

Unit 1

Programming Language

Fundamentals

By Er. Abinash Adhikari

Topics Covered:

1. Introduction to Program and Programming Language
2. Types of Programming Language
3. Language Translator
4. Program Error and Types
5. Program Design Tools

Introduction to Program and Programming Language

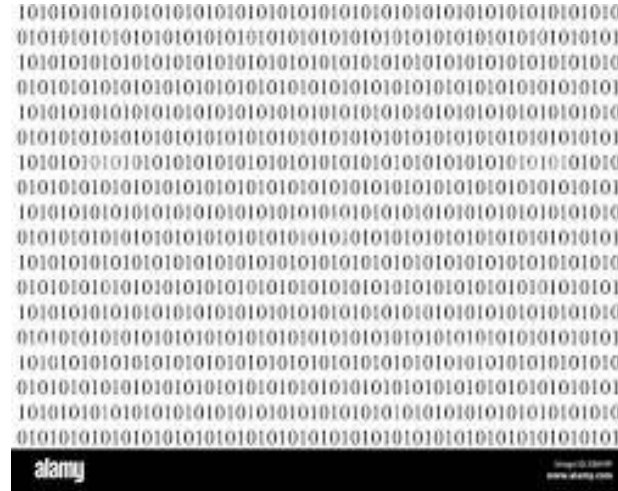
- ❖ Program:
 - A set of instructions for a computer to execute specific tasks.
- ❖ Programming Language:
 - A formal language used to write programs (e.g., Python, Java).
- ❖ Purpose:
 - Enables humans to communicate instructions to machines.



Introduction to Program and Programming Language

❖ Low-level language:

- Directly understandable by the computer.
- Examples:
 - Machine language, assembly language.
- Pros:
 - Efficient, precise control over hardware.
- Cons:
 - Difficult to learn, platform-dependent.



High-Level Language:

- ❖ More human-readable and easier to work with.
- ❖ Examples:
 - Python, Java, C++, JavaScript.
- ❖ Pros:
 - Easier to learn, portable, abstract away hardware details.
- ❖ Cons:
 - Less efficient than low-level languages.

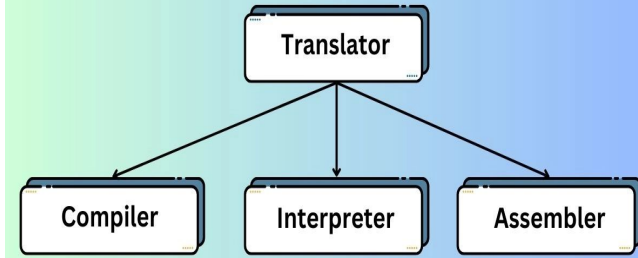


Aspect	Low-Level Languages	High-Level Languages
Abstraction	Close to machine language, with minimal abstraction.	High abstraction, closer to human language.
Syntax	Uses binary or assembly code, difficult to understand.	User-friendly syntax, easy to read and write.
Execution Speed	Very fast and efficient as it's closer to hardware.	Slightly slower as it needs to be translated into machine code.
Memory Management	Programmer handles memory management manually.	Memory management is automated by the language or runtime.
Portability	Platform-specific (not portable between different systems).	Highly portable across different platforms with minimal changes.
Ease of Development	Complex and requires deep knowledge of the hardware.	Easier and faster to develop, even for beginners.
Example Languages	Machine Language, Assembly Language.	Python, Java, C++, JavaScript.

Language Translator

❖ Assembler:

- Converts assembly language code into machine code.
- Assembly language is a low-level language that uses human-readable mnemonics to represent machine instructions.
- Assemblers are often used for system programming and device drivers.
- The output of an assembler is a machine code file that can be directly executed by the computer.



❖ **Compiler:**

- Translates high-level language code into machine code.
- High-level languages are more human-readable and easier to work with.
- Compilers typically generate optimized machine code for better performance.
- The output of a compiler is an executable file that can be run on the target platform.

❖ **Interpreter**

- Executes high-level language code line by line.
- Interpreters do not generate machine code directly, but rather execute the code directly in memory.
- This can lead to slower execution compared to compiled languages, but it can also provide more flexibility and easier debugging.
- Interpreters are often used for scripting languages and rapid prototyping.

Program Error

- ❖ Mistakes in the code that cause incorrect results or program crashes.
- ❖ Types of Errors:

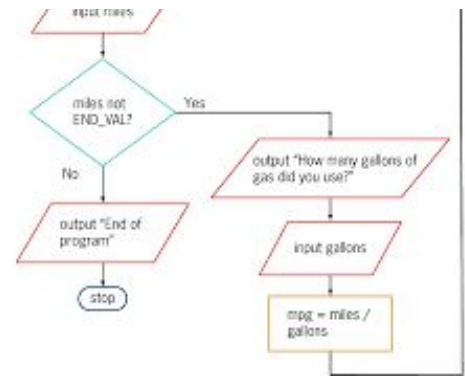
1. Syntax Error:

- Definition: Errors that violate the grammatical rules of a programming language.
- Examples:
 - Missing parentheses or brackets
 - Incorrect keywords or identifiers
 - Typos in code
- Detection: Usually detected by the compiler or interpreter during the compilation or interpretation process.

```
    textA.trim() === '' &&  
    textB.trim() === '' &&  
    touched.textA &&  
    touched.textB  
  }  
}  
  
setFieldValue(props.name, `textA: ${textA}, textB: ${textB}`);  
}, [textB, textA, touched.textA, touched.textB, setFieldValue, props.name]);  
  
return (  
  <input {...props} {...field} />  
  <::meta.touched && !meta.error && <div>[meta.error]</div>  
  />  
);  
}  
}  
  
function App() {  
  // Note that we provide initialValues all 3 fields.  
  const initialValues = { textA: '', textB: '', textC: '' };  
  return (  
    <div className="App">  
      <Formik  
        initialValues={initialValues}  
        onSubmit={async (values) => alert(JSON.stringify(values, null, 2))}
```

2. Semantic Errors

- ❖ Definition: Errors in the logic or meaning of the program.
- ❖ Examples:
 - Incorrect calculations or algorithms
 - Infinite loops
 - Incorrect variable usage
- ❖ Detection: More difficult to detect than syntax errors and often require careful testing and debugging



3. Runtime Errors

- ❖ Definition: Errors that occur during the execution of a program.
- ❖ Examples:
 - Division by zero
 - Out-of-bounds array access
 - Memory leaks
- ❖ Detection: Can be detected by the program itself or by the operating system.



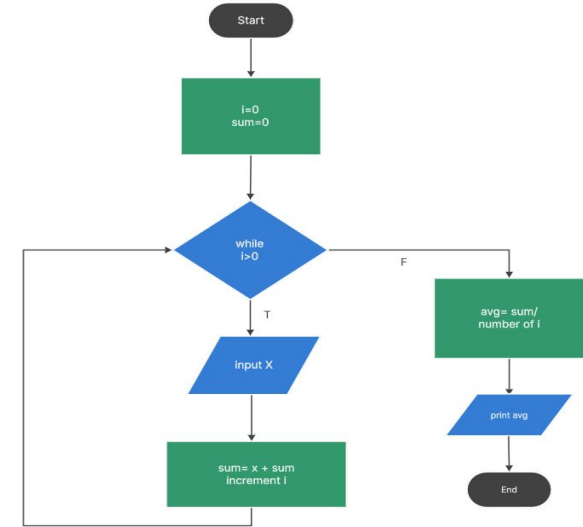
Program Design Tools

- ❖ Program design tools like algorithms and flowcharts help in planning and organizing the steps involved in solving a problem before coding.
- ❖ They improve the clarity and logic of the solution.
- ❖ Let's break down these tools with their properties, methods of writing, and examples.

Program Design Tools

1. Algorithms

- Algorithm is a step-by-step procedure to solve a problem.
- It's a sequence of instructions that performs a specific task or calculation.
- Algorithms are the foundation of computer programming, providing a clear and structured approach to problem-solving.








Properties of Algorithms

1. **Correctness:** An algorithm should produce the correct output for all valid inputs.
2. **Efficiency:** It should use computational resources (time and space) efficiently.
3. **Clarity:** The algorithm should be easy to understand and follow.
4. **Finiteness:** It should terminate after a finite number of steps.
5. **Definiteness:** Each step should be precisely defined and unambiguous.
6. **Effectiveness:** The steps should be feasible and executable.

Program Design Tools

2. Flow Chart

- Flowchart is a graphical representation of an algorithm.
- It uses various shapes connected by arrows to illustrate the sequence of steps and decisions in a process.
- Flowcharts are often used in software development, business processes, and other fields to visualize and understand complex procedures.

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectagle represents a process
	Decision	A diamond indicates a decision

Q. Write an algorithm and flowchart to find out the a circle. (hint : $A=\pi r^2$)

Algorithm

Step 1: Start

Step 2: Read the radius (r) of the circle.

Step 3: Calculate the area using the
formula: $A = \pi * r^2$

Step 4: Print the calculated area

Step 5: End

Flowchart

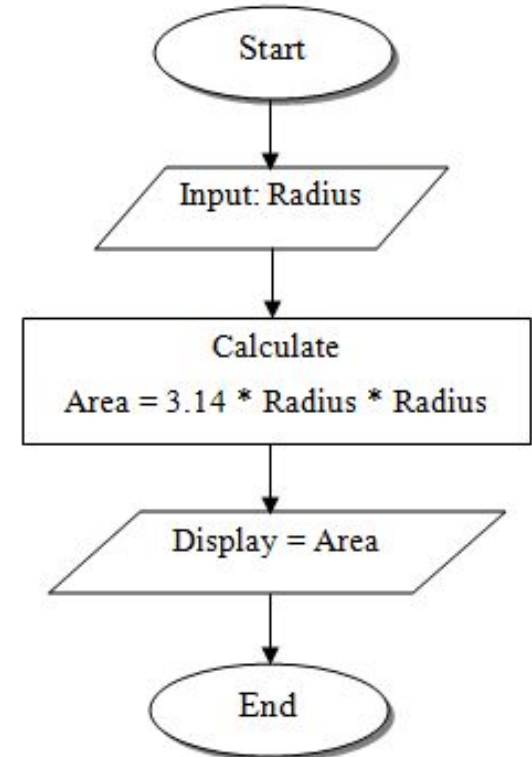


Fig. Flowchart to print Area of Circle

Assignments

1. What is Programming Language? Differentiate between compiler and Interpreter with an example.
2. Define Algorithm and Flowchart with an example.
3. Explain assembler, compiler and interpreter in brief.
4. What are the different types of errors that can occur in the C programming language? Explain the types of programming language.
5. Write an algorithm and flowchart to find out simple interest.(Hint: $SI = P \cdot T \cdot R / 100$)
6. Write an algorithm and flowchart to convert the temperature of centigrade into fahrenheit. ($F = 9/5 \cdot (C + 32)$) ($C = 5/9 \cdot (F - 32)$)
7. Write an algorithm and flowchart to find out the area of rectangle.
8. Write an algorithm and flowchart to convert given time in second into hour, minute and second.
9. Write an algorithm and flowchart to check a given number is odd or even.
10. Write an algorithm and flowchart to find out the middle number among three numbers.