What is meant by ‘Machine language’?
  - *The machine-level language is a language that consists of a set of instructions that are in the binary form 0 or 1. As we know that computers can understand only machine instructions, which are in binary digits, i.e., 0 and 1, so the instructions given to the computer can be only in binary codes.*
- c. Show the difference between Machine language and Assembly language.

**Table 43 Difference between Machine and Assembly Language**

<b>Machine-level language</b>	<b>Assembly language</b>
<i>The machine-level language comes at the lowest level in the hierarchy, so it has zero abstraction level from the hardware.</i>	<i>The assembly language comes above the machine language means that it has less abstraction level from the hardware.</i>
<i>It cannot be easily understood by humans.</i>	<i>It is easy to read, write, and maintain.</i>
<i>The machine-level language is written in binary digits, i.e., 0 and 1.</i>	<i>The assembly language is written in simple English language, so it is easily understandable by the users.</i>
<i>It does not require any translator as the machine code is directly executed by the computer.</i>	<i>In assembly language, the assembler is used to convert the assembly code into machine code.</i>
<i>It is a first-generation programming language.</i>	<i>It is a second-generation programming language.</i>

- d. What are the features/characteristics of high-level programming languages.
- *They contain statements that have an extensive vocabulary of words, symbols, sentences & mathematical expressions, which are very similar to the normal English language.*
  - *Allow modularization (use of sub-routines).*
  - *They are 'user-friendly' and problem-oriented rather than machine-based. This implies that, during a programming session, the programmer concentrates on problem-solving rather than how a machine operates.*
  - *They require one to obey a set of rules when writing the program.*
  - *Programs written in high-level languages are shorter than their low-level language equivalents since one statement translates into several machine code instructions.*
  - *The programs are portable between different computers.*
- e. List 8 factors that need to be considered when selecting a programming language.
- *The availability of the relevant translator.* Translators help in converting the Source codes (program statements written in any of the computer programming languages) to their Object codes (computer language equivalents).
  - *Ease of learning and use.* The programming language chosen should be the one that is easy for users to learn and use
  - *Purpose of the program,* i.e., application areas such as education, business, science, etc.
  - *Execution time:* - Applications that require a quick response are best programmed in machine code or assembly language. High-level languages are not suitable for such an application because they take longer to be translated & executed.
  - *Development time:* - Development time is the time a programmer takes to write and run a program. High-level languages are easy to read, understand

and develop; hence, they require less development time. Machine code & Assembly languages are relatively difficult to read, understand and develop; hence, they are time-consuming.

- *Popularity*: - The language selected should be suitable and/or successful in the market concerning the problems to be solved.
- *Documentation*: - It should have accompanying documentation (descriptions) on how to use the language or maintain the programs written in the language.
- *Maintenance*: - Program maintenance is the activity of incorporating more routines onto the program, modifying the existing routines, or removing the obsolete routines to make the program adapt to a functionally enhanced environment. The maintenance is made easier if the language used is easy to read and understand.

- f. Write pseudocode that will accept 25 integers and displays the number of positive and negative numbers

Step 1: start

Step 2: poscount = 0, negcount = 0

Step 3: for i = 1 to 25

Step 3: read num

Step 4: if num > 0 then

poscount = poscount + 1

else

negcount = negcount + 1

endif

Step 5: end for

Step 6: write poscount, negcount

- g. Write a pseudocode algorithm that will create a conversion table to convert degrees Celsius to degree Fahrenheit. Prompt the user to enter the temperature in degree Celsius and display the temperature in Fahrenheit. (Fahrenheit = 32 + (9 \* Celsius / 5))

**solution**

Step 1: start

Step 2: write "Enter the temperature in degrees Celcius."

Step 3: read Celsius

Step 4: Fahrenheit =  $32 + (9 * \text{celcius} / 5)$

Step 5: write "The temperature in Fahrenheit is:"

Step 6: write Fahrenheit

Step 7: stop

[easytvvet.com](http://easytvvet.com)



## **6.2.5 Learning Outcome 5: Perform Basic Structured programming using C language.**

### **6.2.5.1 Introduction to the learning outcome**

This learning outcome specifies the content of competencies required to design a computer program. It entails information fundamentals of C programming, Control structures in C programming, Subprograms of C language, C language concepts, C programming environment, description of sub programming, C program format.

### **6.2.5.2 Performance Standard**

- 6.2.5.2.1 Fundamentals of C programming are identified.
- 6.2.5.2.2 Control structures in C programming are identified.
- 6.2.5.2.3 Subprograms of C language are explained.
- 6.2.5.2.4 C language concepts are identified.
- 6.2.5.2.5 C programming environment is identified.
- 6.2.5.2.6 Description of sub programming
- 6.2.5.2.7 C program format is explained.

### **6.2.5.3 Information Sheet**

- **C Programming Language**

#### **Overview of C programming language**

C is a structured programming language developed by Dennis Ritchie in 1973 at Bell Laboratories. It is one of the most popular computer languages today because of its structure, high-level abstraction, machine-independent feature, etc.

#### **History of C language: -**

C language has evolved from three different structured language ALGOL, BCPL, and B Language. It uses many concepts from these languages while introduced many new concepts such as datatypes, struct, pointer, etc. In 1988, the language was formalized by the **American**

**National Standard Institute** (ANSI). In 1990, a version of the C language was approved by the **International Standard Organisation** (ISO) and that version of C is also referred to as C89.

The idea behind creating C language was to create an easy language that requires a simple compiler and enables programmers to efficiently interact with the machine/system, just like machine instructions.

### **Characteristics of C language**

- C is a powerful, flexible language that provides fast program execution.
- **C is a *Procedural Language*** i.e. the programmer is required to provide step-by-step instructions for the CPU (central processing unit).
  - The success of C is due to its simplicity, efficiency, flexibility and small memory requirements.
  - *Low Level features*: C's power and fast program execution come from its ability to access low-level commands, similar to assembly language, but with high-level syntax.
  - *Portability*: C programs are portable i.e. they can run on any compiler with little or no modification. Compiler and Preprocessor make it possible for the C program to run it on different PC.
  - *Bit Manipulation*: C Programs can be manipulated using bits and it provides wide variety of bit manipulation operators
  - *Modular Programming*: It is a software design technique that increases the extent to which software is composed of separate parts, called modules. A C-program consists of different modules that are integrated to form a complete program.

- *Efficient Usage of Pointers*: C supports the efficient use of pointers and pointers have direct memory access.

### Structure of a C program

Preprocessor directives

Global declarations

Main()

{

Local declaration Statements

}

Function 1()

{

Local declaration statements

}

Function n()

{

Local declaration statements

}

Local declaration statements

easytvet.com

### Preprocessor directives

- Before a C program is compiled in a compiler, source code is processed by a program called pre-processor. This process is called pre-processing.
- Commands used in pre-processor are called preprocessor directives and they begin with the “#” symbol.

Below is the list of preprocessor directives that the C programming language offers.

Preprocessor                      Syntax/Description

## MACRO

Syntax: #define

This macro defines constant value and can be any of the basic data types.

HEADER                      FILE      Syntax: #include                      <file\_name>

INCLUSION                      The source code of the file “file\_name” is included in the main program at the specified place.

CONDITIONAL                      Syntax: #ifdef,                      #endif,                      #if,                      #else,                      #ifndef

COMPILATION                      Set of commands are included or excluded in the source program before compilation concerning the condition.

OTHER DIRECTIVES                      Syntax: #undef,                      #pragma

#undef is used to undefine a defined macro variable. #Pragma is used to call a function before and after the main function in a C program.

## Example

```
#include<stdio.h>
#include<conio.h>
int                      add                      (int                      a,                      int                      b);
void                      main()
{
    .....
}
```

- i. `#include<file>` variant used for system header files.(searches for a file named file in a list of directives.
- ii. `#include` –used to paste code of a given file into the current one.
- iii. `#define macros` –a segment of code used to replace the value of a macro.
- iv. `#undef` –used to cancel the definition of a macro.
- v. `#ifndef`-checks if a macro is not defined.
- vi. `#define const` –used to define a constant
- vii. `#include<math.h>` -a preprocessor directive used to add math libraries to a program.

## **Input and output statements**

### *Input*

-The process of giving something to the computer is known as input. Input is mostly given through the keyboard. Input functions are:

- `Scanf ()`
- `Gets ()`
- `Getch ()`
- `Getche ()`

### *Output*

The process of getting something from the computer is known as output. The output is mostly displayed on monitors. Output functions are:

- `print ()`
- `puts()`

The functions used for input and output are stored in the header file `stdio. h`. If the programmer uses any of the above function it is necessary to include the header file.

## **C keywords**

They are reserved words with predefine meaning to the compiler. They include:

**Table 44 C keywords**

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
const	extern	return	union
char	float	short	unsigned
continue	for	signed	volatile
default	goto	size	void
do	if	static	while

easyvet.com

## **Variables**

- When we want to store any information(data) on our computer/laptop, we store it in the computer's memory space.
- Instead of remembering the complex address of that memory space where we have stored our data, our operating system provides us with an option to create folders, name them, so that it becomes easier for us to find it and access it.
- Similarly, in C language, when we want to use some data value in our program, we can store it in a memory space and name the memory space so that it becomes easier to access it.
- The naming of an address is known as a **variable**. Variable is the name of a memory location.
- Unlike constant, variables are changeable, we can change the value of a variable during execution of a program. A programmer can choose a meaningful variable name.

- Example: Height, age, are the meaningful variables that represent the purpose it is being used for. Height variable can be used to store a height value. Age variable can be used to store the age of a person

### Rules to name a Variable

1. Variable name must not start with a digit.
2. The variable name can consist of alphabets, digits, and special symbols like underscore `_`.
3. Blank or spaces are not allowed in a variable name.
4. Keywords are not allowed as a variable name.
5. Upper- and lower-case names are treated as different, as C is case-sensitive, so it is suggested to keep the variable names in lower case

### Datatype of Variable

Data types specify how we enter data into our programs and what type of data we enter. C language has some predefined set of data types to handle various kinds of data that we can use in our program. These data types have different storage capacities.

There are four data types in the C language. They are, It can be:

**Table 45 C datatypes**

Types	Data Types
Basic data types	int, char, float, double
Enumeration data type	enum
Derived data type	pointer, array, structure, union
Void data type	void

### Integer data type:

- Integer data type allows a variable to store numeric values.
- The “int” keyword is used to refer to an integer data type.
- The storage size of the int data type is 2 or 4 or 8 bytes.
- It varies depending upon the processor in the CPU that we use. If we are using a 16-bit processor, 2 bytes (16 bit) of memory will be allocated for the int data type.
- Likewise, 4 byte (32 bit) of memory for 32 bit processor and 8 byte (64 bit) of memory for 64 bit processor is allocated for int datatype.
- int (2 byte) can store values from -32,768 to +32,767
- int (4 byte) can store values from -2,147,483,648 to +2,147,483,647.

If you want to use the integer value that crosses the above limit, you can go for “long int” and “long long int” for which the limits are very high.

### ***Character data type:***

Character data type allows a variable to store only one character.

Storage size of character data type is 1. We can store only one character using a character data type.

“char” keyword is used to refer to a character data type.

For example, ‘A’ can be stored using char datatype. You can’t store more than one character using a char data type.

Please refer [C – Strings](#) topic to know how to store more than one characters in a variable.

### ***Floating point data type:***

Floating point data type consists of 2 types. They are,

1. float
2. double

### ***Float:***

- Float data type allows a variable to store decimal values.



- Storage size of float data type is 4. This also varies depending upon the processor in the CPU as “int” data type.
- We can use up to 6 digits after decimal using float data type.
- For example, 10.456789 can be stored in a variable using a float data type.

***Double:***

- A double data type is also the same as float data type which allows up to 10 digits after the decimal.
- The range for double datatype is from 1E-37 to 1E+37.

**C operators and expressions**

-An operator is used to describe an operation applied to one or several objects.

- i. Arithmetic Operators
- ii. Increment and Decrement Operators
- iii. Assignment Operators
- iv. Relational Operators
- v. Logical Operators
- vi. Conditional operators
- vii. Bitwise Operators
- viii. Special Operators

***Arithmetic Operators***

<b><i>Operator</i></b>	<b><i>Meaning of Operator</i></b>
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division

%                                      Remainder after division (modulo division)

***Increment and Decrement Operators:***

<b><i>Operator</i></b>	<b><i>Meaning of Operator</i></b>
------------------------	-----------------------------------

++	Increment operator (unary)
----	----------------------------

a++; //post increment

++	a; //pre increment
----	--------------------

--	Decrement operator (unary)
----	----------------------------

a--; //post decrement

--a; //pre decrement

***Assignment Operators:***

There are two types of assignment operators.

1. Simple Assignment

2. Compound Assignment

<b><i>Operator</i></b>	<b><i>Meaning of Operator</i></b>
------------------------	-----------------------------------

***Simple Assignment***

=	Assignment Operator
---	---------------------

e.g. x=5

5 is assigned to x

***Compound Assignment***

+=	a += b is equivalent to a = a + b
----	-----------------------------------

-=	a -= b is equivalent to a = a - b
----	-----------------------------------

<code>*=</code>	<code>a *= b</code> is equivalent to <code>a = a * b</code>
<code>/=</code>	<code>a /= b</code> is equivalent to <code>a = a / b</code>
<code>%=</code>	<code>a %= b</code> is equivalent to <code>a = a % b</code>
<code>&amp;=</code>	<code>a &amp;= b</code> is equivalent to <code>a = a &amp; b</code>
<code> =</code>	<code>a  = b</code> is equivalent to <code>a = a   b</code>
<code>^=</code>	<code>a ^= b</code> is equivalent to <code>a = a ^ b</code>
<code>&lt;&lt;=</code>	<code>a &lt;&lt;= b</code> is equivalent to <code>a = a &lt;&lt; b</code>
<code>&gt;&gt;=</code>	<code>a &gt;&gt;= b</code> is equivalent to <code>a = a &gt;&gt; b</code>

### ***Relational Operators:***

Relational Operators are used to check the relationship between two operands. If the relation is

true, it returns value 1 and if the relation is false, it returns value 0. Relational operators are used

in decision making and loops in C programming.

<b>Operator</b>	<b>Meaning of Operator</b>
<code>==</code>	Equal to e.g. <code>5 == 3</code> returns false or 0
<code>&gt;</code>	Greater than e.g. <code>5 &gt; 3</code> returns true or 1
<code>&lt;</code>	Less than e.g. <code>5 &lt; 3</code> returns false or 0
<code>!=</code>	Not equal to

5 != 3 returns true or 1

>=

Greater than or equal to

e.g. 5 >= 3 returns true or 1

<=

Less than or equal to

e.g. 5 <= 3 returns false or 0

### ***Logical Operators:***

Logical operators are used to combining expressions containing relation operators. In C, there are 3

logical operators:

#### **Operator**

#### **Meaning of Operator**

&&

Logical AND

e.g. if c=5 and d=2 then

((c == 5) && (d > 5)) returns false or

0

||

Logical OR

e.g. if c=5 and d=2 then

((c == 5) || (d > 5)) returns true or 1

!

Logical NOT

e.g. if c=5 then

!(c == 5) returns false or 0

### ***Conditional Operator:***

The conditional operator takes three operands and consists of two symbols? and.  
Conditional

operators are used for decision making in C.

For example: `c = (c > 0) ? 10 : 20;`

If c is greater than 0, the value of c will be 10 but, if c is less than 0, the value of c will be 20.

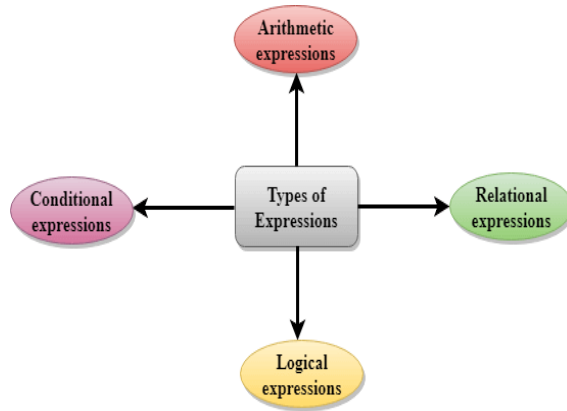
### **Bitwise Operators:**

A bitwise operator works on each bit of data.

<b>Operator</b>	<b>Meaning of Operator</b>
&	Bitwise AND
	Bitwise OR
^	Bitwise Exclusive OR
<<	Shift Left
>>	Shift Right
~	Bitwise Complement (One's Complement)

### **C expressions**

An expression is a formula in which operands are linked to each other by the use of operators to compute a value. An operand can be a function reference, a variable, an array element, or a constant.



**Figure 181: C expressions**

## Types of C Expressions

### Arithmetic Expressions

An arithmetic expression is an expression that consists of operands and arithmetic operators. An arithmetic expression computes a value of type int, float or double.

When an expression contains only integral operands, then it is known as pure integer expression when it contains only real operands, it is known as pure real expression, and when it contains both integral and real operands, it is known as mixed-mode expression

### Example

$$6*2/(2+1 * 2/3 + 6) + 8 * (8/4)$$

Evaluation of expression	Description of each operation
$6*2/(2+1 * 2/3 + 6)$ $+ 8 * (8/4)$	An expression is given.

$6*2/(2+2/3 + 6) + 8 * (8/4)$	2 is multiplied by 1, giving value 2.
$6*2/(2+0+6) + 8 * (8/4)$	2 is divided by 3, giving value 0.
$6*2/ 8+ 8 * (8/4)$	2 is added to 6, giving value 8.
$6*2/8 + 8 * 2$	8 is divided by 4, giving value 2.
$12/8 +8 * 2$	6 is multiplied by 2, giving the value 12.
$1 + 8 * 2$	12 is divided by 8, giving value 1.

### Relational Expressions

- A relational expression is an expression used to compare two operands.
- It is a condition that is used to decide whether the action should be taken or not.
- In relational expressions, a numeric value cannot be compared with the string value.
- The result of the relational expression can be either zero or non-zero value. Here, the zero value is equivalent to a false, and the non-zero value is equivalent to a true.

Relational Expression	Description
$x\%2 == 0$	This condition is used to check whether the x is an even number or not. The relational expression results in value 1 if x is an even number otherwise results in value 0.
$a!=b$	It is used to check whether a is not equal to b. This relational expression results in 1 if a is not equal to b otherwise 0.
$a+b == x+y$	It is used to check whether the expression "a+b" is equal to the expression "x+y".
$a>=9$	It is used to check whether the value of a is greater than or equal to 9.

### Logical Expressions

- A logical expression is an expression that computes either zero or non-zero value.
- It is a complex test condition to take a decision.

Let's see some examples of logical expressions.

Logical Expressions	Description
$( x > 4 ) \ \&\& \ ( x < 6 )$	It is a test condition to check whether the x is greater than 4 and x is less than 6. The result of the condition is true only when both conditions are true.
$x > 10 \    \ y < 11$	It is a test condition used to check whether x is greater than 10 or y is less than 11. The result of the test condition is true if either of the conditions holds value.
$! ( x > 10 ) \ \&\& \ ( y == 2 )$	It is a test condition used to check whether x is not greater than 10 and y is equal to 2. The result of the condition is true if both conditions are true.

### Conditional Expressions

- A conditional expression is an expression that returns 1 if the condition is true otherwise 0.
- A conditional operator is also known as a ternary operator.

### The Syntax of Conditional operator

Suppose **exp1**, **exp2** and **exp3** are three expressions.

$exp1 \ ? \ exp2 \ : \ ex$

### Control structures

Control structures are blocks of statements that determine how program statements are to be executed.



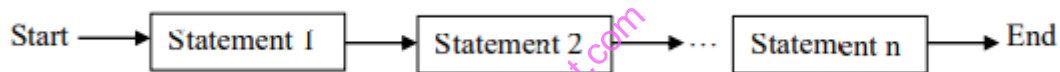
-Control statements deal with situations where processes are to be repeated several numbers of times or where decisions have to be made.

There are 3 control structures used in most of the structured programming languages:

- i. Sequence.
- ii. Selection.
- iii. Iteration (looping).

### i. Sequence control structures

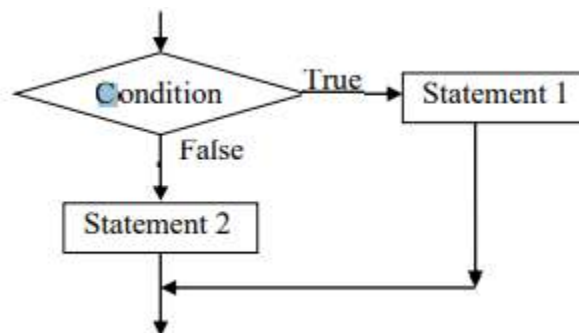
-In Sequence program execution, the program statements are executed one after another in the order in which they are written. In other words, the computer reads instructions from a program file line-by-line starting from the first line sequentially towards the end of the file.



**Figure 182: Sequence control structure**

### ii. selection control structures

-Selection involves choosing a specified group of instructions/statements for execution. In Selection control, a logical test is carried out, and one or more statements are usually selected.



**Figure 183: Selection control structure**

There are 4 types of selection control structures used in most high-level programming.

languages:

- a. IF – THEN
- b. IF – THEN – ELSE
- c. Nested IF
- d. CASE – OF

### **IF – THEN**

IF – THEN structure is used if only one option is available, i.e., it is used to perform a certain action if the condition is true but does nothing if the condition is false.

The general format of the IF-THEN structure is:

```
IF < Condition > THEN
```

```
Program statement to be executed if the condition is true;
```

```
ENDIF
```

If the condition is TRUE, the program executes the part following the keyword 'THEN'.

If the condition is FALSE, the statement part of the structure is ignored, and the program continues.

with the statements below the ENDIF.

### **Example**

In a school, the administration may decide to reward only those students who attain a mean mark of 80% and above.

Pseudocode

```
IF Mark > 80 THEN
```

```
Yes Print " Give reward"
```

```
ENDIF
```

### **IF – THEN -ELSE**

The IF-THEN-ELSE structure is suitable when there are 2 available options to select from.

The general format of the IF-THEN-ELSE structure is:

IF < Condition > THEN

Statement 1; (called the THEN part)

ELSE

Statement 2; (called the ELSE part)

ENDIF (indicates the end of the control structure)

NESTED IF

-Nested IF structure is used where 2 or more options have to be considered to make a selection.

The general format of the Nested IF structure is:

IF < Condition 1 > THEN

Statement 1

ELSE

IF < Condition 2 > THEN

Statement 2

ELSE

IF < Condition 3 > THEN

Statement 3

ELSE

Statement 4;

ENDIF

ENDIF

ENDIF

**The CASE structures**

-CASE-OF allows a particular group of statements to be chosen from several available groups.

-It is therefore used where the response to a question involves more than two choices/alternatives.

The general format of the CASE structure is:

```
CASE Expression OF
    Label 1: statement 1
    Label 2: statement 2
    Label 3: statement 3
    .
    .
    Label n: statement n
ELSE
    Statement m
ENDCASE
```

### iii. Iteration/Loop structures

-Looping refers to the repeated execution of the same sequence of statements to process individual data.

-The program is designed to execute the same group of statements repeatedly for a specified number of times or until a certain condition is met/satisfied.

The loop structure consists of 2 parts:

- 1). Loop body, which represents the statements to be repeated.
- 2). Loop control, which specifies the number of times the loop body is to be repeated.

Consider following the link below for further reading on Loop Structures

<https://youtu.be/qUPXsPtWGoY>

### Subprograms

-Also called sub procedures – a group of statements that together perform a given task. They are:

Procedures and functions

Scope of variables –coverage /extent of a variable reference in a program.

Variable scoping can be;

- *Global variables*
- *Local variables*

Parameter passing –the parameters can be passed in two ways

- *Call by value*
- *Call by reference*

## **C program format**

Structure of c language

The structure of the C program is as follows:

```
#include<stdio.h>
void main (void)
{
    print (“ ”);
    getch();
}
```

**#preprocessor**

This symbol directs the compiler that content on this line is already processed.

**include**

This directive indicates that the standard library object file has to be attached to the program.

**<stdio.h>**

This is a header file which contains instruction for the predefined functions of the language.

Standard input/output header file contains the instruction for getting and displaying data on a screen.

Types of Header files:

Header file in angle brackets is language define ( <header file> ) and header file in double-quotes is user define ( "header file" ).

Main Function  
This directs the compiler to start the compiling from the point.

Void main (void)  
The first void show we are not getting any value from the program. The second void in a bracket called argument or parameter shows we are not giving or passing any value to the program.

{ } Delimiters  
The body of the program is written in between these delimiters.

Opening brace "{" shows the start of the program.

Closing brace "}" shows the end of the program.

Body of program  
Body of the program consists of statements or a set of statements. Each statement ends with a semicolon (;) known as terminator which tells the compiler where one statement is ended.

Printf(" ")  
Printf() function is used to display content written in double-quotes on screen.

#### 6.2.5.4 Learning Activities

##### Practical activity

Using your personal computer, perform the following activity

A class trainer designed a simple program that would help her do the following:

- (a) Enter marks obtained in 4 subjects and calculate the average marks for each student.
- (b) Depending on the Average mark obtained, the program should assign grade as follows:

Between 80 and 100 – A

Between 70 and 79 – B

Between 60 and 69 – C

Between 50 and 59 – D

Below 50 – E

- (c) The program should then display the Average grade.

Using pseudocode and a flowchart, write an algorithm that shows the design of the program.

#### 6.2.5.6 Self-Assessment

You are provided with the following questions for self -assessment, attempt them and check your responses

1. What is C programming language?
2. Give five characteristics of C Programming Language
3. What do the following fundamentals mean in C programming language? Give two examples for each
  - a) C keywords
  - b) Variables
  - c) C operators
  - d) C expressions
4. Give the difference between Sequence and Selection control structures in C programming

5. A lady deposits 2,000 shillings in a Microfinance company at an interest rate of 20% per annum. At the end of each year, the interest earned is added to the deposit and the new amount becomes the deposit for that year.

Write pseudocode for a program that would track the growth of the deposits over seven years.

#### **6.2.5.7 Tools, Equipment, Supplies, and Materials**

- Flow charts
- Data flow diagram
- Decision table
- Decision tree
- Web Authoring tools
- Notepad
- Computer
- compiler
- Digital instructional material including DVDs and CDs

#### **6.2.5.7 References**

Blignaut, A. S., Hinojosa, J. E., Els, C. J., & Brun, M. (2010). ICT in education policy and practice in developing countries: South Africa and Chile compared through SITES 2006. *Computers & Education*, 55(4), 1552–1563.

*Computer Hardware*. (n.d.). Retrieved September 30, 2020, from <https://web.stanford.edu/class/cs101/hardware->

KIRIHATA, Y. (2009). *Information processing apparatus and method, computer-readable recording medium, and an external storage medium* (United States Patent No. US20090241114A1).

Lehmann, S., & Schweitzer, B. (2006). *Apparatus, system, and method for creating customized workflow documentation* (United States Patent No. US20060059423A1).



Nagar, T. (2019, December 3). *What is software and types of software with examples?* YourStory.Com. <https://yourstory.com/mystory/what-software-types-examples>

### 6.2.5.8 Model answers to self -assessment

1. Give five characteristics of C Programming Language

- *C is a powerful, flexible language that provides fast program execution.*
- *C is a Procedural Language i.e. the programmer is required to provide step-by-step instructions for the CPU (central processing unit).*
- *The success of C is due to its simplicity, efficiency, flexibility and small memory requirements.*
- *Low Level features: C's power and fast program execution come from its ability to access low-level commands, similar to assembly language, but with high-level syntax.*
- *Portability: C programs are portable i.e. they can run on any compiler with little or no modification. Compiler and Preprocessor make it possible for C program to run it on different PC.*
- *Bit Manipulation: C Programs can be manipulated using bits and it provides a wide variety of bit manipulation operators*
- *Modular Programming: It is a software design technique that increases the extent to which software is composed of separate parts, called modules. A c program consists of different modules that are integrated to form a complete program.*
- *Efficient Usage of Pointers: C supports the efficient use of pointers and pointers has direct memory access.*

2. What do the following fundamentals mean in C programming language? Give two examples for each

e) C keywords

--Keywords are predefined, reserved words used in programming that have special meanings to the compiler. **Keywords** are part of the syntax and they cannot be used as an identifier

f) Variables

- A variable is a named storage location **Variables** which is used to store information to be referenced and manipulated in a computer program. Example of variable include height, age etc

g) C operators

An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. Example of C operators include arithmetic operators, assignment operators, relational operators, logical operators etc

h) C expressions

- An expression is a formula in which operands are linked to each other by the use of operators to compute a value. An operand can be a function reference, a variable, an array element or a constant. Examples include arithmetic expressions, logical expressions, conditional operators etc

3. Give the difference between Sequence and Selection control structures in C programming

- In Sequence program execution, the program statements are executed one after another in the order in which they are written while in Selection control, a logical test is carried out, and one or more statements are usually selected

for execution depending on whether the condition given is True or False.

4. A lady deposits 2,000 shillings in a Microfinance company at an interest rate of 20% per annum. At the end of each year, the interest earned is added to the deposit and the new amount becomes the deposit for that year.

Write pseudocode for a program that would track the growth of the deposits over seven years.

*START NPUT Initial Deposit*

*INPUT Interest Rate*

*SET Deposit to Initial deposit (i.e., 2000)*

*SET Year to 0*

*WHILE Year <= 7 DO*

*Interest = Deposit x Interest rate*

*Total = Deposit + Interest*

*Deposit = Total {the new deposit}*

*Year = Year + 1*

*ENDWHILE*

*PRINT Deposit, Year*

*STOP*

*easytvvet.com*

## 6.2.6 Learning Outcome 6: Perform Basic Internet programming

### 6.2.6.1 Introduction to the learning outcome

This learning outcome specifies the content of competencies required to perform basic internet programming internet-based programming concepts, web programming approaches, web programming languages, web programming interfaces, and HTML coding.

### 6.2.6.2 Performance Standard

- 6.2.6.2.1 Internet-based programming concepts are identified.
- 6.2.6.2.2 Web programming approaches are identified.
- 6.2.6.2.3 Web programming languages are identified.
- 6.2.6.2.4 Web programming interfaces are identified.
- 6.2.6.2.5 HTML coding is done.

### 6.2.6.3 Information Sheet

#### 1. Basic internet programming

#### Concepts of internet programming

- Network concepts
- Web concepts
- Internet addresses
- Sockets
- Programming network applications

#### Network concepts

A network is in this respect a collection of interconnected computers and/or other kinds of equipment Terminology:

- **Inode**, a machine that is connected to the network (computer, printer, bridge, vending machine)
- **host**, a fully autonomous computer connected to the network

- **address**, each node has a unique address (several bytes)
- **packet**, modern networks are packet-based, meaning that the information is broken down to and sent as small chunks, each chunk of information handled separately.
- **protocol**, rules, specifying how to perform communication.

## Web services

### *Web Application (Webapp)*

A web application (or web app), unlike standalone application, runs over the Internet. Examples of web apps are google, amazon, eBay, Facebook and the UCT website. A web app is typically a 3-tier (or multi-tier) client-server database application run over the Internet and it comprises five components:

### Basic Concepts

- **HTTP Server:** E.g., Apache HTTP Server, Apache Tomcat Server, Microsoft Internet Information Server (IIS), Nginx, Google Web Server (GWS), and others. You will learn how to install Apache HTTP and Tomcat web servers in the next chapter.
- **HTTP Client (or Web Browser):** E.g., Internet Explorer (MSIE), Firefox, Chrome, Safari, and others.
- **Database:** E.g., Open-source MySQL, MariaDB, Apache Derby, MySQL, SQLite, PostgreSQL, OpenOffice's Base; Commercial Oracle, IBM DB2, SAP Sybase, MS SQL Server, MS Access; and others.
- **Client-Side Programs:** could be written in HTML Form, JavaScript, VBScript, Flash, and others.
- **Server-Side Programs:** could be written in Java Servlet/JSP, ASP, PHP, Perl, Python, CGI, and others.

## Programming network applications

-Alongside the technical "evolution", communication between the application and also between parts of applications residing on a different computer become more and more common

Examples of asynchronously communicating applications: web browsers, e-mail, news.

Some other examples: Distributed databases, sound, radio, video and internet telephony.

Need for applications where the participants are aware of each other:

Shared bulletin boards, whiteboards, shared word processors, control systems (eg. robots), and (not the least) games (like RuneScape and world of warcraft).

There is support in the networks, where we will look closer on the internet

Kinds/types of application programs

- E-mail
- News
- Web-based databases
- Client-server,
- per-to-peer
- Telephone
- Video

easyvet.com

### **Internet addresses**

-Internet is the most knowledgeable and most widespread network.

Designed it to be robust (errors are unusual)

First version 1969, ARPANET, designed by ARPA, a DoD unit.

In 1983 there were 562 computers on the ARPANET

In 1986 there were 5000 computers

In 1987 – 28000,

In 1989 – 100000,

In 1990 – 300000,

In 2009 – 1.67 billion (a rough estimate on June 30)Layers

A network is built as a set of layers

IP, Internet Protocol- the network layer protocol (the reason for the name "Internet")

TCP, Transport Control Protocol-a connection-based protocol which ensures correct data exchange between two nodes

UDP, User Datagram Protocol-a protocol which allows the transmission of independent packets from one node to another with no guarantee concerning delivery or order of delivery

## **Sockets**

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to. An endpoint is a combination of an IP address and a port number.

## **2. Web programming approaches**

### **✓ Server-side**

The Server-side is the systems that run on the server, and the client-side is the software that runs on a user's web browser. The basic hosting of your files is achieved through a web server whose responsibilities are described below.

Server-side development is much more than web hosting: it involves the use of a programming technology like PHP or ASP.NET to create scripts that dynamically generate content.

It is important to remember that when developing server-side scripts, you are writing software, just like a C or Java programmer would do, with the major distinction that your software runs on a web server and uses the HTTP request-response loop for most interactions with the clients. This distinction is significant, since it invalidates many classic software development patterns, and requires different thinking for many seemingly simple software principles like data storage and memory management.

### **✓ Client-side**

Client-side web development involves interactivity and displaying data, server-side is about working behind the scenes to manage data. The idea of client-side scripting is an important one in web development. It refers to the client machine (i.e., the browser) running code locally rather than relying on the server to execute code and return the result. Many client-side languages have come into use over the past decade including Flash, VBScript, Java, and JavaScript. Some of these technologies only work in certain browsers, while others require plug-ins to function.

## Web programming languages

### HTML

-HTML (HyperText Markup Language) a *language* used to develop web pages.

**Hypertext** means that some text in the HTML document carries a link to a different location, which can be on the same page or another page. On clicking this 'hot spot', the viewer is transferred to that location.

**Markup** means that specific portions of a document are *marked up* to indicate how they should be displayed in the browser.

-HTML simply consists of tags that are placed around elements, which then changes the properties of these enclosed elements. There are hundreds of HTML tags and some of these are proprietary, which means that only some browsers recognize them.

```
<html>
  <head>
    <title>Internet programming</title>

  </head>
  <body>
</body>
</html>
```

For further reading on HTML,click on the link below

<https://youtu.be/UB1O30fR-EE>

### PHP



Open source

PHP not only carries all the goodness of ASP but also is more secure and handles databases more easily. It is a known fact that **PHP on Apache Web server runs faster than ASP**. PHP code is embedded inside the HTML page and can link to databases to generate dynamic HTML content. PHP scripts can be made to run on any operating system with little or no modification.

### **Pros**

- Can be embedded in HTML; seen as a dedicated web language.
- Deceptively simple to learn.
- Touted as a good beginner language.
- Large array of built-in functions for everything from PDF creation to credit-card transactions to database interaction
- Available at a majority of hosting services
- Relatively good integration with apache.

### **Cons**

- While the runtime services are good, the language itself is quite horrible
- Encourages writing bad code ^2 {too vague}'
- PHP implementation is full of bugs and might change semantics without a note, causing constant security upgrades that break your code
- PHP encourages writing web programs in an insecure manner
- Not too fast (good for building database interfaces, not making calculations)

## **JAVASCRIPT**

Javascript is a programming language that runs on a web browser. Jscript is Microsoft's' implementation of Javascript for Internet Explorer. *Javascript is not a subset of Java*, in fact, the two languages share little in common. Javascript runs on the browser (client) and does not require any server software. Thus, it is a client-side scripting language. Since all execution takes place on the browser, Javascript is responsible for most of the interactivity on a web page. Image change or text color change on mouse-over, creating mouse trails are all

possible through Javascript. The language has also been widely used for basic form validation.

Javascript is commonly embedded inside the HTML page and is thus visible to the visitor.

Javascript can also be written to run on a server and this is based on the ASP model promoted by Microsoft.

#### **Pros**

- Most web developers familiar with language from the client-side.

#### **Cons**

- Server-side standards not settled

Consider the following video links for further reading on Javascript

<https://youtu.be/uDwSnnh11Ng>

<https://youtu.be/2nZiB1JItbY>

### **3. Web Programming Interfaces**

#### **Common client interface**

CCI defines is a set of interfaces and classes whose methods allow a client to perform typical data access operations. The following CCI interfaces and classes illustrate how to use the CCI

- **ConnectionFactory:** Provides an application component with a Connection instance to an EIS.
- **Connection-** Represents the connection to the underlying EIS.
- **ConnectionSpec:** Provides a means for an application component to pass connection request-specific properties to the ConnectionFactory when making a connection request.
- **Interaction:** Provides a means for an application component to execute EIS functions, such as database stored procedures.
- **InteractionSpec:** Holds properties about an application component's Interaction with an EIS.
- **Record:** The superclass for the different kinds of record instances. Record instances may be Mapped Record, Indexed Record, or ResultSet instances, which all inherit from the Record interface.
- **Record Factory:** Provides an application component with a Record instance.

- Indexed Record: Represents an ordered collection of Record instances based on the java.util.List interface.

A client or application component that uses the CCI to interact with an underlying EIS does so in a prescribed manner. The component must establish a connection to the EIS's resource manager, and it does so using the ConnectionFactory. The Connection object represents the actual connection to the EIS and it is used for subsequent interactions with the EIS.

### **Common gateway interface**

As you traverse the vast frontier of the World Wide Web, you will come across documents that make you wonder, "How did they do this?" These documents could consist of, among other things, forms that ask for feedback or registration information, imagemaps that allow you to click on various parts of the image, counters that display the number of users that accessed the document, and utilities that allow you to search databases for particular information. In most cases, you'll find that these effects were achieved using the Common Gateway Interface, commonly known as CGI. CGI is the part of the Web server that can communicate with other programs running on the server. With CGI, the Web server can call up a program, while passing user-specific data to the program (such as what host the user is connecting from, or input the user has supplied using HTML form syntax). The program then processes that data and the server passes the program's response back to the Web browser.

## **HTML**

### **Tags**

An HTML element is identified in the HTML document by tags. A tag consists of the element name within angle brackets. The element name appears in both the beginning tag and the closing tag, which contains a forward slash followed by the element's name, again all enclosed within angle brackets. The closing tag acts like an off-switch for the on-switch that is the start tag

- ✓ Parcelling
- ✓ Coding

### **6.2.6.4 Learning Activities**

#### **Practical activity**

#### **Instructions**

In your institutions' computer lab perform the following practicals

- i. Write a basic structure of an HTML document and give a complete explanation of each of the tags used
- ii. Create your first web page which will print the text “Hello World!” on the screen.

#### **6.2.6.5 Self-Assessment**

1. What is a web programming language?
2. Give the meaning of the following Network Concepts.
  - i. Node
  - ii. Host
  - iii. Packet
  - iv. Protocol
3. What is the role of web programming language?
4. What is the difference between Client-side and Server-side web programming approaches?

#### **6.2.6.6 Tools, Equipment, Supplies and Materials**

- Flow charts
- Data flow diagram
- Decision table
- Decision tree
- Web Authoring tools
- Notepad
- Computer
- Software
- Digital instructional material including DVDs and CDs

#### **6.2.6.7 References**

Alzahrani, A. A. (2020). 4GL Code Generation: A Systematic Review. (*IJACSA*) *International Journal of Advanced Computer Science and Applications*. 6(11). Retrieved from [https://www.researchgate.net/profile/Abdullah\\_Alzahrani13/publication/342652608](https://www.researchgate.net/profile/Abdullah_Alzahrani13/publication/342652608)

[\\_4GL\\_Code\\_Generation\\_A\\_Systematic\\_Review/links/5f48e73892851c6cfd046e6/4GL-Code-Generation-A-Systematic-Review.pdf](#).

Connolly, R. (2015). *Fundamentals of web development*. Pearson Education.

Kumar, H. (2019). Programming software and Computer Languages. *International journal of advance research and innovative ideas in education*. 3(5).

Selby, C. (2011). Four approaches to teaching programming. Retrieved from

Shaydulin, R., & Sybrandt, J. (2017). To agile, or not too agile: A comparison of software development methodologies. *arXiv preprint arXiv:1704.07469*.

### 6.2.6.7 Model answers to self-assessment

1. What is web programming language

- *The main difference between server-side scripting and client-side scripting is that the server side scripting involves server for its processing. On the other hand, client-side scripting requires browsers to run the scripts on the client machine but does not interact with the server while processing the client-side scripts.*

2. Give the meaning of the following Network Concepts

- i. **Node**, - *a machine that is connected to the network (computer, printer, bridge, vending machine)*
- ii. **Host** - *a fully autonomous computer connected to the network*
- iii. **packet**, - *information that is broken down to and sent as small chunks, each chunk of information handled separately.*
- iv. **Protocol** - *rules, specifying how to perform communication in a networked environment*

3. What is the role of web programming language

--*The capabilities of the Internet have been enhanced and extended by using programming languages with HTML. These languages have been responsible for the dynamic and interactive nature of the Net. New languages and language extensions are being developed to increase the usability of the Internet*

4. Give the difference between Client-side and Server-side web programming approaches

--*Client-side scripting generally refers to the class of computer programs on the web that are executed client-side, by the user's web browser. Client-side scripts are often*

*embedded within an HTML or XHTML document (hence known as an "embedded script"), but they may also be contained in a separate file, to which the document (or documents) that use it make reference (hence known as an "external script") while -server-side scripting are executed by the web server when the user requests a document. They produce output in a format understandable by web browsers (usually HTML), which is then sent to the user's computer.*

*-The user cannot see the script's source code (unless the author publishes the code separately), and may not even be aware that a script was executed. Documents produced by server-side scripts may, in turn, contain client-side scripts.*

5. Give a brief description of the following web programming interfaces
  - i. Common client Interface(CCI)
  - ii. Common Gateway Interface (CGI)

*easyvet.com*