Topic: Output Devices

Objectives

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

1. Define output devices.
2. Identify different types of output devices (Monitor, Printer, Speaker, Plotter).
3. Describe the features and uses of output devices.
4. Explain the structure, types, and functions of a monitor and Printer.

Definition of Output Devices

Output devices are hardware components used to convey information from a computer system to the user or another device. They convert processed data into human-readable or machine-readable form.

Examples of Output Devices include: **Monitor, Printer, Speaker, Plotter.**

Features and Uses of Output Devices

- **Monitor**:
  - *Features*: Flat-panel screens (LED or LCD), screen resolution, size, and colour depth.
  - *Uses*: Viewing data, multimedia content, and interacting with software applications.
- **Printer**:
  - *Features*: Print quality (DPI), speed, type (laser, inkjet).
  - *Uses*: Producing text and image hard copies.
- **Speaker**:
  - *Features*: Sound quality, power output, and connectivity.

    - *Uses*: Listening to audio output, music, or voice communication.

- **Plotter**:
    - *Features*: Precision drawing capability, size support.
    - *Uses*: Printing engineering drawings, maps, and advertisements.

The Monitor: Structure, Types, and Functions

Structure of a Monitor

A monitor consists of a screen, a casing, control buttons, and connection port

Types of Monitors

I. **Monochrome Monitor**:

Displays images in two colours (black and white or green and black).

II. **Colour Monitor**:

Displays images in multiple colours using RGB (Red, Green, Blue) technology.

Functions of a Monitor

1. Displays the user interface and data.
2. Shows visual output from software applications.
3. Provides feedback from the system to the user.

The PRINTER

Printers are **output devices** that produce **hard copy** documents or files.

They are classified into **Impact** and **Non-Impact** types.

Types of Printers

1. Impact Printers

Use physical contact to print on paper.

Examples of Impact Printers:

a. **Dot Matrix Printer**:

Uses a print head with pins to strike an ink ribbon. Produces characters and graphics by impact.

b. **Character or Daisy wheel Printer**:

Prints one character at a time. Slower compared to other types.

c. **Line Printer**:

Prints an entire line at a time.

2. Non-Impact Printers

Do not use mechanical contact; rely on other methods such as ink or toner.

Examples of Non-Impact Printers:

1. **Inkjet Printer:**

Sprays ink directly onto the paper. Produces high-quality images and graphics. Common for home and office use.

2. **Laser Printer**:

- Uses laser technology and toner powder to print. Fast and efficient, suitable for large volumes.

3. **Thermal Printer**:

- Uses special heat-sensitive paper. Common in receipt printing.

Comparative Study of Printers

Features	**Dot-Matrix Printer**	**Inkjet Printers**	**Laser-Jet Printers**

Technology	Impact	Non-Impact	Non-Impact
Print Quality	Low	High	Very High
Speed	Slow	Moderate	Fast
Cost	Low	Moderate	High
Noise Level	High	Low	Low

Topic: COMPUTER SYSTEM SOFTWARE

Lesson Objectives:

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

1. Define software.
2. Identify and explain the types of software.
3. Differentiate between system software and application software.
4. Explain different types of system software, including operating systems, translators, and utility programs.
5. Identify and provide examples of operating systems.
6. Explain the different types of translators with examples.

Teaching Aids

- Charts showing types of software
- Computer for demonstration (CLI vs GUI)
- Whiteboard for explanations

Introduction

- Start with a discussion: "Imagine using a computer without any programs. What would happen?"
- Introduce the topic: **Computer System Software**

Definition of Software

Software refers to a set of instructions or programs that direct a computer to perform specific tasks.

Without software, a computer is just an inactive machine.

Types of Software

Software is broadly classified into two types:

A. System software and

B. Application Software

A. System Software

System software controls and manages computer hardware and provides a platform for running application software. It includes:

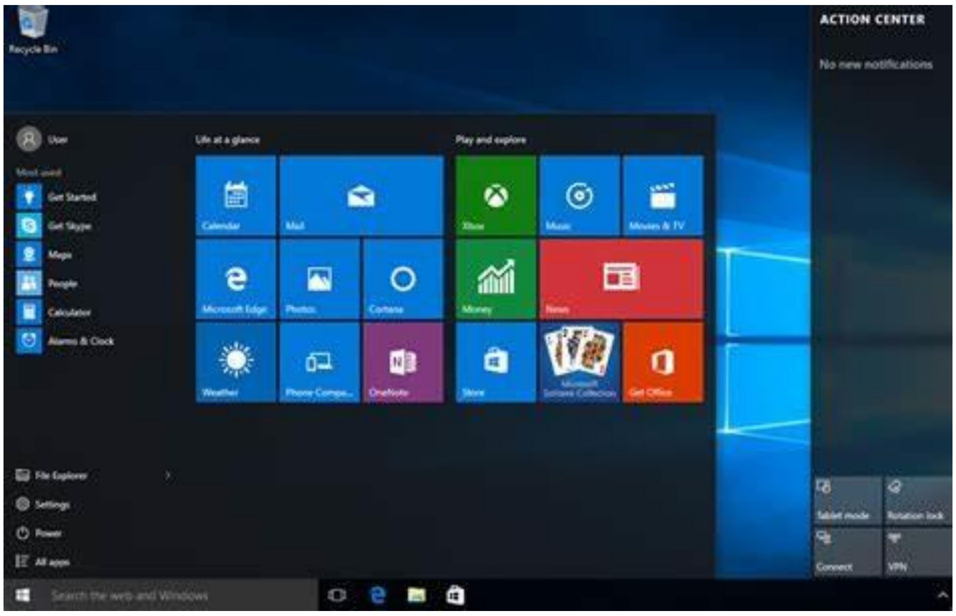
1. Operating System
2. Translators and
3. Utilities software

1. **Operating System (OS):** The main software that manages computer resources and provides an interface for users.

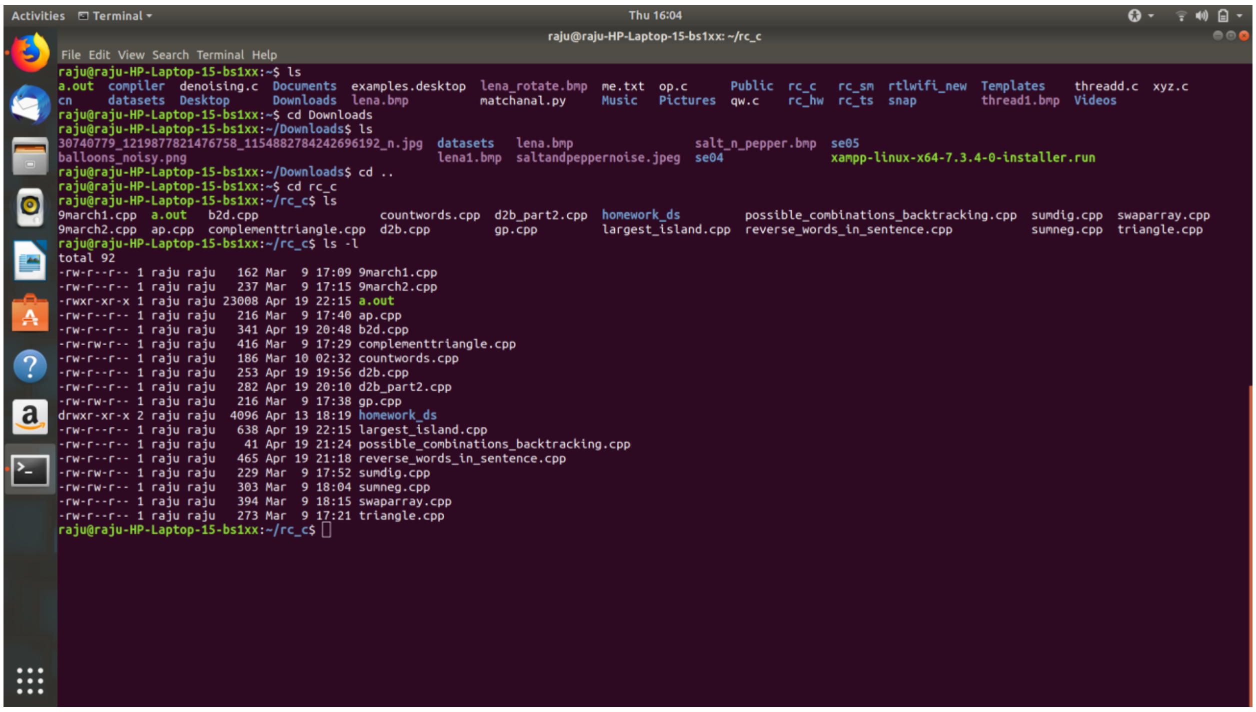
   Examples: MS Windows, Linux, macOS

Types of OS:

I. **Graphical User Interface (GUI):** This is a type of user interface that allows users to interact with electronic devices using graphical elements such as windows, icons, buttons, and menus. e.g., Windows, macOS



II. **Command Line Interface (CLI):** is a text-based user interface used to interact with a computer system. e.g., Linux (CLI mode), DOS.


Activities
Terminal
Thu 16:04
raju@raju-HP-Laptop-15-bs1xx: ~/rc_c
File Edit View Search Terminal Help

```
raju@raju-HP-Laptop-15-bs1xx:~$ ls
a.out  compiler  denoising.c  Documents  examples.desktop  lena_rotate.bmp  me.txt  op.c      Public  rc_c   rc_sm  rtlwifi_new  Templates    threadd.c  xyz.c
cn     datasets  Desktop      Downloads  lena.bmp          matchanal.py     Music   Pictures  qw.c    rc_hw  rc_ts  snap         thread1.bmp  Videos
raju@raju-HP-Laptop-15-bs1xx:~$ cd Downloads
raju@raju-HP-Laptop-15-bs1xx:~/Downloads$ ls
30740779_1219877821476758_1154882784242696192_n.jpg  datasets   lena.bmp                 salt_n_pepper.bmp  se05
balloons_noisy.png                                   lena1.bmp  saltandpeppernoise.jpeg  se04               xampp-linux-x64-7.3.4-0-installer.run
raju@raju-HP-Laptop-15-bs1xx:~/Downloads$ cd ..
raju@raju-HP-Laptop-15-bs1xx:~$ cd rc_c
raju@raju-HP-Laptop-15-bs1xx:~/rc_c$ ls
9march1.cpp  a.out   b2d.cpp                 countwords.cpp  d2b_part2.cpp  homework_ds         possible_combinations_backtracking.cpp  sumdig.cpp  swaparray.cpp
9march2.cpp  ap.cpp  complementtriangle.cpp  d2b.cpp         gp.cpp         largest_island.cpp  reverse_words_in_sentence.cpp           sumneg.cpp  triangle.cpp
raju@raju-HP-Laptop-15-bs1xx:~/rc_c$ ls -l
total 92
-rw-r--r-- 1 raju raju   162 Mar  9 17:09 9march1.cpp
-rw-r--r-- 1 raju raju   237 Mar  9 17:15 9march2.cpp
-rwxr-xr-x 1 raju raju 23008 Apr 19 22:15 a.out
-rw-r--r-- 1 raju raju   216 Mar  9 17:40 ap.cpp
-rw-r--r-- 1 raju raju   341 Apr 19 20:48 b2d.cpp
-rw-rw-r-- 1 raju raju   416 Mar  9 17:29 complementtriangle.cpp
-rw-r--r-- 1 raju raju   186 Mar 10 02:32 countwords.cpp
-rw-r--r-- 1 raju raju   253 Apr 19 19:56 d2b.cpp
-rw-r--r-- 1 raju raju   282 Apr 19 20:10 d2b_part2.cpp
-rw-rw-r-- 1 raju raju   216 Mar  9 17:38 gp.cpp
drwxr-xr-x 2 raju raju  4096 Apr 13 18:19 homework_ds
-rw-r--r-- 1 raju raju   638 Apr 19 22:15 largest_island.cpp
-rw-r--r-- 1 raju raju    41 Apr 19 21:24 possible_combinations_backtracking.cpp
-rw-r--r-- 1 raju raju   465 Apr 19 21:18 reverse_words_in_sentence.cpp
-rw-rw-r-- 1 raju raju   229 Mar  9 17:52 sumdig.cpp
-rw-rw-r-- 1 raju raju   303 Mar  9 18:04 sumneg.cpp
-rw-r--r-- 1 raju raju   394 Mar  9 18:15 swaparray.cpp
-rw-r--r-- 1 raju raju   273 Mar  9 17:21 triangle.cpp
raju@raju-HP-Laptop-15-bs1xx:~/rc_c$ 
```

2. **Translators:** These are system software's that Convert programming languages into machine language for execution by the computer.

   a. **Types of translators:**

      i. **Assembler:** Converts assembly language to machine language.

ii. **Compiler:** Translates high-level language programs into machine code at once.

iii. **Interpreter:** Translates and executes high-level language programs line by line.

3. **Utility Software:** System tools that enhance system performance and security. Examples: Antivirus software, Disk Cleanup, Backup software.

B. Application Software: Software's designed for specific user tasks such as word processing, gaming, and media playback.

Examples: Microsoft Word, Google Chrome, Photoshop

Summary & Conclusion

- Software is essential for computers to function.
- Two main types: System Software (OS, Translators, Utilities) and Application Software.
- OS can be GUI-based (Windows, macOS) or CLI-based (DOS, Linux CLI).
- Translators include assemblers, compilers, and interpreters.

Assessment Questions

1. Define software.
2. Mention two types of system software and explain them.
3. Give three examples of operating systems and classify them as GUI or CLI.
4. Explain the difference between a compiler and an interpreter.
5. Name three utility programs and their functions.

Assignment

1. Research and list five differences between system software and application software.
2. Find out the latest versions of at least three operating systems and their release years.

Topic: COMPUTER APPLICATION SOFTWARE

Learning Objectives: By the end of the lesson, students should be able to:

1. Define application software.
2. Identify different types of application software.
3. List and explain examples of utility programs.
4. Describe different categories of application packages.
5. Recognize packages for specialized areas.

Teaching Aids:

- Computer
- Projector/slides

Introduction

- Ask students if they have used any computer software (e.g., Microsoft Word, games, or antivirus software).
- Introduce the term **Application Software** and explain that it is software designed to perform specific tasks for users.

Definition of Application Software

Application software refers to programs designed to help users perform specific tasks such as word processing, calculations, or graphic design.

Types of Application Software

1. **User Application Programs:** These are programs written by individuals or organizations for specific purposes. Examples include custom-made software for businesses.

2. **Application Packages:** These are pre-designed software packages developed for general or specific purposes. Examples include Microsoft Word, Excel, and CorelDRAW.

Categories of Application Packages

1. **Word Processing Software** – e.g., Microsoft Word, Google Docs.
2. **Spreadsheet Software** – e.g., Microsoft Excel, Google Sheets.
3. **Graphics Software** – e.g., Adobe Photoshop, CorelDRAW.
4. **Database Management Software** – e.g., Microsoft Access, Oracle.
5. **Game Software** – e.g., FIFA, Grand Theft Auto (GTA).

Packages for Specialized Areas

These are application packages designed for specific industries or sectors. Examples include:

- **Accounting Software** – e.g., QuickBooks, Sage.
- **Payroll Programs** – e.g., PaySoft, ADP Payroll.
- **Banking Software** – e.g., Finacle, Temenos.
- **Educational Management Software** – e.g., School Management Systems.
- **Statistical Packages** – e.g., SPSS, MATLAB.
- **Hospital Management Software** – e.g., Medisoft, Clinic Master.

Utility Software

Utility programs are system support software that help maintain and protect a computer in order to enhance performance. Examples include:

- **Editors** – e.g., Notepad, WordPad.
- **Antivirus Software** – e.g., Avast, Norton, McAfee.
- **Device Drivers**

Homework Assignment

- Research and write short notes on five different application software and their uses.
- Find out the most common application software used in banks and hospitals and state their functions.

Topic: PROGRAMMING LANGUAGE

Learning Objectives: By the end of the lesson, students should be able to:

1. Define a programming language.
2. Identify and explain the levels of programming languages.
3. Describe the features of each level of programming language.
4. Provide examples of programming languages.
5. Compare the levels of programming languages.

Teaching Aids:

- Projector/slides (if available)
- Printed notes/handouts
- Charts displaying programming language levels

Introduction

- Begin with a question: "What do you think makes computers understand and execute commands?"
- Introduce the concept of **Programming Language** as a formal language used to communicate with computers.

Definition of Programming Language

A **Programming Language** is a set of rules and syntax used to write computer programs that perform specific tasks.

A **syntax** refers to the set of rules and structures that define the correct way to write code in a particular programming language.

It acts as a medium between humans and computers, enabling communication and task execution.

Levels of Programming Language

Programming languages are classified into three levels:

1. **Machine Language (ML)**

   Also known as **first-generation language (1GL)**. Written in binary (0s and 1s) and directly understood by the computer.

   Highly efficient but difficult for humans to write and understand.

2. **Low-Level Language (LLL)**

   Also called **assembly language** or **second-generation language (2GL)**.

   Uses symbolic codes (mnemonics) instead of pure binary. Requires an assembler to convert it into machine language.

   Faster and more efficient than high-level languages but still complex.

3. **High-Level Language (HLL)**

   Also known as **third-generation language (3GL) and beyond**.

   Uses English-like syntax (e.g., print, if, while) making it easier for programmers.

   Examples include Python, Java, C++, and JavaScript.

   Requires a compiler or interpreter to convert code into machine language.

Features of Each Level of Programming Language

1. **Machine Language (ML):**
    - Fastest execution speed.
    - Hard to write, read, and debug.
    - Directly understood by computers.
2. **Low-Level Language (LLL):**
    - More readable than ML but still complex.
    - Requires an assembler for execution.
    - Offers efficient memory usage.
3. **High-Level Language (HLL):**
    - Easier to learn and write.
    - Requires a compiler or interpreter.
    - More portable across different computer systems.

Comparison of Levels of Programming Language

Feature	**Machine Language**	**Low-Level Language**	**High-Level Language**
Readability	Hard	Moderate	Easy
Speed	Fastest	Faster	Slower
Debugging	Very Difficult	Difficult	Easy
Learning Curve	Very High	High	Low

Assignment

- Research and write about one high-level programming language and its real-world applications.
- Explain the advantages of using high-level languages over low-level and machine languages.

Conclusion:

Programming languages serve as the backbone of software development. Understanding their levels and characteristics enables students to appreciate how software is created and optimized.

Topic: BASIC PROGRAMMING LANGUAGE

Learning Objectives: By the end of the lesson, students should be able to:

1. Define BASIC and explain its full meaning.
2. Identify and use basic statements in BASIC programming.
3. Understand BASIC characters.
4. Use arithmetic operators and expressions in BASIC.
5. Evaluate arithmetic expressions in BASIC.
6. Write and understand a simple BASIC program.

Teaching Aids:

- Printed notes/handouts
- Whiteboard/marker
- Projector/slides (if available)

Introduction

BASIC stands for **Beginners' All-Purpose Symbolic Instruction Code**, it is a high-level programming language designed to be simple and easy to learn for beginners. It was developed to help beginners learn programming.

Key features of BASIC include:

I. Ease of Use: Designed for beginners, with straightforward syntax and commands.

II. Interactivity: Early versions allowed for immediate execution of commands, making it suitable for learning and experimentation.

III. Portability: It can be implemented on various computer systems.

BASIC Statements

Basic statements in BASIC programming include:

I. **LET** – Assigns values to variables (e.g., LET X = 5).

II. **READ** – Reads data from a DATA statement.

III. **INPUT** – Accepts input from the user.

IV. **DATA** – Stores a set of values for use in the program.

V. **END** – Terminates the program execution.

VI. **PRINT** – Displays output on the screen.

BASIC Characters

- Letters (A-Z)
- Digits (0-9)
- Special characters (+, -, *, /, =, ,, ;, :)
- Keywords and reserved words like PRINT, LET, INPUT, etc.

Arithmetic Operators

Operators used in BASIC programming include:

I. **Addition (+)**

II. **Subtraction (-)**

III. **Multiplication (*)**

IV. **Division (/)**

V. **Exponentiation (^)**

BASIC Arithmetic Expressions

Expressions are formed using arithmetic operators and numbers.

Example expressions:

- A = 5 + 3
- B = (10 - 2) * 4
- C = 2^3

Evaluation of Arithmetic Expressions

Demonstrate how to evaluate expressions using **BODMAS** (Bracket, Order, Division, Multiplication, Addition, Subtraction).

Example:

    X = 2 + 3 * 4

    X = 2 + 12

    X = 14

Writing a Simple BASIC Program

Example program 1:

```
10 PRINT "Enter two numbers"

20 INPUT A, B

30 LET SUM = A + B

40 PRINT "The sum is ", SUM

50 END
```

Example program 2:

Here's a simple BASIC program that prompts the user to enter two numbers and then calculates and displays their sum:

```
10 PRINT "Enter the first number: "

20 INPUT A

30 PRINT "Enter the second number: "

40 INPUT B

50 LET C = A + B
```

```
60 PRINT "The sum of "; A; " and "; B; " is "; C

70 END
```

Explanation of the program:

- Line 10: Prints a message asking the user to enter the first number.

- Line 20: Takes the first number as input and stores it in variable A.

- Line 30: Prints a message asking the user to enter the second number.

- Line 40: Takes the second number as input and stores it in variable B.

- Line 50: Calculates the sum of A and B and stores it in variable C.

- Line 60: Prints the result, displaying the sum of the two numbers.

- Line 70: Ends the program.

You can run this program on any BASIC interpreter or emulator.

Conclusion:

BASIC is an easy-to-learn programming language, helping students understand fundamental programming concepts. Mastering basic statements, arithmetic operations, and simple programs will build a strong foundation for learning more advanced languages.

Assignment

I. Write a BASIC program to calculate the average of three numbers entered by the user.

II. List and explain three different BASIC arithmetic operators.