# Introduction to Technical Programming

# Lecturer Guide

# Module 1: Computer hardware and software

1.1 Different types of hardware of a common system

After you have completed this module, you should be able to :

* Identify the components of the system unit
* Explain the term CPU and its purpose
* Describe the term CPU and explain the impact of using various different types of CPU’s
* Describe different types of CPUs with regard to use, and power
* Define the term computer memory
* Discuss the primary purpose of memory
* Differentiate between different types of memory and their purpose (Range: RAM, ROM, CMOS, Cache memory, Flash Memory)
* Explain how data is stored on memory
* Define the purpose of the motherboard and its components
* Describe different types of input hardware with regard to use and classification e.g. direct and indirect entry
* Discuss how data is transferred between memory i.e. primary and secondary and the CPU
* Describe the Flow/transfer of data between components. (Range: USB – PnP, U3, Point-to-point connections)
* Describe the factors to consider when choosing an input device. (Range: Ergonomic considerations, Wireless vs cables)
* Describe different types of output hardware
* Describe the purpose and use of devices such as docking stations for mobile and laptop computers.
* Describe the term software
* Describe the purpose and function of software
* Describe the basic concepts of software. (Range: Software as programs. Identify software components. Concept of a graphical user interface (GUI).)
* Contrast: System software vs application software
* Differentiate between: Shareware, Freeware, Open-Source Software and Proprietary software, Firmware
* Discuss the process of how software is obtained and installed.
* Differentiate between online software and installed software.
* Discuss the following terms in relation to software. (Range: Compatibility issues, Versions, patches and service packs, Updating software)
* Launch a new Linux terminal on the Raspberry Pi
* Use the man command to get help
* Expand a Linux file path and explain each element
* List the contents of the current folder using the ls command
* Change directly location using the cd command
* Create a new folder using the mkdir command
* Remove a folder using the rmdir command
* Remove a file using the rm command
* Rename a file using the mv command
* Copy a file using the cp command
* Clear the command prompts screen using the cls command
* Run an executable file from the command line

# FORMATIVE ASSESSMENT 1.1 INDIVIDUAL TASK

1.1.1 A system unit is the part of a computer that houses the primary devices that perform operations and produce results for complex calculations. (2)

1.1.2 The purpose of the CPU is to carry out the set of instructions given to the processor from a program. (2)

1.1.3 Information Processing Cycle Stages

1. Fetch- Instruction from Memory (Instruction Fetch, IF)
2. Decode the instructions into binary (Instruction Decode, ID)
3. Execute action and move to next step or calculate address (EXE)
4. Access memory operand (MEM)
5. Write back result to register (WB) (5)

1.1.4 The rate at which one operation is completed in a second is measured in **hertz**. (1)

1.1.5 THREE main components of the CPU.

* Control Unit
* Arithmetic Logic Unit
* Registers (3)

1.1.6 Differences between RISC and CISC

* In RISC, the instruction set is reduced, and most of these instructions are very primitive, while in CISC, the instruction set is very large that can be used for complex operations.
* RISC computer’s execution time is very less, whereas CISC computer’s execution time is very high.
* In RISC, the decoding of instructions is simple, whereas, in CISC, the decoding of instructions is complex. (6)

1.1.7 FOUR functions of a computer

1. **Data processing-** Data can take many forms, and the processing requirements are numerous.
2. **Data storage-** Even if the computer is processing data on the fly, it must temporarily store at least those pieces of data that are currently being worked on.
3. **Data movement-** The operating environment of a computer is made up of devices that act as data sources or destinations.
4. **Control-** A control unit manages the computer's resources and orchestrates the performance of its functional parts in response to commands.

1.1.8 The memory hierarchy is an improvement of computer storage into a hierarchy-based modal on response time. (2)

1.1.9 Static Random Access Memory and Dynamic Random Access Memory. (2)

1.1.10 Secondary memory is computer memory that is non-volatile and persistent in nature and is not directly accessed by a computer/processor. (2)

1.1.11 Hardware Devices

|  |  |  |
| --- | --- | --- |
| **Input devices** | **Processing devices** | **Output Devices** |
| Mouse | Motherboard | Monitor/Screen |
| Keyboard | Arithmetic Logic Unit | Printer |
| Light pen | Processor | Plotter |

**(8)**

**Total:42 Marks**

# FORMATIVE ASSESSMENT 1.2 INDIVIDUAL TASK

1.2.1 Software – set of instructions, data or programs used to operate computers and execute specific tasks. (2)

1.2.2 THREE types of language translators (3)

Translators

Compilers

Assemblers

1.2.3 Compiler-A compiler is a computer program that transforms whole code written in a high-level programming language at once into the machine code whereas an interpreter is a computer program, which converts each high-level program statement into the machine code line by line. (4)

1.2.4

**Freeware-** Freeware is the software that is available to use for free of cost without any limitations.

**Middleware**- is software that is used to bridge the gap between applications and other tools or databases. Some examples of middleware activities include handling data and [API](https://www.techtarget.com/searchapparchitecture/definition/application-program-interface-API) management, authentication and messaging services.

**Shareware**- The software is copyrighted and distributed for free only for testing purposes. After the trial period ends, you must pay.

**Open source-** This is provided for use, modification, and redistribution. Open-source software is downloaded from the internet at no cost.

(8)

1.2.5 **FOUR** categories of system software.

* Operating System
* Device Driver
* Language Translator
* Utility Programs

1.2.6 Benefits of online software

1. No software to install- Software is installed from the cloud. Once users have a license, it is easy to install. In addition, you can always retrieve the licence if lost.
2. Complete flexibility on device type-Users do not need to be very careful with regards to storing the software.
3. No maintenance required. - The responsibility is taken care of by the providers of the software or those who manage the cloud storage.
4. Connect from anywhere at any time with internet- users can install the software from any geographical location and any time of the day.
5. Centralised storage offering safe and secure data (10)

1.2.7 Compatibility is the capacity for two systems to work together without having to be altered to do so. (2)

**Total:29 Marks**

# FORMATIVE ASSESSMENT 1.3 INDIVIDUAL TASK

1.3.1 To launch Linux terminal on Raspberry Pi, you need to do the following steps:

Step 1: From the Raspberry Pi icon, select Accessories

Step 2: From the Accessories, click terminal (3) 1.3.2 ls- is used to list all files and directories in a file path on the Raspberry terminal. (1)

1.3.3 Raspberry Pi terminal commands

pi@raspberrypi:~/$cd Desktop

pi@raspberrypi:~Desktop $ mkdir terminal\_commands

pi@raspberrypi:~/Desktop $ cd terminal\_commands

pi@raspberrypi:~/Desktop/terminal\_commands $ pwd

/home/pi/Desktop/terminal\_commands

pi@raspberrypi:~/Desktop/terminal\_commands $

pi@raspberrypi:~/Desktop/terminal\_commands $ touch summative.docx

(6)

1.3.4 Adding text into the file called summative.docx

pi@raspberrypi:~/Desktop/terminal\_commands $echo “Hello fellow programmers”>> summative.docx

(3)

1.3.5 Merging contents of files.

pi@raspberrypi:~/Desktop/terminal\_commands $

pi@raspberrypi:~/Desktop/terminal\_commands$touch formative.docx

$ echo "Terminal commands are areasy">>formative.docx

$ cat formative.docx summative.docx >> final.docx

$ tac final.docx

(8)

1.3.6 Pairing the commands to the use

|  |  |
| --- | --- |
| **Command** | **Function** |
| rm | This command is used to delete files within a directory. |
| rmdir | To permanently delete an empty directory. |
| man | Is used to know more about a command and how to use it |
| mkdir | Used to create a new directory |

**(4)**

**Total: 25 Marks**

# SUMMATIVE ASSESSMENT 1.4 INDIVIDUAL TASK

1.4.1 The difference is that a Memory buffer register (MBR) contains a word to be stored in memory or sent to the I/O unit or is used to receive a word from memory or from the I/O unit. whereas Memory address register (MAR) specifies the address in memory of the word to be written from or read into the MBR. (4)

1.4.2SRAM: is a memory chip that is faster and uses less power than DRAM. whereas DRAM is a memory chip that can hold more data than an SRAM chip, but it requires more power. (4)

1.4.3 FIVE factors to consider when choosing an input device.

1. **User Needs**- this is the urgency of use of the device in the computer room by users also whether it will satisfy the needs of the user.
2. **Initial cost-**the amount it can cost when buying/purchasing the devices
3. **Maintenance Cost**-the amount that can be used to maintain the servicing of these devises should be considered.
4. **Mode Of Transmission**- how will you transport them to the computer room should be considered.
5. **Compatibility With Available Hardware**- will the devices fit to other devices in the room already. A device that fits your needs but doesn't work with your computer is useless.
6. **User- Friendliness**- whether the devices will be used to solve problems and easy to be used by users (10)

1.4.4 Types of ROM

1. **MROM (Masked Read Only Memory)**
2. **PROM (Programmable Read Only Memory)**
3. **EPROM (Erasable and Programmable Read Only Memory)**
4. **EEPROM (Electrically Erasable and Programmable Read Only Memory)** **(4)**

**1.4.5 Differences between microcontroller and microprocessor**

1. **Microprocessors only have a Central Processing Unit, whereas Micro Controllers have a CPU, Memory, and I/O all integrated into a single chip.**
2. **Microprocessors are used mainly in personal computers, whereas microcontrollers are used in embedded systems.**
3. **Microprocessors are based on the Von Neumann model while microcontrollers are based on the Harvard architecture.** (6)

1.4.6 Proprietary software is computer software where the source codes are publicly not available only the company that has created can modify it.  (2)

1.4.7 Forward compatibility is a design principle in which a program or piece of hardware is designed to work with new software or devices in the future. (1)

1.4.8 False (1)

1.4.9 The terminal will print the message in quotation marks and repeat the process id twice. Example: The process id is 590590 (2)

1.4.10 Working with terminal commands

pi@raspberrypi:~/$cd Desktop

pi@raspberrypi:~Desktop $ mkdir Practical

pi@raspberrypi:~/$cd Practical

pi@raspberrypi:~Desktop/Practical $ echo "It is the control unit that determines which machine instruction is to be executed next" >> next\_file.txt

pi@raspberrypi:~Desktop/Practical $ echo " The control unit is the brains of the computer " >> next\_file.tx

pi@raspberrypi:~Desktop/Practical $ pwd

(6)

**Total : 40 marks**

# PRACTICAL ACTIVITY PAIR WORK

**Software Installation**

Search for Anydesk and install the software on your machine. You can go to this link: <https://anydesk.com/en/downloads/thank-you?dv=win_exe>

Share the pin numbers of the software with your friend and try controlling each other’s machine.

The software allows remote control of one’s PC. This is quite helpful if you are working with a friend or classmate, and he/she gets stuck but you are in a different geographical location. Also, if your friend or colleague is not able to follow the instructions, you can take full control of his or her computer and start assisting remotely.

# Module 2 Problem solving in computer programming

After you have completed this module, you should be able to :

* Define the term problem solving
* Define the term computational thinking
* Describe the phases of the PLDC (Program Development Life Cycle)
* Describe the purpose of problem solving leading to solutions
* Explain and apply various problem-solving steps.
* State in own words Clarity on what needs to be done
* What is known or given?
* What is missing or needed?
* Devise a plan/algorithm (storyboard – visual or
* textual)
* Look for patterns
* Look at related problems, known solutions
* Examine simpler or special cases
* Make a table, create diagram, use guess and check,
* work backwards, identify sub-goal
* Carry out the plan/implement the algorithm (write
* the code)
* Look back/test (see if it works)
* Check results against original problem. Does it make sense? Is there another solution?)
* Use appropriate tools and techniques to present a solution. Range:
* User stories (written by the client and provide the requirements)
* Noun-verb analysis of user stories
* List of nouns provides identification of objects and state
* List of verbs provides identification of behaviour
* Acceptance tests (does the program meet the
* requirements?)
* Define the term algorithm and its purpose in the problem-solving process. (Range: Basic concepts of an algorithm.
* What is an algorithm? Develop a clear understanding of the problem presented.)
* Implement and understand the basic algorithmic constructs used to create a flowchart. Range: Input, Output, Processing and Calculations, Selection, Iteration
* Create a flowchart to present a particular algorithm and its associated tasks
* Interpret a basic flow chart and describe its intended operation / function

# FORMATIVE ASSESSMENT 2.1 INDIVIDUAL TASK

2.1.1 Problem solving is the sequential process of analysing information related to a given situation and generating appropriate response options. (2)

2.1.2 Computational thinking is an interrelated set of skills and practices for solving complex problems, a way to learn topics in many disciplines, and a necessity for fully participating in a computational world. . (2)

2.1.3 **FOUR** cornerstones of computational thinking are:

* **decomposition**
* **pattern recognition**
* **abstraction**
* **algorithms**  (4)

2.1.4 Program Development Lifecyle

(6)

2.1.7 SIX steps which must followed in problem solving.

* 1. Understand the Problem
  2. Formulate a Model
  3. Develop an Algorithm
  4. Write the Program
  5. Test the Program
  6. Evaluate the Solution

(6)

2.1.8 Write an algorithm that reads three numbers and prints the value of the largest number. (10)

2.1.8 Algorithm

Step1: Start

Step2: Read/input A,B and C

Step3: If (A>=B) and (A>=C) then Max=A

Step4: If (B>=A) and (B>=C) then Max=B

Step5:If (C>=A) and (C>=B) then Max=C

Step6: Print Max

Step7: End

(10)

2.1.9 Algorithm is the sequence of steps to be performed in order to solve a problem by the computer. (2)

**Total 32 Marks**

# FORMATIVE ASSESSMENT 2.2 INDIVIDUAL TASK

2.2.1 FOUR testing phases stages of program development lifecycle

*Software Testing* is a method to check whether the actual software product matches expected requirements and to ensure that software product is[defect](https://www.guru99.com/defect-management-process.html)free.

* **Integration testing**- The goal of integration testing is to test the interfaces between modules and expose any defects that may arise when these components are integrated and must interact with one another.
* **Unit testing**- Unit testing is typically performed throughout the application development process, with the goal of ensuring that every single unit or component works as expected.
* **Acceptance Testing**- It is a type of testing done to ensure that the users' requirements are met prior to delivery and that the software works properly in the user's working environment.
* **System Testing :** **Complete and integrated software’s are tested in system testing, which means that all of the system elements forming the system are tested as a whole to meet the system's requirements.**
* **Acceptance testing** is a test used to determine whether or not the requirements of a specification or contract have been met.
* **Regression testing**- This determines if adding additional features results in a decrease in an application's functionality.

2.2.2 Flowchart



(10)

2.2.3 SIX characteristics of a good algorithm.

* **Input-must have 0 or more inputs**
* **Output** − should have 1 or more well-defined outputs.
* **Finiteness**-must terminate after several steps
* **Feasibility**-should be feasible with available resources
* **Independent**-must have step by step directions independent from other programs (6)

2.2.4 Average of 25 scores.

A picture containing diagram

Description automatically generated

2.2.5 Algorithm to calculate area of Circle.

**Step1**: Start

**Step 2**:Input the Radius r of the Circle

**Step3**: Area PI\*r\*r // calculation of area

**Step4**: Print Area

**Step 5**: Stop

(6)

2.2.6 SIX benefits of using flowcharts.

* **Communication:** Flowcharts are a better way to communicate the logic of a system to all parties involved.
* **Effective analysis:** A flowchart can help you analyse a problem more effectively.
* **Proper documentation:** Program flowcharts are useful for program documentation, which is required for a variety of reasons.
* **Efficient Coding:** During the systems analysis and program development phases, the flowcharts serve as a guide or blueprint.
* **Proper Debugging:** The flowchart aids in the debugging process.
* **Efficient Program Maintenance:** The use of a flowchart simplifies program maintenance. It allows the programmer to focus his or her efforts more effectively on that aspect. (12)

**Total : 66 Marks**

# SUMMATIVE ACTIVITY 2.3 INDIVIDUAL TASK

2.3.1 **FOUR** cornerstones of computational thinking are:

* **decomposition** - breaking down a complex problem or system into smaller, more manageable parts
* **pattern recognition** –similarities among and within problems
* **abstraction** – focusing on the important information only, ignoring irrelevant detail
* **algorithms** - developing a step-by-step solution to the problem, or the rules to follow to solve the problem (8)

2.3.2 A conditional flowchart is a design technique used when a condition is imposed on a problem. (2)

2.3.3 Flowchart for Fahrenheit to Celsius

Graphical user interface, application

Description automatically generated

**(5)**

2.3.4 Quadratic equation

Algorithm (7)

Step 1: Start

Step 2: Input a, b, c

Step 3: Calculate d. d = sqrt(b x b – 4 x a x c

Step 4: Calculate x1. x1 = (–b + d) / (2 x a)

Step 5: Calculate x2. x2 =(–b – d) / (2 x a)

Step 6: Print x1 and x2

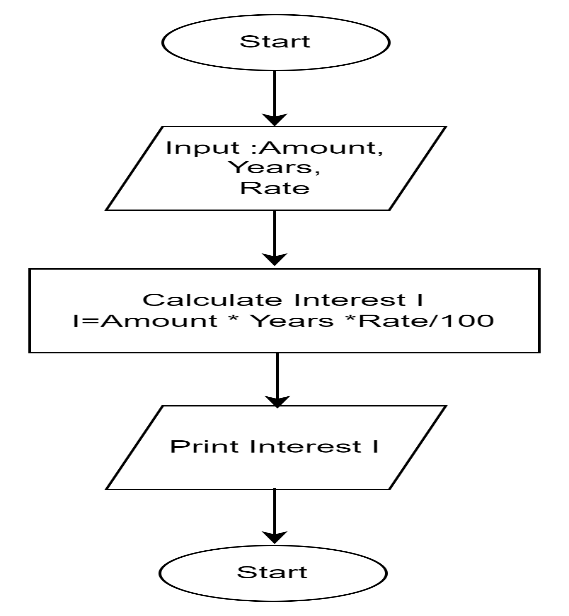
Step 7: Stop

**Flowchart**  (7)

Qr code

Description automatically generated with medium confidence

2.3.5 Computing Interest. **I=Amount \* Years \*Rate/100**

 (7)

2.2.6 THREE limitations of flowcharts

* **Complex logic:** The program logic can be quite complicated at times and in such instances, flowchart become more complex and clumsier.
* **Alterations and Modifications:** If changes are required, the flowchart may need to be completely redrawn.
* **Reproduction:** Because flowchart symbols cannot be typed, reproduction of flowcharts is difficult. (6)

* + 1. The symbol denotes \_\_\_\_\_\_\_

1. I/O
2. Flow
3. **Process**
4. Decision

2.2.8 A box that can represent two different conditions.  
a) Rectangle  
**b) Diamond**  
c) Circle  
d) Parallelogram

2.2.9 In computer science, algorithm refers to a pictorial representation of a flowchart.  
a) True  
**b) False**

2.2.10 The operation represented by parallelograms is \_\_\_\_\_\_\_\_.  
**a) Input/Output**  
b) Assignment  
c) Comparison  
d) Conditions

**Total :46 Marks**

# Module 3:Concepts of programming for single board microprocessor or microcontrollers

After you have completed this module, you should be able to :

**3.1 Introduction to IO on single board computing**

* Exploring the Arduino board
* Expand the term GPIO
* Compare and contrast some of the major advantages of python compared to other programming languages.
* Explain and identify where the GPIO pins are located
* Differentiate between a compiler and an interpreter
* Explain the purpose of the GPIO pins
* Discuss the major characteristics of the python programming language as an interpreted one
* Define the term physical computing
* Differentiate between a shell and an IDE
* Read and interpret a Pi GPIO Pin guide
* Define the term physical computing
* Explain what the terms 3V3, 5V, GND GP2 means on the GPIO board
* Discuss how Arduino is used to enable physical computing

**3.2 Visual Programming and solution development**

* Construct (code/write) using the visual tool, debug and run simple programs incorporating: Declaration of variables of different types, use and assignment of values to variables, incorporating program constructs with sequence, selection and iteration structures. Expose and apply various programming concepts as part of the coded solution such as:
* retrieving remainders: modulus
* differentiate between real value division and integer division
* comparison operators and performing logical comparisons
* incorporate and write code constructs to perform basic calculations such as area, volume, VAT and simple formulae, typical calculations done in other subjects
* include conditional constructs [if and ifthen-else] (up to a maximum of two nested levels)
* Include iteration (looping) structures [fixed counter loop]
* incorporate a combination of iteration and condition structures as part of the solution (i.e. program code)
* Write code which applies programming language tools and constructs to draw various shapes (turtle type commands) on an output screen/window. Reinforce concepts such as:
* Sequence
* Selection
* Iteration
* Creation of objects and shapes
* Design a coding solution to a problem incorporating a combination of different programming constructs which include:
* Sequence
* Selection
* Iteration
* Design and develop solutions for specific problems that include computational thinking and applying software engineering principles.
* Explore lists/arrays (storing and accessing a list of numbers and strings) and containers. (Range: Manipulating lists/arrays such as adding, deleting, replacing, inserting items.)

# FORMATIVE ASSESSMENT 3.1 INDIVIDUAL TASK

3.1 **Single-board computing** is a complete, functioning computer in which the microprocessor, input/output functions, memory, and other features are all built on a single circuit board, with RAM built in at a pre-determined amount and with no expansion slots for peripherals. (2)

3.2 Types of Arduino boards

* Arduino Uno (R3)
* Arduino Nano
* Arduino Micro
* Arduino Due
* Arduino Mega (R3) Board
* Arduino Robot

**ANY THREE** (3)

3.3 Pulse-width modulation (2)

3.4 C++ vs Python (12)

|  |  |  |
| --- | --- | --- |
| **Parameter** | **C++** | **Python** |
| **Extension** | C++ program are saved with .cpp extension. | Python programs are saved with .py extension. |
| **Compilation** | C++ is precompiled. | Python is interpreted. |
| **Speed** | C++ is faster once compiled as compared to python. | Python is slower since it uses interpreter |
| **Nature of variables** | C++ is statically typed. You can not declare a variable without a data type. | Python is dynamically typed. You do not need to indicate data type on declaration |
| **Memory Management** | C++ does not support automatic memory management (no garbage collector but can be implemented manually) | Python offers automatic memory management (garbage collector) |
| **Functions** | C++ accepts and returns a predefined type of value according to the definition. | There is no limitation on the type of the argument and the type of its return value. |

3.5 A compiler translates the entire source code in a single run whereas an interpreter translates the entire source code line by line.  (2)

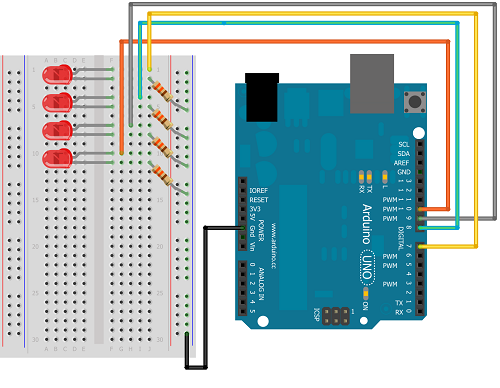
3.6 Binary LED Counter

**Solution**

Connection on the breadboard

**A circuit board with wires

Description automatically generated with low confidence**



**Code**

//declaring the variables for the four pins

int pin1=7;

int pin2=8;

int pin3=9;

int pin4=10;

void setup() {

  // setting up all the pins to OUTPUT mode

  pinMode(pin1, OUTPUT);

  pinMode(pin2, OUTPUT);

  pinMode(pin3, OUTPUT);

  pinMode(pin4, OUTPUT);

}

void loop() {

  //LED Binary Counter code

    //0000

  digitalWrite(pin1, LOW);

  digitalWrite(pin2,LOW);

  digitalWrite(pin3, LOW);

  digitalWrite(pin4, LOW);

  delay(5000);

    //0001

  digitalWrite(pin1, LOW);

  digitalWrite(pin2,LOW);

  digitalWrite(pin3, LOW);

  digitalWrite(pin4, HIGH);

  delay(5000);

  //0010

  digitalWrite(pin1, LOW);

  digitalWrite(pin2,LOW);

  digitalWrite(pin3, HIGH);

  digitalWrite(pin4, LOW);

  delay(5000);

    //0011

 digitalWrite(pin1, LOW);

  digitalWrite(pin2,LOW);

  digitalWrite(pin3, HIGH);

  digitalWrite(pin4, HIGH);

  delay(5000);

  //0100

   digitalWrite(pin1, LOW);

  digitalWrite(pin2,HIGH);

  digitalWrite(pin3, LOW);

  digitalWrite(pin4, LOW);

  delay(5000);

  //0101

   digitalWrite(pin1, LOW);

  digitalWrite(pin2,HIGH);

  digitalWrite(pin3, LOW);

  digitalWrite(pin4, HIGH);

  delay(5000);

  //0110

   digitalWrite(pin1, LOW);

  digitalWrite(pin2,HIGH);

  digitalWrite(pin3, HIGH);

  digitalWrite(pin4, LOW);

  delay(5000);

  //0111

   digitalWrite(pin1, LOW);

  digitalWrite(pin2,HIGH);

  digitalWrite(pin3, HIGH);

  digitalWrite(pin4, HIGH);

  //1000

   digitalWrite(pin1, HIGH);

  digitalWrite(pin2,LOW);

  digitalWrite(pin3, LOW);

  digitalWrite(pin4, LOW);

  delay(5000);

  //1001

    digitalWrite(pin1, HIGH);

  digitalWrite(pin2,LOW);

  digitalWrite(pin3, LOW);

  digitalWrite(pin4, HIGH);

  delay(5000);

  //1010

   digitalWrite(pin1, HIGH);

  digitalWrite(pin2,LOW);

  digitalWrite(pin3, HIGH);

  digitalWrite(pin4, LOW);

  delay(5000);

  //1011

   digitalWrite(pin1, HIGH);

  digitalWrite(pin2,LOW);

  digitalWrite(pin3, HIGH);

  digitalWrite(pin5, HIGH);

  delay(5000);

  //1100

   digitalWrite(pin1, HIGH);

  digitalWrite(pin2,HIGH);

  digitalWrite(pin3, LOW);

  digitalWrite(pin4, LOW);

  delay(5000);

  //1101

   digitalWrite(pin1, HIGH);

digitalWrite(pin2,HIGH);

digitalWrite(pin3, LOW);

digitalWrite(pin4, HIGH);

delay(5000);

//1110

 digitalWrite(pin1, HIGH);

  digitalWrite(pin2,HIGH);

  digitalWrite(pin3, HIGH);

  digitalWrite(pin4, LOW);

delay(5000);

//1111

  digitalWrite(pin1, HIGH);

  digitalWrite(pin2, HIGH);

  digitalWrite(pin3, HIGH);

  digitalWrite(pin4, HIGH);

  delay(5000);

}

(20)

# FORMATIVE ASSESSMENT 3.2 INDIVIDUAL TASK

# Module 4 Programming tools and utilities

After you have completed this module, you should be able to :

* Define *the* term compiler
* Define the term source code
* Explain what a decompiler is used for
* Define the term interpreter
* Explain the difference between a compiler and an interpreter
* Explain what a binary is and when it is produced
* Explain the difference been C and C++ compiler
* List and Explain the basic three stage compiler design (Front Middle Back end)
* Install and configure C/C++ compiler on desktop PC
* Define the term IDE
* Explain what an IDE is used for
* Install and configure IDE on desktop PC
* Explain what the acronym GCC stands for
* Explain what the GCC collection contains
* List compilers included in the GCC
* Explain the difference between GCC and MinGW
* Install and configure GCC C/C++ compiler on Raspberry Pi
* Install and configure Geany on Raspberry Pi
* Define the term debugging
* Explain why an application needs to be debugged
* List common debugging techniques (Interactive, Print, Remote)
* Define the term breakpoint as it relates to debugging
* Define the term stepping as it relates to debugging
* Explain what the GDB tool is used for
* Install and configure MinGW on desktop PC
* Locate the GDB application in the MINGW installation
* Add the MINGW bin directory to the operating system path environment variable
* Compile source code wit the -g flag in IDE
* Explain the concept of a debugger frontend
* Install and configure gdbgiu on Desktop PC
* Start gdbgui application
* Load a compiled C/C++ binary (.exe)
* Run the loaded binary
* Step through the running binary line by line
* Investigate (watch) the values of local variables
* Explain what Git is and what Git is used for
* Discuss the three main goals of the Git VCS
* Explain the relationship between local Git directory and directory located on server
* Discuss the software license used for Git
* Name and explain the advantages and disadvantages to the software license used by Git
* Explain why Git was created
* Explain the relationship between Git and hosting providers like GitHub
* List major open-source projects using Git
* Install and configure Git on desktop PC
* Install and configure Git on Raspberry Pi
* List and explain common terminology associated with distributed VCS’s

# FORMATIVE ASSESSMENT 4.1 INDIVIDUAL TASK

4.1.1 A compiler is a language processor that reads a whole source program written in high- level language in one go and converts it into an equivalent program written in machine code. (2)

4.1.2 Compiled languages.

* C, C++
* C#
* Java
* Erlang
* Haskell
* Rust
* Go.

**ANY OF THE THREE (3)**

4.1.3 The back end of compiler includes those portions that depend on the target machine and generally those portions do not depend on the source language, just the intermediate language. These include

1. Code optimization

2. Code generation, along with error handling and symbol- table operations.

(3)

4.1.4 Advantages of a compiler

**Advantages of compilers**

1. **Independence**-No other program or application is required to run the executable file of source codes.
2. **Optimisation**-The compiled program is well optimized and runs faster.
3. **Data Security**-The compiler generates executable files that can be executed on any other system.
4. **Speed-**Compiler are faster than interpreter

**Disadvantages of Compilers**

1. **Hardware Specific**
2. **Time consuming-** Compilation process takes time before an executable file is produces
3. **Extra memory**-Because source code object code file and executable files
4. **Debugging Difficulty**- All errors are shown at once. (6)

4.1.5 An interpreter program executes other programs directly, running through program code and executing it line-by-line. (2)

4.1.6 Differences between compiler and interpreter.

| **Compiler** | **Interpreter** |
| --- | --- |
| A compiler translates the entire source code in a single run. | An interpreter translates the entire source code line by line. |
| Compilers are faster than an interpreter. | Interpreters are slower than the compiler. |
| CPU utilization is more. | CPU utilization is less as compared to the compiler. |
| Both syntactic and semantic errors can be checked simultaneously. | Only syntactic errors are checked. |
| The compiler is larger than interpreter therefore require more memory. | Interpreters are often smaller than compilers. |
| The localization of errors is difficult. | The localization of error is easier than the compiler. |

(12)

**Total : 28 Marks**

# FORMATIVE ASSESSMENT 4.2 GROUP TASK

**4..2.1** As part of software installation, you are expected to be able to install any IDE and the C++ compiler. We have listed a lot of IDE’s such as Code::Blocks, Geany, CLion, VSCode.

**Task.**

In groups, you are required to uninstall Code::Blocks and Mingw.

Reinstall Code::Blocks and Mingw

Test if gcc/g++ are available.

Run a simple hello world program.

(20)

# FORMATIVE ASSESSMENT 4.3 INDIVIDUAL TASK

4.3.1 GCC stands for “GNU Compiler Collection”. GCC is an integrated distribution of compilers for several major programming languages such as C, C++, Fortran, Ada, D, and Go. (2)

4.3.2 GCC stands for GNU Compiler Collection -on open-source collection of compilers for various languages based on a common core compiler and linker whereas MinGW stands for Minimal GNU for Windows. This is a copy GCC plus other common utilities and libraries used in compiling and linking C or C++ code using the GNU tools. (2)

4.3.3 An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. (2)

4.3.4 Installing Geany on a Raspberry Pi.

Step 1: Navigate to the Raspberry Icon

Step 2: Select accessories and select terminal.

Step 3: Type the following command:

sudo apt-get install geany

Step 4: Press enter. Geany IDE will be installed.

(4)

4.3.5 Complete the following table to differentiate G++ from GCC. (8)

|  |  |  |
| --- | --- | --- |
| **Parameter** | **G++** | **GCC** |
| **Uses** |  |  |
| **Compilation** |  |  |
| **Compilation Command** |  |  |
| **File Linking in Library** |  |  |

4.3.5 Differences between G++ and GCC

|  |  |  |
| --- | --- | --- |
| **Parameter** | **G++** | **GCC** |
| Uses | G++ is used for compiling C++ | Used for compiling C |
| Compilation | can compile either.cpp or.c files, but they will only be treated and released as C++ files. | can compile either.cpp or.c files, but they will be treated and released as C++ or C |
| Command for Compilation | g++ fileName.cpp -o binary. | gcc fileName.c -o binary. |
| File Linking in Library | When we use the G++ command to link the object files, the files automatically link in the standard C++ libraries. | It does not happen in the case of GCC. |

(8)

**Total : 18 marks**

# FORMATIVE ASSESSMENT 4.4 INDIVIDUAL TASK

4.4.1

a) Syntax errors- is a bug that occurs when a computer program has an incorrectly typed statement.

b) Runtime errors-is a type of error which during runtime and could be caused by inputting wrong data type.

c) Logic errors-occurs when program runs but gives incorrect results. (6)

4.4.2 FOUR reasons for debugging computer programs.

a) Debugging ensures that the final product is bug free.

b) Competitive advantage-Bug free programs will build a good reputation for the software firm.

c) Saves Time**-** Performing debugging at the initial stage saves the [time of software developers](https://www.educba.com/career-as-a-software-developers/).

d) Debugging ensures that requirements are met. (4)

4.4.3 Remote debugging technique. (1)

4.4.4 Stepping is a debugging technique that involves executing code one instruction or line at a time. (2)

4.4.5 Setting up a breakpoint

Text

Description automatically generated

Figure 4. 1: Sample code

Output.

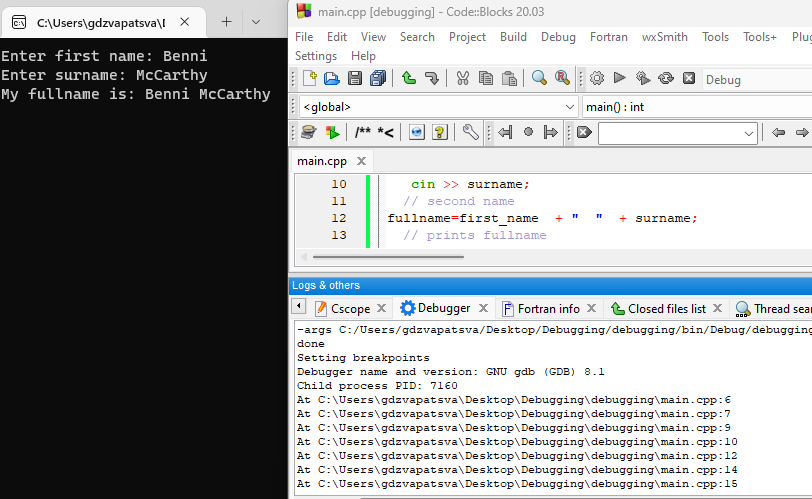


Figure 4. 2:Output

(10)

**Total: 22 marks**

# FORMATIVE ASSESSMENT 4.5 GROUP TASK

4.5.1 Installing gdbgui and debugging a simple program

Make use of the following program.

Text

Description automatically generated

This section will need to be repeated again once students master big programs, debugging very small programs at this point might not show the intended purpose.

# FORMATIVE ASSESSMENT 4.6 INDIVIDUAL TASK

4.6.1 Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. (2)

4.6.2 Git is a free and open-source distributed version control software designed to handle everything from small to very large projects with speed and efficiency whereas GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git.

(4)

4.6.3 FIVE benefits of using Git.

1. Data replication and redundancy are both possible.
2. It is a service with high availability.
3. There can only be one Git directory per repository.
4. Excellent network and disc performance are achieved.
5. On any project, collaboration is very simple.

4.6.4 $ git init (2)

4.6.5 Git -v or git –version or git version gives the version of git software installed on the machine (pc or laptop). (2)

4.6.6 Adding files to Git

Sample Solution

* gdzvapatsva@GDzvapatsva-LP MINGW64 ~/Desktop

$ mkdir git\_revision

* gdzvapatsva@GDzvapatsva-LP MINGW64 ~/Desktop

$ cd git\_revision

* gdzvapatsva@GDzvapatsva-LP MINGW64 ~/Desktop/git\_revision

$ echo "Working with git and github is saves time" >> file1.txt

* gdzvapatsva@GDzvapatsva-LP MINGW64 ~/Desktop/git\_revision

$$ git init

Initialized empty Git repository in C:/Users/gdzvapatsva/Desktop/git\_revision/.git/

* gdzvapatsva@GDzvapatsva-LP MINGW64 ~/Desktop/git\_revision (master)

$ git add .

warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the next time Git touches it

* gdzvapatsva@GDzvapatsva-LP MINGW64 ~/Desktop/git\_revision (master)

$ git commit -m "initial push"

[master (root-commit) 59dc420] initial push

1 file changed, 1 insertion(+)

create mode 100644 file1.txt

* gdzvapatsva@GDzvapatsva-LP MINGW64 ~/Desktop/git\_revision (master)

$ git remote add origin https://github.com/gpdzvapatsva/git\_revision.git

* git branch -M main
* git push -u origin main

To https://github.com/gpdzvapatsva/git\_revision.git

\* [new branch] main -> main

branch 'main' set up to track 'origin/main'.

(20)

**Total :35 marks**

# SUMMATIVE ACTIVITY 4.7 INDIVIDUAL TASK

4.7.1 The fundamental distinction between distributed and centralized version control is that although versions in distributed version control can be saved both in local machine repositories and remote repositories, versions in centralized version control are saved in the remote repository **only.** (4)

4.7.2 Source code is the term used to describe programming statements that are created by a programmer using a text editor or visual programming tool and then saved in a file.(2)

4.7.3 FOUR examples of IDE’s used for C++ programming. (4)

* Geany
* VSCode
* Code::Blocks
* CLion
* CodeLite

4.7.4 C++ uses cout command to display output to the screen. (2)

4.7.5 Advantages of using GIT.

* When using DVCS, all contributors can work concurrently
* Allows programmers to work remotely
* Speeds up development process (3)

4.7.6 git clone <https://github.com/gpdzvapatsva/git_revision.git> (2)

4.7.7 The git config list command will display every Git config property found in every Git file with a different scope. (2)

4.7.8 GNU Compiler Collections, or GCC, is used to compile primarily C and C++ code. The g++ command invokes the GNU C++ compiler and is used to prepare, compile, assemble, and link source code to create executable files. G++ is primarily used to compile C++ while gcc is used for C programs. G++ links the object files in std libraries while gcc does not do the same. (4)

4.7.9 git config --global user.name "cplusguru" (2)

4.7.10 THREE common debugging techniques

* Print debugging
* Interactive debugging
* Remote debugging  (3)

**Total :28 Marks**

# Module 5: Introduction to a high-level programming language

By the end of the module, students should be able to:

* Create a new project using an IDE
* Create a new file with extension CPP
* Explain why C++ source files use the .cpp extension and C++ source files uses the .c extension.
* Write the C++ code to display output “Hello World”.
* Compile a C++ application using an IDE
* Explain what type of file is produced by the C++ compilation process
* Show or hide the IDE message window
* Investigate the IDE message window to determine if compilation succeeded
* Investigate the IDE message window to locate line numbers that contain errors
* Build a C++ application using an IDE
* Explain what file type is produced by the C++ build process
* Run a C++ application using the IDE
* List and explain the different parts of a simple C/C++ application
* Explain case sensitivity as it applies to C/C++
* Explain the term free-format language and how that applies to C/C++
* Explain the term block-structured language and how that applies to C/C++
* Explain what a comment is used for in the C/C++ language
* Explain how the compiler will treat a comment
* List and explain two types of comments allowed by the C/C++ compiler
* Discuss the importance of adding comments to a C/C++ application

# FORMATIVE ASSESSMENT 5.1 INDIVIDUAL TASK

5.1.1 High-level languages enable programmers to write instructions in a more understandable language than low-level languages. Examples of high-level languages includes Python, C++, Java etc. (2)

5.2.2 An integrated development environment (IDE) is a software suite that consolidates basic tools required to write and test software. Examples of IDE’s which can be used for C++ are Code::Blocks, Geany, VSCode etc. (2)

5.2.3 Extension for C++ is .CPP. (2)

5.2.4 Compiling a C++ program entails converting the source code (.cpp files) into an executable or library that can run on a specific platform. This process can be divided into three key stages:

* **Pre-processing**-The outputted code will be ready to compile by the end of the pre-processor stage.
* **Compilation-**C++ compilation is a two-step procedure. First, the compiler converts the source code to assembly language. Second, using an assembler, the assembly language is converted into actual machine code. The resulting output is a collection of files known as an object file. The object code file has .obj or .o file extension
* **Linking**-The final stage generates the executable or library.

Diagram

Description automatically generated

**(4)**

5.2.5 Structure of C++ program

Diagram

Description automatically generated with medium confidence

(10)

5.2.6 C:\Users\gdzvapatsva\Desktop\exam>notepad

Type the following

#include <iostream>

using namespace std.

int main (){

/\*This is the main block

I am learning about multiline comments

\*/

cout<<"Compiling C++ using command line”.

return 0.

}

* Save as exams.cpp

Navigate to exams folder on desktop and enter the following commands

* g++ -g exams.cpp -o exams1.exe
* exam>exams1 (10)

5.2.7 Compiling C++ on Geany

* Build
* Compile
* Run (3)

5.2.8 Compiling is the process of converting the high-level source code files to machine level code whereas Building is the process of converting the high-level language to a executable. It would involve compiling and linking . (4)

**Total: 34 marks**

# FORMATIVE ASSESSMENT 5.2 INDIVIDUAL TASK

5.2.1

* Preprocessor directive- Preprocessor directives are lines included in the code of programs preceded by a hash sign (#).They examine the code before actual compilation of code begins and resolves all these directives before any code is generated by regular statements. Some of the examples are #define, #include. (2)
* Header - Header files contain definitions of function and data types; these header files are imported into any C++ program using the preprocessor directive #include. The preprocessor directive tells the compiler to process these files before compilation. Example of a header file is <iostream>, <fstream>, <cmath.h>, <complex.h>.The .h extension is not mandatory. (2)
* Namespace- Namespaces give us a place to define or declare identifiers, such as variables, methods and classes. Namespaces are used to organize code into logical groups and to prevent name collisions, which can occur when your code base includes multiple libraries. (2)

5.2.2 C++ Program to print two lines.

#include <iostream>

using namespace std;

int main()

{

cout << "\n\n Print a welcome text in a separate line :\n";

cout << "----------------------------------------------\n";

cout << " Welcome to NCV programming\n" ;

cout << " Developing future coders for the digital world"<<endl ;

return 0;

}

(8)

5.2.3 Debugging Solution

#include <iostream>

using namespace std;

namespace addition

//This is a block of code

{

void add\_numbers(){

cout << " The sum of 29 and 30 is : "<< 29+30 <<"\n\n" ;

}

}

int main()

{

addition::add\_numbers();

}

5.2.4 True or False

a) Since C is a subset of C++, all C programs will run under C++ compilers. **False**

b) By merely looking at one or two lines of code, we can easily recognise whether a program is written in C or C++. **False. Most lines are the same in C or C++**

c) The main {} function in C++ and in C are the same. **False. C++ returns integer by default while C returns void.** (3)

5.2.5 Comments in C++ are simply a piece of source code ignored by the compiler.  They help the programmer to add additional information to source code. (2)

5.2.6 C++ comments

* //single-line comment
* /\* block comment \*/ (4)

5.2.7 A free-format language is a programming language in which character positioning on the page in program text is irrelevant. Program text does not need to be placed in specific columns. C++ is a free-format programming language. (2)

**Total:30 Marks**

# SUMMATIVE ACTIVITY 5.3 INDIVIDUAL TASK

5.3.1 Steps to start a new project in Geany

Step1: Open Geany from the Taskbar

Step 2: Click on File and select New

Step 3: Click on Document then Set FileType and select programming languages. Choose C++ source file

Step 4: Click File and select Save As. Assign a new name addition.cpp

Step5: Type the simple program

Step 6: Click Build and select Compile

Step 7: Click *Build* and select *Build*

Step 8: Click *Build* and select *Execute* (8)

5.3.2 True (1)

5.3.3 Solution

#include <iostream>

using namespace std;

namespace languages

//This is a block of code

{

void display(){

cout << " C++ is an extension of C "<<endl;

}

}

namespace jobs

{

void display(){

cout << " I want to work as a backend developer";

}

}

int main(){

languages::display();

jobs::display();

}

(5)

5.3.4 Object files with .O extension are produced at compilation stage. (2)

5.3.5

1. **Compile-time**: These occur due to misuse of language constructs, such as syntax errors. Normally easy to find by using compiler tools and warnings to fix reported problems.
2. **Run-time**: These are much harder to figure out, as they cause the program to generate incorrect output (or “crash”) during execution. Example is a runtime error.

(4)

5.3.6 A block-structured programming language is one that allows the creation of blocks, including blocks nested within other blocks. C++ define blocks of scope using some sort of delimiter pair for e.g., braces in { } to denote a method.

namespace jobs

{

void display(){

cout << " I want to work as a backend developer";

}

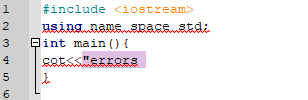
(4)

5.3.7 **Importance of comments**

1. Saves time- new programmers who join the organisation can quickly understand the code segments by reading comments.
2. Comments helps to identify important blocks of code easily.
3. Comments help to understand why programmers used a certain way to get a solution rather than the other option.
4. Comments help to add description and clarity of what certain blocks of code do by adding context. (4)

5.3.8 A breakpoint is a deliberate halting or pausing point placed in a program for debugging purposes in software development. (2)

5.3.9 A syntax error is a bug that occurs when a computer program has an incorrectly typed statement. Here is an example of incorrectly typed reserved words which will generate syntax errors.



Line 2 and 4 have errors in spelling of reserved words namespace has no space in between.

Line 4 the correct spelling is cout.

(3)

**Total :31 Marks**

# Module 6: Data Types, Variables and Output

After you have completed this module, you should be able to :

* List Data Types in C/C++
* Define the term data type
* Define the term literal value
* Name and explain two numerical data types used in C/C++
* Explain the difference between Integer data types and floating-point data types
* List the different integer data types
* List the different floating-point data types
* Explain the float, double data types in terms of:
* What the type can consist of
* What the maximum and minimum value can be for the type
* What operations is allowed on the type
* Example literal values allowed for the type
* Example literal values not allowed for the type
* Whether the type is signed or unsigned
* Number of bytes memory used to store type value
* Explain the int, char, bool, float double data types in

terms of:

* What the type can consist of
* What the maximum and minimum value is
* What operations is allowed on the type
* Example literal values allowed for the type
* Example literal values not allowed for the type
* Whether the type is signed or unsigned
* Number of bytes memory used to store type value
* Use variables as part of a solution
* Define the term variable
* Explain how variables are used in C/C++ applications
* Identify and list C/C++ key words
* List and describe the syntax rules for naming variables
* Write C/C++ code to create variables of different types
* Write C/C++ code to assign literal values to variables of different types
* Write C/C++ code store the result of an arithmetic expression (simple and complex) in a variable
* Write C/C++ code to print the value associated with a variable (combined with string literal for context)
* Write C/C++ code store the result of an arithmetic expression that contains variables on both sides of the operand (simple and complex) in a variable
* Write C/C++ code to assign the value of one variable to a second variable
* Explain the difference between variable declaration and variable initialisation
* Write C/C++ code to initialise variables of different types
* Write C/C++ code to increase or decrease the value of a variable (accumulation statements)
* Write C/C++ code to increase or decrease the value of a variable with 1 using the increment
* Operator
* Write C/C++ code to print the value associatedwith a variable (combined with string literal for context)
* List the different arithmetic operators
* Define the term binary operator
* Define the term operand
* List and explain (with examples) what a binary arithmetic expression consists of
* Write C/C++ code that uses printf or cout to display the value of different arithmetic expressions (combined with string literal for context)
* List and explain the data type produced based on the data type of the operands involved
* List and explain the rules involved when creating complex arithmetic expressions containing multiple operands.
* List and describe the three levels of precedence.
* Write C++ code that uses printf or cout to display the value of different complex arithmetic expressions (combined with string literal for context)
* Identify or correct the general form for multiple variable declarations
* Explain why multiple declarations should be of the same type
* Write C/C++ code that declares multiple variables in a single line
* Escape characters and New Lines

Escape characters

* Explain what meaning the backslash character will have in C/C++
* Define the term escape character
* Define the term escape sequence
* Write C/C++ code that uses cout to display/use variable (combined with string literal and variable values for context):
* Backslash
* Question mark
* Single and Double quotation
* Horizontal tab
* Alert
* Give examples of where output in an application could be formatted to be more attractive to the user
* Explain what field with manipulators are used for
* Write C/C++ code that uses printf and formatting strings to:
* Set the field width
* Set the default leading fill character
* Left justify numbers
* Right justify numbers
* Set floating point precision

**FORMATIVE EXERCISES GROUP/INDIVIDUAL TASKS**

**Task 6.1 : Program to show the following output making use of sizeof keyword**

**Solution**

#include <iostream>

using namespace std;

int main()

{

//printing the different size of data types

cout << "\n\n The following are the sizes of different data types :\n";

cout << "------------------------------------------\n";

cout << " The sizeof(char) is :" << sizeof(char) << " bytes \n" ;

cout << " The sizeof( short) is :" << sizeof(short) << " bytes \n" ;

cout << " The sizeof(int) is :" << sizeof(int) << " bytes \n" ;

cout << " The sizeof(long) is :" << sizeof(long) << " bytes \n" ;

cout << " The sizeof(long long) is:"<< sizeof(long long) << " bytes \n";

cout << " The sizeof(float) is :" << sizeof(float) << " bytes \n" ;

cout << " The sizeof(double) is :" << sizeof(double) << " bytes \n";

cout << " The sizeof(long double) is:" << sizeof(long double)<<"bytes \n";

cout << " The sizeof(bool) is :" << sizeof(bool) << " bytes \n\n";

return 0;

}

**Task 6.2 : output of binary literal addition**

Solution

sum is 3

**TASK 6.3 :Evaluating a quadratic equation and find the roots (x1,x2).**

0b110x 2 – 0b110000x – 0b110110 = 0 a=6, b=48, c=54

**Solution**

#include <iostream>

#include <cmath>

using namespace std;

//declaring variables

float x1, x2;

int main()

{

// 0b110x2 – 0b110000x − 0b110110 = 0 a=0b110, b=0b110000, c=0b110110

//processing

x1=(-(-0b110000)+ sqrt((0b110000\*0b110000)-4\* 0b110\*-0b110110))/(2\*0b110);

x2=(-(-0b110000)-sqrt((0b110000\*0b110000)-4\* 0b110\*-0b110110))/(2\*0b110);

//displaying the output

cout << "The value of x1 is :"<<x1 << endl;

cout << "The value of x2 is :"<<x2 << endl;

return 0;

}

**Output**

The value of x1 is :9

The value of x2 is :-1

**TASK 6.4 :Program in C++ to swap two numbers.**

**Solution**

#include <iostream>

using namespace std;

int main()

{

cout << "\n\n Swap two numbers :\n";

cout << "-----------------------\n";

int num1, num2, temp;

cout << " Input 1st number : ";

cin >> num1 ;

cout << " Input 2nd number : ";

cin >> num2;

temp=num2;

num2=num1;

num1=temp;

cout << " After swapping the 1st number is : "<< num1 <<"\n" ;

cout << " After swapping the 2nd number is : "<< num2 <<"\n\n" ;

**Task 6.5 :Using a ternary operator to write a program to check if a person qualifies to vote or not.**

**SOLUTION**

#include <iostream>

using namespace std;

int main()

{

int age;

cout << "Please enter your age: !" << endl;

//entering age

cin>>age;

//ternary condition to check age

string result = (age >=18 ) ? " can vote " : " can not vote";

//displaying output

cout<<"Based on the analysis of your age, your " <<result;

return 0;

}

Output

Please enter your age: !

15

Based on the analysis of your age, your cannot vote

**TASK 6.6 : Program to print the sum of two numbers using variables. The program must ask user to input the values of the two numbers and display the output as:**

**The sum of 67 and 23 is : 90**

**Solution**

#include <iostream>

using namespace std;

int main()

{

//declaring variables

int a=0, b=0, sum=0;

cout << "\n\n Program to print the sum of two numbers :\n";

cout << "-----------------------------------\n";

//accepting input

cout << "Enter the value of a\n";

cin>>a;

cout << "Enter the value of b\n";

cin>>b;

//processing

sum=a+b;

//displaying output

cout << " The sum of "<< a << " and "<<b <<" is : "<< sum <<"\n\n" ;

return 0;

}

Output

Enter the value of a

67

Enter the value of b

23

The sum of 67 and 23 is : 90

**TASK 6.7**

Simplify 4 + (2 + 1)2.

4 + (2 + 1)2

= 4 + (3)2

= 4 + 9

= **13**

**Task 6.8 :Program to compare the following complex numbers.**

complex\_two(3.14, 1.5);

complex\_three(1.5, 3.14);

**Result**

#include <iostream>

#include <complex>

using namespace std;

int main()

{

//declaring the complex number variables and assigning values

complex<double> complex\_two(3.14,1.5);

complex<double> complex\_three(1.5, 3.14);

//comparing complex numbers

bool result=(complex\_three==complex\_two);

//displaying output

cout <<result<<endl;

return 0;

}

**Output**

0

**Task 6.9 :Identifying errors**

Correct the program to produce the following output:

Cape Town temperature is 26 degrees celcius

Joburg temperature is 26 degrees celcius

Durban temperature is 26 degrees celcius

**Solution**

#include <iostream>

using namespace std;

int main()

{

//declaring multiple variables

float cpt\_temp, jbg\_temp, durb\_temp;

cpt\_temp=jbg\_temp=durb\_temp=26;

cout<<"Cape Town temperature is "<<cpt\_temp<<"degrees celcius"<<endl;

cout<<"Joburg temperature is "<<jbg\_temp<<"degrees celcius"<<endl;

cout<<"Durban temperature is "<<durb\_temp<<"degrees celcius"<<endl;

return 0;

}

**Task 6.10 :Program to compute quotient and remainder.**

**Solution**

#include <iostream>

using namespace std;

int main()

{

//declaring the variables

int dividend, divisor, quotient,remainder;

cout << "\n\n Calculating quotient and remainder :\n";

cout << "-------------------------------------\n";

//capturing input

cout<<" Input the dividend : ";

cin>>dividend;

cout<<" Input the divisor : ";

cin>>divisor;

//processing

quotient=dividend / divisor;

remainder=dividend % divisor;

//displaying the result

cout<<" The quotient of the division is : "<< quotient << endl;

cout<<" The remainder of the division is : "<< remainder << endl;

cout << endl;

return 0;

}

**Output**

Calculating quotient and remainder :

-------------------------------------

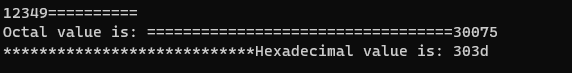
Input the dividend : 35

Input the divisor : 4

The quotient of the division is : 8

The remainder of the division is : 3

**Task 6.11** :Program to show the different values of an integer using the setbase() function.



**Solution**

#include <iomanip>

using namespace std;

int main () {

int x = 12349;

cout.fill('=');

cout.width(15); cout << left <<x<< '\n';

cout.fill('=');

cout.width(50); cout << left<<"Octal value is: " << setbase(8) << x << endl;

cout.fill('\*');

cout.width(50); cout <<right<< "Hexadecimal value is: " << setbase(16) << x << endl;

return 0;

}

# FORMATIVE ASSESSMENT 6.1 INDIVIDUAL TASK

6.1.1 A data type specifies the type of data that a variable can store such as integer, floating, character etc. (2)

6.1.2 Three categories of data types in C++

* Primitive/Built-in data types e.g int, float, char, wide character, Boolean, double, void
* Abstract data types e.g functions, Array, Pointers and Reference data types
* Derived data types e.g class, structure, union, enumeration, Typedef defined data types (9)

6.1.3 Literals are data used for representing fixed values e.g integer literals, boolean literals, float literals.

6.1.4 Representing decimal to the different number literals

|  |  |  |
| --- | --- | --- |
| **Decimal Number** | **Type of integer literals** | **Example** |
| 45 | Binary-literal (base 2) | 0b101101 |
| **87** | Binary-literal (base 2) | 0B1010111 |
| **45** | **Octal-literal (base 8)** | 055 |
| **124** | **Octal-literal (base 8)** | 0174 |
| **197** | Hexadecimal literal(base 16) | 0x C5 |
| 350 | Hexadecimal literal(base 16) | 0x 15E |

(6)

6.1.5 c) Explanation: The const will declare with a specific type of value and #define is used to declare user defined constants. (2)

6.1.6 B. Explanation: Logical values can be either true or false, so the boolean type is suited for it. (2)

6.1.7 Conversion error. Data is a string. (2)

6.1.8 Sizeof is a unary compile-time operator used to determine variables, data types, and constants' sizes in bytes. (2)

6.1.9 A variable is a place to store data and values in memory location. (2)

**[Total=27 Marks]**

# FORMATIVE ASSESSMENT 6.2 INDIVIDUAL TASK

6.2.1 Multiple variables

int x,y,z;

x=y=z=53;

OR

int x=53,y=53,z=53;

cout<<z<<endl; (2)

6.2.2 An **operator** is a symbol that operates on a value to perform specific mathematical or logical computations. (2)

6.2.3 THREE categories of operators (3)

* Unary Operators
* Binary Operators
* Ternary Operators (3)

6.2.4 A binary expression is one in which there are two or more operands and one or more operators. Operands are the values which are acted upon by the operator for example.

answer= num2 + num3

num2 and num3 are the operands; + is the operator. (4)

6.2.5 Multiple declarations.

int num2=35, num3=17, answer=0;

answer= num2 + num3;

cout<<num2 << " + " << num3 <<" = "<< num2+num3<<endl;

(6)  
6.2.6 Program to calculate area and perimeter of circle.

#include <iostream>

int main()

{

using namespace std;

//declaring variables

float perimeter=0, area=0, radius=0;

//declaring a constant pi

const float pi= 3.14;

cout << "Enter radius of circle" << endl;

//accepting input

cin >> radius;

//calculating perimeter

perimeter = 2\*pi\*radius;

//calculating area

area=pi\*radius\*radius;

//displaying results

cout << "Perimeter whose radius is " <<radius<<" is " <<perimeter<<"cm2"<< endl;

cout << "Area whose radius is " <<radius<<" is " <<area<<"cm2"<< endl;

return 0;

}

(9)

6.2.7 Comparing values using comparison operators and logical operators.

#include <iostream>

int main()

{

using namespace std;

int first\_number;

int second\_number;

cout<<"Please enter the value of first number"<<endl;

cin >> first\_number;

cout<<"Please enter the value of second number"<<endl;

cin >> second\_number;

//relational and logical operators

cout << ((first\_number < 50) && (first\_number < second\_number) )<< endl;

return 0;

}

The output is 1 (true) (10)

6.2.8 Define the term typecasting. (2)

6.2.8 Casting is a conversion process wherein data can be changed from one type to another.   (2)

**[Total =41 Marks]**

# FORMATIVE ASSESSMENT 6.3 INDIVIDUAL TASK

6.3.1 Variable naming rules

* Variable names must not start with a number
* Variable names can consist of alphanumeric characters.
* Special characters are not allowed when naming variables. Underscore is permitted.
* No use of keywords in variable names
* Variables must have unique names, i.e., you cannot have two variables with the same name.
* Variable names are case sensitive, so firstname and FIRSTNAME are two different variables.

**ANY CORRECT FIVE** (5)

6.3.2 Operator precedence

1. Answer is 4
2. Answer is 0
3. Answer is 3
4. Answer is -2 (8)

6.3.3 It is important so that there are no space allocation issues. In fact, any attempt to use different data types as follows will result in compilation errors.

int a = 1, float b = 2, c = 3, d = 4; (2)

6.3.4

amount = 2 \* count++ -count ;

The output is 2

The code is evaluated as follows:

Amount=(2\*count++)-count

Amount=6 -4 (We used 4 since the value of count was incremented though the original value was the one used in the expression within the bracket. (3)

**[Total =18 marks]**

# FORMATIVE ASSESSMENT 6.4 INDIVIDUAL TASK

6.4.1

cout << " The town of \"Mthatha\" is in the Eastern Cape "<<"\n" ;

(2)

6.4.2 Escape sequences in [C++](https://geekonpeak.com/tag/c/) are character combinations that comprise a backslash (\) followed by some character or symbol. (2)

6.4.3

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

bool is\_a\_programmer=true;

cout << boolalpha << is\_a\_programmer << '\n';

cout << noboolalpha << is\_a\_programmer << '\n';

}

(4)

**6.4.4 Escape Sequence**

|  |  |
| --- | --- |
| **Character** | **Description** |
| \b | used to delete the previous character (backspace) |
| \n | is used to jump to the next line. |
| \t | Horizontal tab |
| \v | Vertical tab |
|  |  |
| \? | Question mark |
| \a | Beep sound |
| \’ | Single quote |
| \” | Double qoute |

(4)

**[Total =12 marks]**

# FORMATIVE ASSESSMENT 6.5 INDIVIDUAL TASK

6.5.1 Manipulators are functions specifically designed to be used in conjunction with the insertion (<<) and extraction (>>) operators on stream objects. (2)

6.5.2 Manipulators with no arguments

1. endl: A new line is entered, and the output stream is flushed after each new line.
2. ends: It is also defined in ostream and it inserts a null character into the output stream.
3. flush:  it forces all the output written on the screen or in the file.

(6)

6.5.3 Manipulators with arguments

**setw()** function allows programmer to set the minimum width of the next output via the insertion operator.

**Syntax**

*cout<<setw(spaces)<<string/variable<<endl;*

**Example**

cout<<setw(20)<<"We are developers"<<setw(20)<<"We are coders"<<endl;

Output

We are developers We are coders

**setfill()-**It replaces setw(whitespaces )’s with a different character.

cout<< setfill('\*') << endl;

cout<<setw(20)<<"We are developers"<<setw(20)<<"We are coders"<<endl;

\*\*\*We are developers\*\*\*\*\*\*\*We are coders

**setprecision():** The setprecision() utility is often used to alter the sequence of numbers displayed inside a floating-point integer output sequence

Example

double my\_number=234.678;

cout<< setprecision(3)<<my\_number<<endl;

**Output**

235

(9)

**[Total =17 marks]**

# SUMMATIVE ACTIVITY 6.6 INDIVIDUAL TASK

6.6.1 Data Types

|  |  |  |
| --- | --- | --- |
| **Item** | **Value** | **Data Type** |
| a | is\_a\_student=false; | bool |
| b | "We are learning about data types"; | string |
| c | 15; | int |
| d | 0b11; | int |
| e | L'x'; | wchar\_t |
| f | 233434.56343; | double |
| g | 277.65 | float |
| h | 45e-2 | Float |

(8)

6.6.2 Swapping numbers without using a temporary variable.

#include <iostream>

using namespace std;

int main()

{

cout << " Swap two numbers :\n";

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

int x1, x2;

cout << " Input 1st number : ";

cin >> x1 ;

cout << " Input 2nd number : ";

cin >> x2;

//swapping numbers

x1=x1+x2;

x2=x1-x2;

x1=x1-x2;

cout << " After swapping the 1st number is : "<<x1 <<"\n" ;

cout << " After swapping the 2nd number is : "<< x2<<"\n\n" ;

}

(10)

6.6.3

a) True

b) True (2)

6.6.4 Assignment operators

|  |  |  |
| --- | --- | --- |
|  | Simple expression | Expression using assignment operators |
| i | a = a+2 ; | a += 2 ; |
| ii | a = a-3 ; | a -= 3 ; |
| iii | a = a\*2 ; | a \*= 2 ; |
| iv | a = a%2 ; | a %= 2 ; |
| v | b = b+(c+2); | b += c + 2 ; |
| vi | d =d (e \* -5); | d \*= e - 5 ; |

(6)

6.6.5 Both the ++x and x++ operators increase variable x by 1, but the ++x pre-increment operator follows the "change-then-use" philosophy while the x++ post-increment operator follows the "use-then-change" principle. (4)

6.6.6 setbase()-Sets the basefield to one of its possible values: dec , hex or oct.

cout<< setbase(8)<<145<<endl;

cout<< setbase(16)<<145<<endl;

cout<< setbase(10)<<0b11<<endl;

**OUTPUT**

221

91

3

(8)

6.6.7 Complex numbers are the numbers that are expressed in the form of a+ib where, a,b are real numbers and 'i' is an imaginary number called “iota”. (2)

6.6.8 Complex number multiplication

#include <iostream>

#include <complex>

using namespace std;

int main()

{

//declaring the complex number variables and assigning values

complex<double> complex\_two(2.2);

complex<double> complex\_three(3.6, 1.14);

//adding complex numbers

complex<double> complex\_four =complex\_two \* complex\_three;

//displaying output

cout <<"The answer for multiplying " << complex\_two<< " x " <<complex\_three<< " = "<< complex\_four<<endl;

return 0;

}

OUTPUT

The answer for multiplying (2.2,0) x (3.6,1.14) = (7.92,2.508)

(10)

**[Total =50 marks]**