Programming Concepts and Languages

Structured Programming - CIS122

Programming Concepts

A program is a set of instructions telling a computer how to perform various tasks.

- A programming language is used by programmers to create a program.
- Programs are also referred to as source code
- Coding is the act of writing source code.

Language Characteristics and Classifications

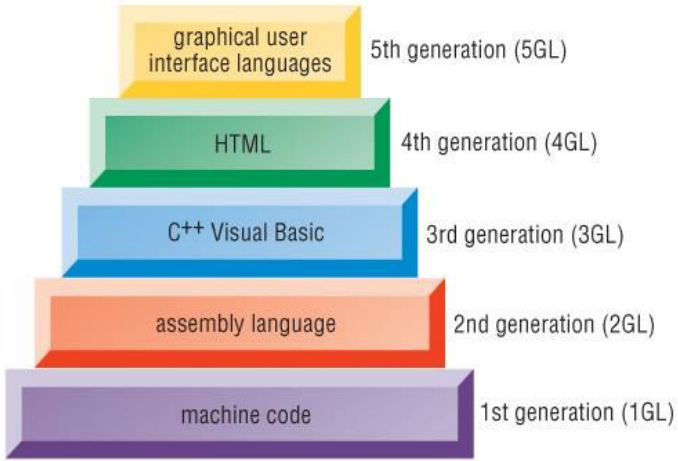
- Programming languages contain smaller vocabularies than human languages.
- Programming language syntax, or structure, also tends to be less complex.
- A program with a single syntax error (a mistake in the way programming elements are strung together) will not work at all.

High-Level vs. Low Level Languages

- A low-level language is a binary language consisting of 1s and 0s.
  - It is also known as machine code.
  - It runs faster and takes up less disk space.
- A high-level language is relatively similar to natural languages such as English.
  - It is easier to learn and use

Computer Programming Language Generation

A computer programming language generation is a group of languages that were developed at the same time. Each generation builds on the contributions of the one before it.

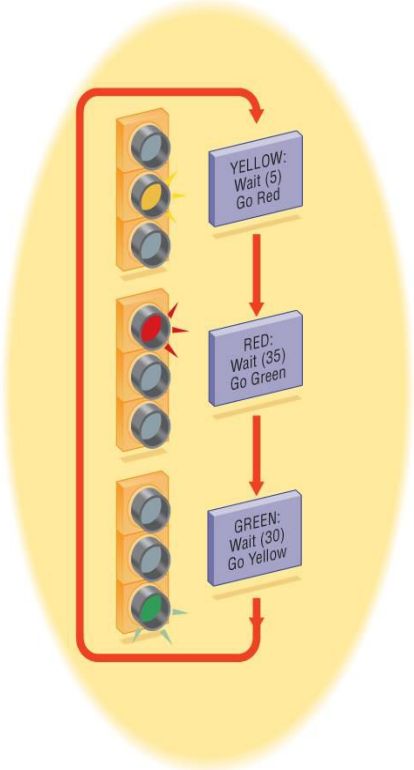


Classic Programming Elements

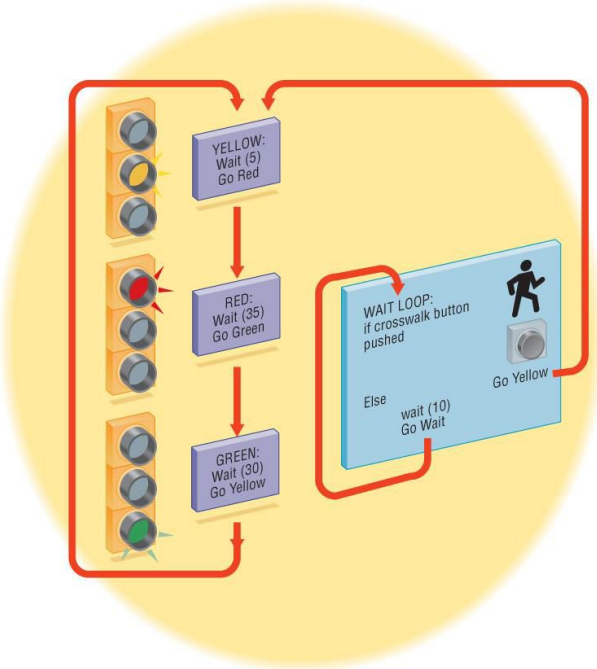
- **Variables** data values stored in computer memory
- **Executable statements** perform actions and proceed to the next statement in the sequence
- **Looping** allows a program to return to a previously executed instruction and repeat it
- **Decision statements** points in a program where different actions may be performed depending on specific conditions

Looping

Looping allows a computer program to continuously repeat the same steps, such as a program designed to direct a traffic light to display yellow, red, and green lights at a consistent rate.



Decision Statement's Effect on Looping



Using an **if then** statement , *based on a particular action*, such as pushing the crosswalk button, a program can interrupt the looping pattern, making the program more useful.

Problem Solving Techniques

In the **divide and conquer** approach , programmers tackle one small piece of the puzzle at a time.

- The **top down design** approach helps programmers break a large project into manageable parts.

Problem Solving Steps

1. Identify the problem.
2. Analyze the problem
3. Brainstorm solutions and choose the best one.
4. Write the algorithm.
5. Prototype the solution.
6. Implement and test the solution.

Algorithm

An algorithm is a complete list of steps for solving a problem.

- Algorithms are written in pseudocode
- The step by step pseudocode algorithm for changing a lightbulb is given at the right.

STEP 1: Get stepladder from closet.
STEP 2: Place stepladder under fixture.
STEP 3: Turn switch off.
STEP 4: Climb up ladder.
STEP 5: Remove old lightbulb from socket.
STEP 6: Climb down ladder.
STEP 7: Put old lightbulb into trash can.
STEP 8: Get new lightbulb from cupboard.
STEP 9: Climb up ladder.
STEP 10: Put new lightbulb into socket.
STEP 11: Climb down ladder
STEP 12: Turn switch on.
STEP 13: Put stepladder into closet.

The Evolution of Programming Approaches

In the process of coding, the lines of code keep multiplying and thus, the size of the software increases. Gradually, it becomes impossible to remember the flow of the program.

- It becomes difficult to share, debug or modify the program.

Solution: Structured Programming

- Uses subroutines and loops instead of go-to statements and reduces coding time.

Structured Programming

- This type of programming presents guidelines for an *organized*, *logical approach* to programming.
- The programmer thinks in terms of structured groups of instructions built on **routines** , which are sections of programs that handle specific functions.
- Routines are then broken down into steps.

Modules

In **modular code** , programmers create code **modules** that handle the separate components of a program.

- Modules are reusable and help in tracking down the source of errors.
- Modularity describes how well source code is divided into individual modules.

– A **macro** is a recording of steps to perform a repetitive activity.

Object Oriented Programming

Object oriented programming (OOP) defines each module (object) with definite rules for interfacing and a protected set of variables.

- Protected variables allow a programmer to prevent data from being altered during program execution.

Rapid Application Development

Rapid application development (RAD) reduces cost by decreasing time needed to develop a project.

Programmers using RAD follow these guidelines:

- Use visual development tools whenever possible.
- Rapidly prototype new projects in order to reduce redesign time.
- Approach coding with these priorities:
  - Use existing code first.
  - Buy someone else's existing code second.
  - Write new code last.

Development & Documentation Tools

Compilers and Interpreters

– A **compiler** is a program that translates programming language source code into machine code.

– An **interpreter** translates instructions one by one as the source code is being executed, rather than all at once.

- It identifies errors as they are encountered.

Difference between Interpreter & Compiler

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

Debuggers

– A bug is a computer error.

– A debugger is a software tool that helps programmers find errors quickly.

– Debuggers are an integral component of compilers and interpreters.

Documentation Tools

The written notes that explain how a program works are called documentation.

- A flowchart provides a visual diagram of an algorithm.
- CASE tools help a programming team schedule and coordinate its operations.
- A comment is an informational message inserted into program source code to explain it to later readers.

Flowchart

The symbols in Fig. 1 are used in flowcharts to represent the logic of a program.

In Fig. 2, those symbols are used to show the logic of a program that from 1 to 10.

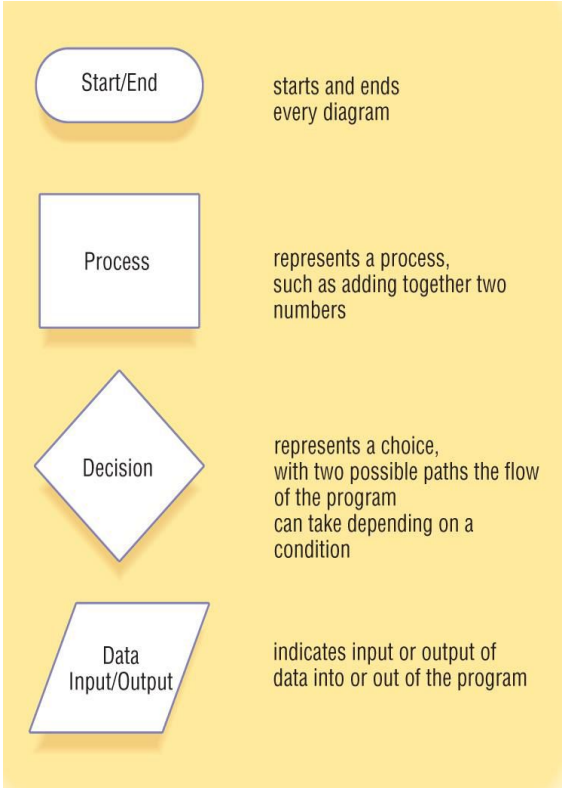


Fig. 1

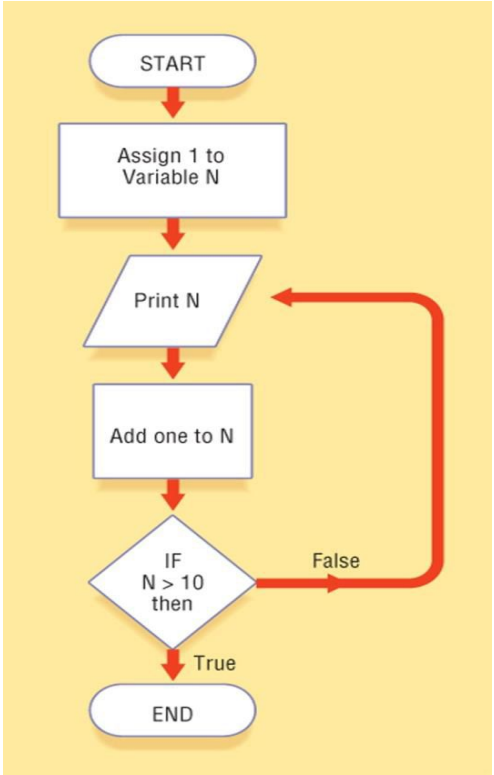


Fig. 2

Programming Errors

What are the main types of program errors?

- A **syntax error** (compiler error) is usually due to a typing mistake or a misunderstanding of the rules of a language.

- A **logic error** occurs when a program's syntax is correct, but the program instructs the computer to perform an action incorrectly.

Programming Errors

- A **run time error** refers to mistakes that occur when the application is running.
    - A crash bug causes a program to stop running.
    - An infinite loop causes a program to perform the same set of instructions over and over.
- A **style error** occurs as a result of poorly written programming code.
    - Dead code makes source code hard to read.

Commonly Used Programming Languages

Language	Compiled	Interpreted	Object-Oriented
C	Yes	No	No
C++	Yes	No	Yes
Java	Yes	Yes	Yes
Visual Basic	No	Yes	No