INTRODUCTION TO PROGRAMING

Chapter 1

An Overview of Computers and Programming Languages

GV: Thái Hùng Văn

Khoa Công Nghệ Thông Tin
Trường Đại Học Khoa Học Tự Nhiên
ĐHQG-HCM

Objectives

In this chapter, you will:

- Learn about different types of computers.
- Explore the hardware & software components of a computer system.
- Learn about the language of a computer.
- Learn about the evolution of programming languages.
- Examine high-level programming languages.
- Discover what a compiler is and what it does.
- Examine C /C++ programs
- Examine how a C/C++ program is processed.
- Learn what an algorithm is and explore problem-solving techniques.

Outline

- Definitions & basic concepts
- Elements of a Computer System
- Programming Languages
- Software Program Development
- Examine Programs (in Scratch, C /C++ …)

Definitions & basic concepts

Definitions & basic concepts

- A **computer** is a machine or device that performs processes, calculations and operations based on instructions provided by a **program**.
- Computers are designed to execute applications and provide solutions by combining integrated **hardware** and **software** components. They are used as control systems for a wide range of industrial and consumer devices.
- A "complete" computer including the hardware, the operating system (main software), and necessary peripheral equipment can be called a **computer system**.
- Today, the **IoT** allows connecting computer devices to the Internet and to other connected devices. Many computers can work together to share information, operate as design.

Definitions & basic concepts

- Computers have greatly affected our daily lives - helping us complete an extremely wide range of tasks by **software programs**
- **Program** is a set of step-by-step instructions that tells or directs the computer what to do. It sequences the tasks a user wants to be done and produces the results or output needed
- Computer programs (**software**) are designed specifically for each task.
- A **programmer** is the person who designs a program, converts problem solutions into instructions for the computer.
- Computer programs are created with programming languages.
- A **programming language** is a formal language, comprises a set of instructions that produce various kinds of output, used to implement **algorithms** to perform specific tasks.
- There are various types of programming language you can choose from. **C /C++** is an example of a programming language.

Generations of Computer

- The development of electronic computers can be divided into 5 generations (until 2019) depending upon the technologies used:
  - 1st gen: vacuum tube (~1940-1956)
  - 2nd gen: transistor (~1956-1963)
  - 3rd gen: integrated circuits (IC) (~1964-1971)
  - 4th gen: microprocessor (~1971-2010)
  - 5th gen: artificial intelligence (AI) (~2010 to present)
- Future generations might be **quantum computers**. It has been figuring out algorithms at 10^8 times the speed that a traditional computer can, and that could make a huge difference in the processing power at our disposal in the future.

[https://www.sciencealert.com/google-s-quantum-computer-is-100-million-times-faster-than-your-laptop]

- [There are several new computer types that can be used in the future: Quantum, Chemical, DNA, Optical, Spintronics-based, Wetware/Organic computer, etc]

Etymology of “Computer”

- "Computer" comes from the Latin "**putare**" which means both to **think** and to **prune.** Computare (**com -** means "**together**") also meant **calculate**. [https://www.bbc.com/news/blogs-magazine-monitor-35428300]
- The ***Online Etymology Dictionary*** gives the first attested use of "computer" word in the 1640s, meaning "**one who calculates**“.
- It states that the use of the term to mean "**calculating machine**" (is from 1897.
- It indicates that the "modern use" of the term, to mean "**programmable digital electronic computer**" dates from 1945 under this name
- Now, a computer is a **machine or device** that **performs processes by a program.**

Elements of a Computer System

Computer Types

By size and form-factor

- Mainframe computer
- Supercomputer
- Minicomputer
- Microcomputer
- Workstation
- Personal computer
- Laptop
- Tablet computer
- Smartphone
- Single-board computer

By achitecture

- Analog computer
- Digital computer
- Hybrid computer
- Harvard architecture
- Von Neumann architecture
- Reduced instruction set computer

By purpose

- General-purpose computer
- Special-purpose computer

Hardware

- The term ***hardware*** covers all of those parts of a computer that are tangible physical objects.
- Circuits, chips, graphic cards, sound cards, RAM, mobo, displays, PSUs, cables, printers, keyboards, and "mice" input devices are all hardware.
- A **general purpose computer** has 4 main components: **ALU, CU, memory, I/O devices**. These parts are interconnected by **buses** (wires or build-in board)
- Inside these parts are thousands to trillions of **electrical circuits** that can be turned on / off (means bit 1 /bit 0) by electronic **switches**.
- The circuits are arranged in **logic gates** so that one or more circuits can control the state of other circuits.

Hardware - Input devices

- When unprocessed data is sent to the computer with the help of input devices, the data is processed and sent to output devices.
- The input devices may be hand-operated or automated. The act of processing is mainly regulated by the CPU.
- Some examples of input devices are:
  - Keyboard
  - Mouse /Trackball /TouchPad /TouchScreen
  - Microphone
  - Digital camera /video recorder /webcam
  - Sanner
  - Joystick
  - Air Mouse (with Keyboard and Mic)

Hardware - Output devices

- An output device is any peripheral that receives data from a computer.
- Output /input devices are all peripheral hardware, and they are usually connected to a computer by cables, or wireless networks *(most of them use the 2.4 GHz band as the preferred frequency range for communication)*
- Some examples of input devices are:
  - Monitor / TV /Video card
  - Speaker /Headphone /Sound card
  - Projector
  - Printer

Hardware – I/O Devices Quiz

1. How to use multiple monitors on a computer (PC) ?
2. How to use one monitor on multiple computers ?
3. How to use one speaker on multiple computers ?
4. How to use one printer on multiple computers ?
5. How to use one mouse and keyboard on multiple computers ?
6. How to remote control other computers ?
7. How to use the mouse and keyboard on smartphone ?
8. How to convert a wired I/O device to wireless device?

Hardware - ALU

- The set of arithmetic operations that a particular ALU supports may be addition, subtraction, multiplication, division, sine, cosine, square roots, etc. Some can only operate on integers.
- However, any computer can be programmed to perform any arithmetic operation *(but it will take more time to do so if its ALU does not directly support)*.
- An ALU may also compare numbers or perform other logic operations (And /Or /Xor /Not) and return boolean truth values.
- Computers may contain multiple ALUs, allowing them to process several instructions simultaneously.
- Graphics processors and computers with SIMD and MIMD features often contain ALUs that can perform arithmetic on vectors and matrices.

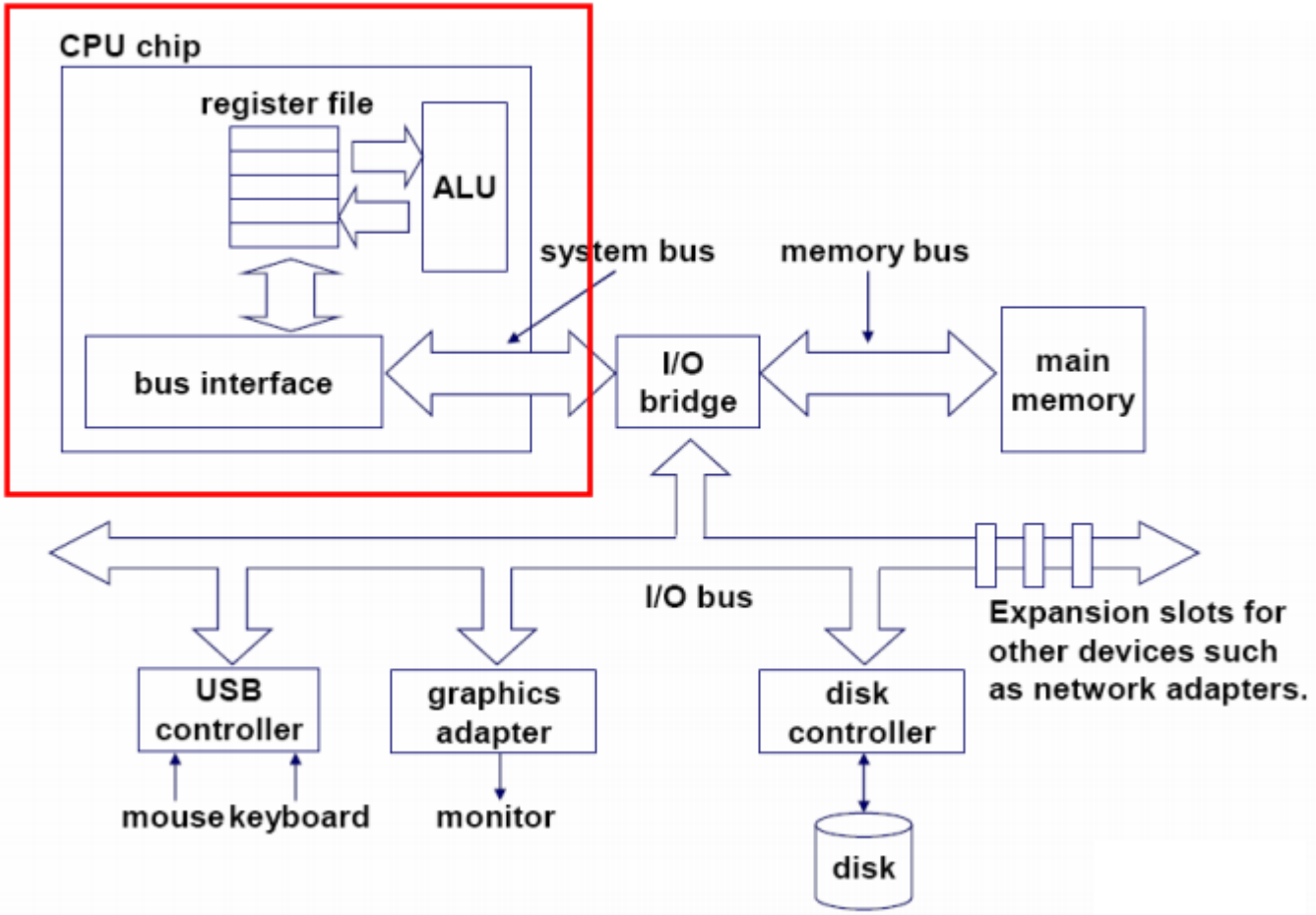
Hardware - CU

- The CU (control system /central controller) reads and interprets (decodes) the program instructions, transforming them into control signals that activate other computer parts
- A key component is the program counter (PC), a special register that keeps track of which location in memory the next instruction is to be read from.
- The control system's function is as follows:
  - Read the code for the next instruction by the PC.
  - Decode the numerical code for the instruction into a set of commands
  - Increment the PC so it points to the next instruction.
  - Read whatever data the instruction requires (the location stored within code)
  - Provide the necessary data to an ALU or register.
  - If the instruction requires an ALU or specialized hardware to complete, instruct the hardware to perform the requested operation.
  - Write the result from ALU back to memory location /register /output device.
  - Jump back to step (1).

Hardware - CPU

- The CU, ALU, and registers are collectively known as a central processing unit (CPU).
- Since the mid-1970s CPUs have been constructed on a single IC called a **microprocessor**.
- Nowadays, general-purpose computers often use **multi-core processor** (a single IC that contains **two or more separate processing units**, called **cores**)
- Each core can read and execute program instructions, as if the computer had several processors.

Hardware – Single-core computer



Hardware - Memory

- The computer's memory can be viewed as a list of cells where numbers can be placed or read.
- Each cell has a numbered "address" and can store a single number.
- In almost computers, each memory cell is set up to store binary numbers in groups of 8 bits (called a **byte**).
- To store large numbers, several consecutive bytes may be used (typically, 2, 4 or 8).
- The computer can store any kind of information in memory (if they can be represented numerically)
- Modern computers have billions or even trillions of bytes of memory.

Hardware – Memory - types

- **RAM** (Random-Access Memory) is the main memory with big size (usually, in GB). It can be read and written anytime by CPU. The data in RAM is lost when the computer is shut off or restart.
- **ROM** (Read-Only Memory) is typically used to store the computer's initial start-up instructions.
- **Registers** are the smallest and the fastest storage. They are organized inside the CPU and provide the fastest way to access data.
- (CPU) **cache** is a type of cache memory that a processor uses to access data and programs much more quickly than through host RAM. It enables storing and providing access to frequently used programs and data.

Software

- Software consists of programs written to perform specific tasks.
- Two types of programs:
  - System programs
  - Application programs
- System Programs
  - System programs control the computer.
  - When computer turn on, the operating system (OS) is load and manages all of the other application programs.
- Application programs
  - The application programs make use of the OS by making requests for services through a defined application program interface (API).

Programming Language

Computer Language

- Machine language is the most basic language of a computer.
- Machine instructions are sequences of 0s and 1s.
- Every computer directly understands its own machine language.

➔ Very difficult for programmers

Assembly Language

- Early computers programmed in machine language (ML) ➔ so difficult to programming
- Assembly languages (AL) were developed to make programmer's job easier.
- In AL, an instruction is an easy-to-remember form called a mnemonic.
- Assembler: Translates AL instructions into ML

High-level Language

- Programming in AL is still difficult and inconvenient
- High-level languages (HLLs) make programming easier. Closer to spoken languages.
- There are many HLLs in use:
  - C
  - C++
  - C#
  - Java
  - Java Script
  - Python
  - Golang

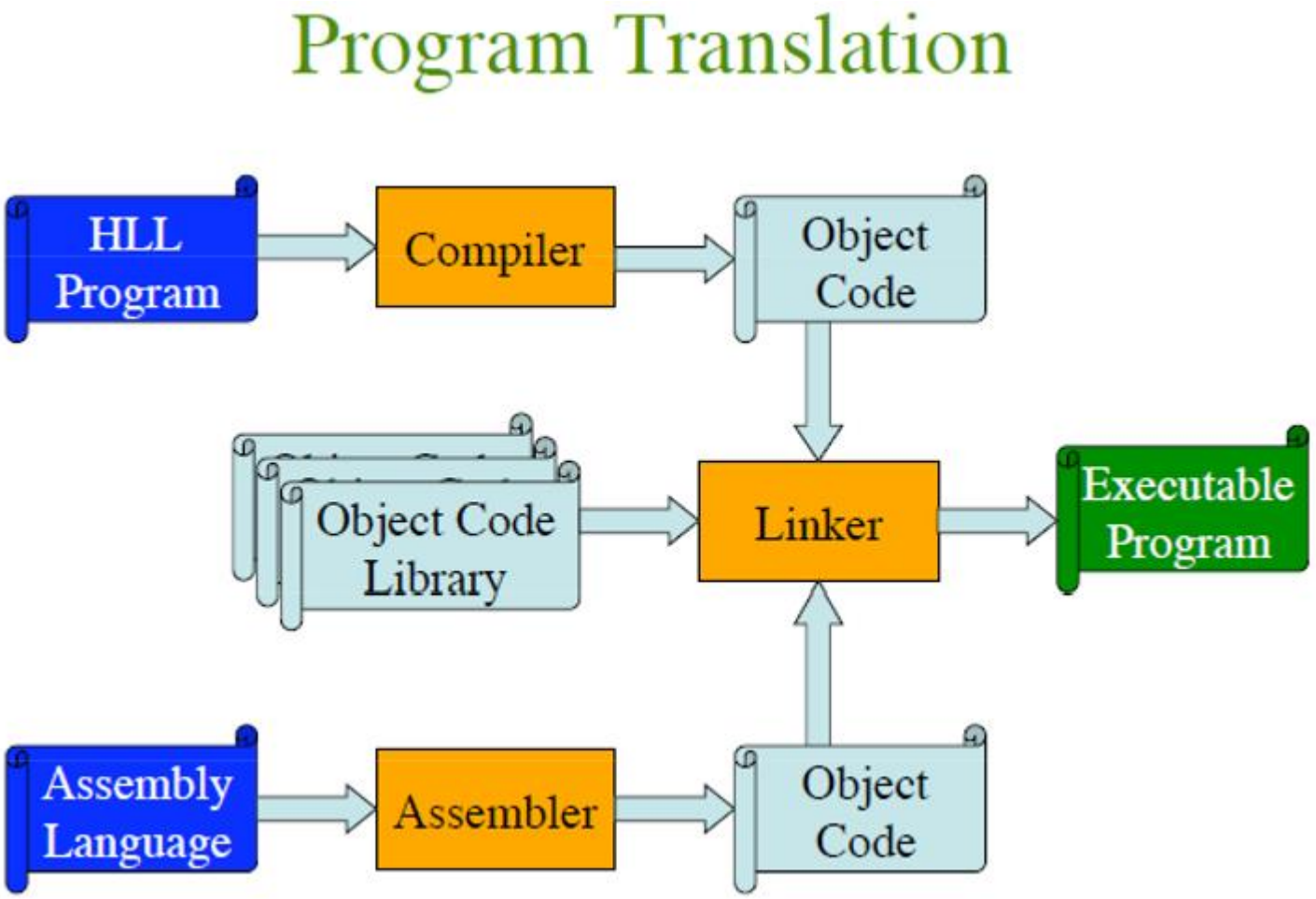
Evolution of Programming Languages

1. ML or First Generation Programming Language (GPL) – lowest level of programming.
2. AL or Second GPL – considered as low level language uses Mnemonic codes (abbreviations that easy to remember).
3. HLL or Third GPL – language is written in English like manner.
4. Very HLL or Fourth GPL
5. Natural Language – Fifth GPL resemblance to English language.

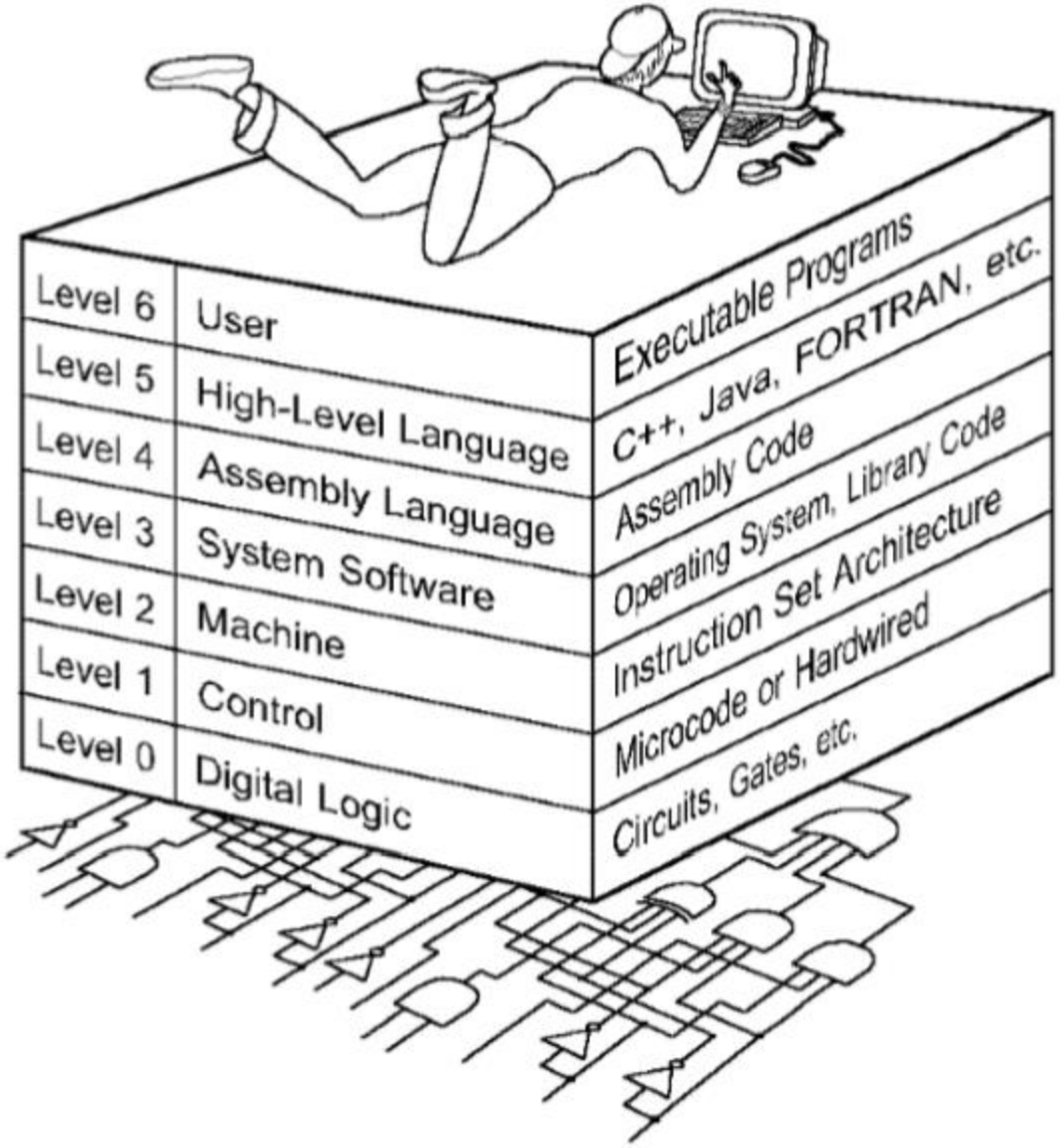
Compiler – Interpreter - Assembler

- Computer only understands ML instructions. Instructions written in AL or HLL must be translated to ML instructions
- Assembler: One AL instruction is mapped to one ML instruction
- Compiler /Interpreter: translates a program written in a HLL into the equivalent ML, generally one HLL instruction is mapped to many ML instruction.
- Compiler translates all HLL program, Interpreter translates one by one

Program Translation Diagram

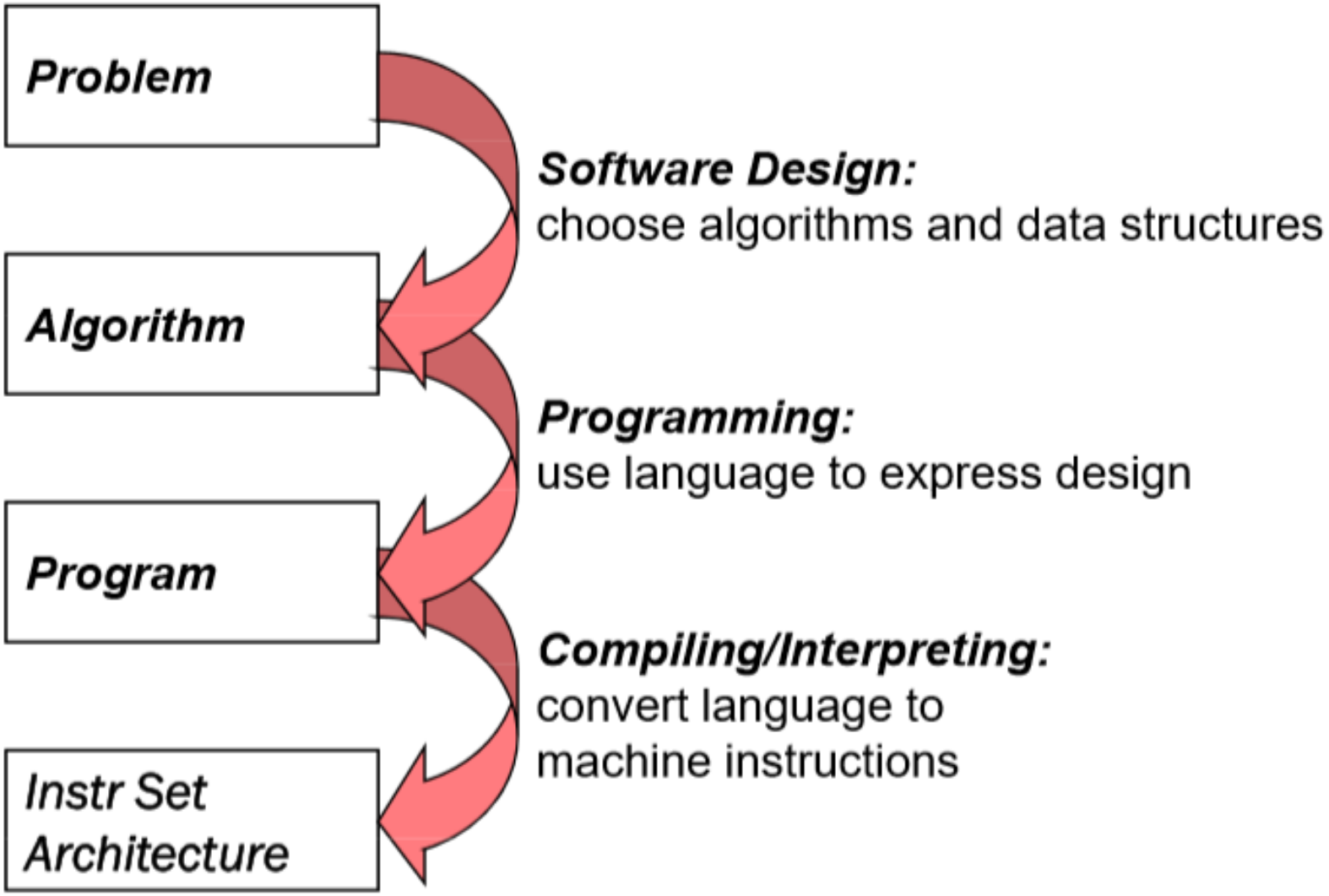


Computer Level Hierarchy



Software Program Development

Solving problem using computer



Program Development Life Cycle

- When we want to develop a program using any programming language, we follow a sequence of steps.
- The PDLC is a set of steps or phases that are used to develop a program. Generally, they are as follow:

1. ***Defining*** *the Problem*
2. ***Analyzing*** *the Problem*
3. ***Designing*** *the Program*
4. ***Coding*** *the Program*
5. ***Testing & Debugging*** *the Program*
6. ***Documenting*** *the Program*
7. ***Deploying*** *and* ***Maintaining*** *the Program*

PDLC # *Defining the Problem*

- The first step is to define the problem. In this phase we need to understand the problem statement, what is our requirement, what should be the output of the problem solution.
- In major software projects, this is a job for system analyst, who provides the results of their work to programmers in the form of a program specification.

PDLC # *Analyzing the Problem*

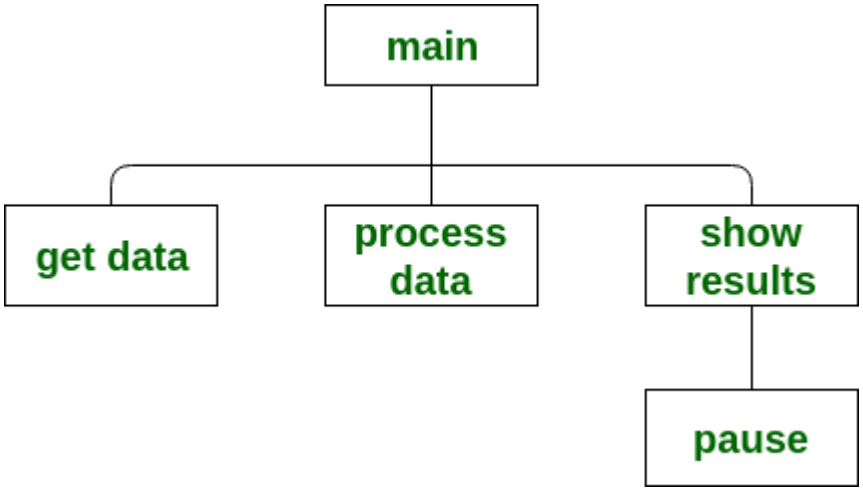
- Read the case thoroughly.
- Define the central issue.
- Define the firm's goals.
- Identify the constraints to the problem.
- Identify all the relevant alternatives.
- Select the best alternative.
- Develop an implementation plan.

PDLC # *Designing the Program*

- Starts by focusing on the main goal and then breaking the program into manageable components, each of which contributes to this goal. This approach is called ***top-bottom program design*** or ***modular programming***.
- Identify ***main routine***, which is the one of program's major activity.
- Try to divide the various components of the main routine into smaller parts called ***modules***.
- For each module, programmer draws a conceptual plan using an appropriate **program design tool** to visualize how the module will do its assign job

1. **Structure Charts**, also called *Hierarchy chart*

- Show top-down design of program.
- Each box in the structure chart indicates a task that program must accomplish.
- The Top module, called the *Main module* or *Control module*.



2. Algorithms

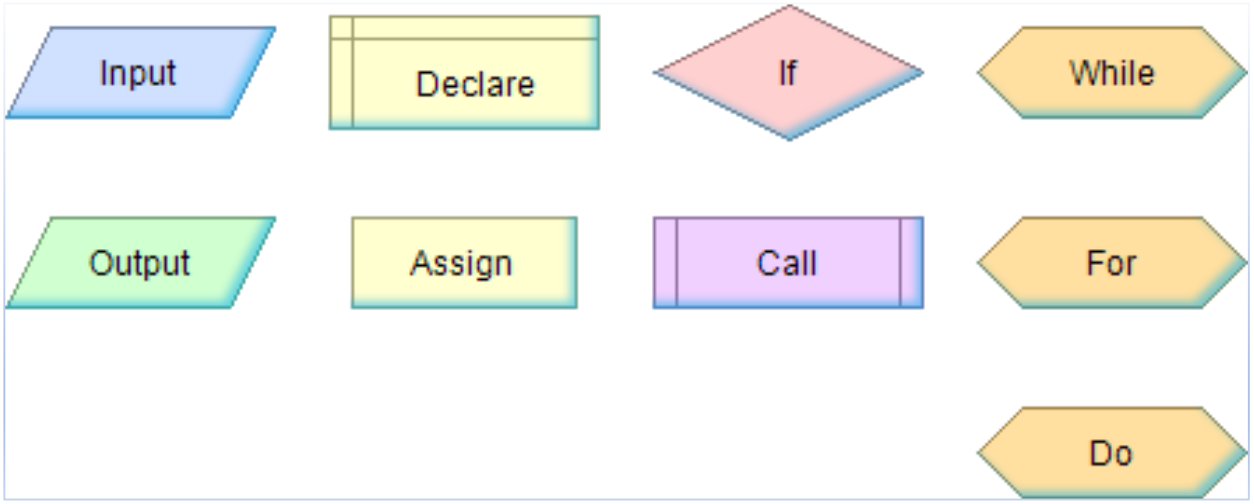
- An ***algorithm*** is a step-by-step description of how to arrive at a solution in the most easiest way.
- Algorithms are not restricted to computer world only. In fact, we use them in everyday life.

Example: ***Algorithm to find roots of a quadratic equation aX^2+bX+c =0.***

```
Step 1: Input a, b, c
Step 2: If a = 0: Solving linear equation bX+c =0 and return
Step 3: Calculate discriminant (=b^2-4ac)
Step 4: If discriminant < 0
                    return No solutions
        Else
                    r1 ← (-b+√D)/2a
                    r2 ← (-b-√D)/2a
Step 5: Stop
```

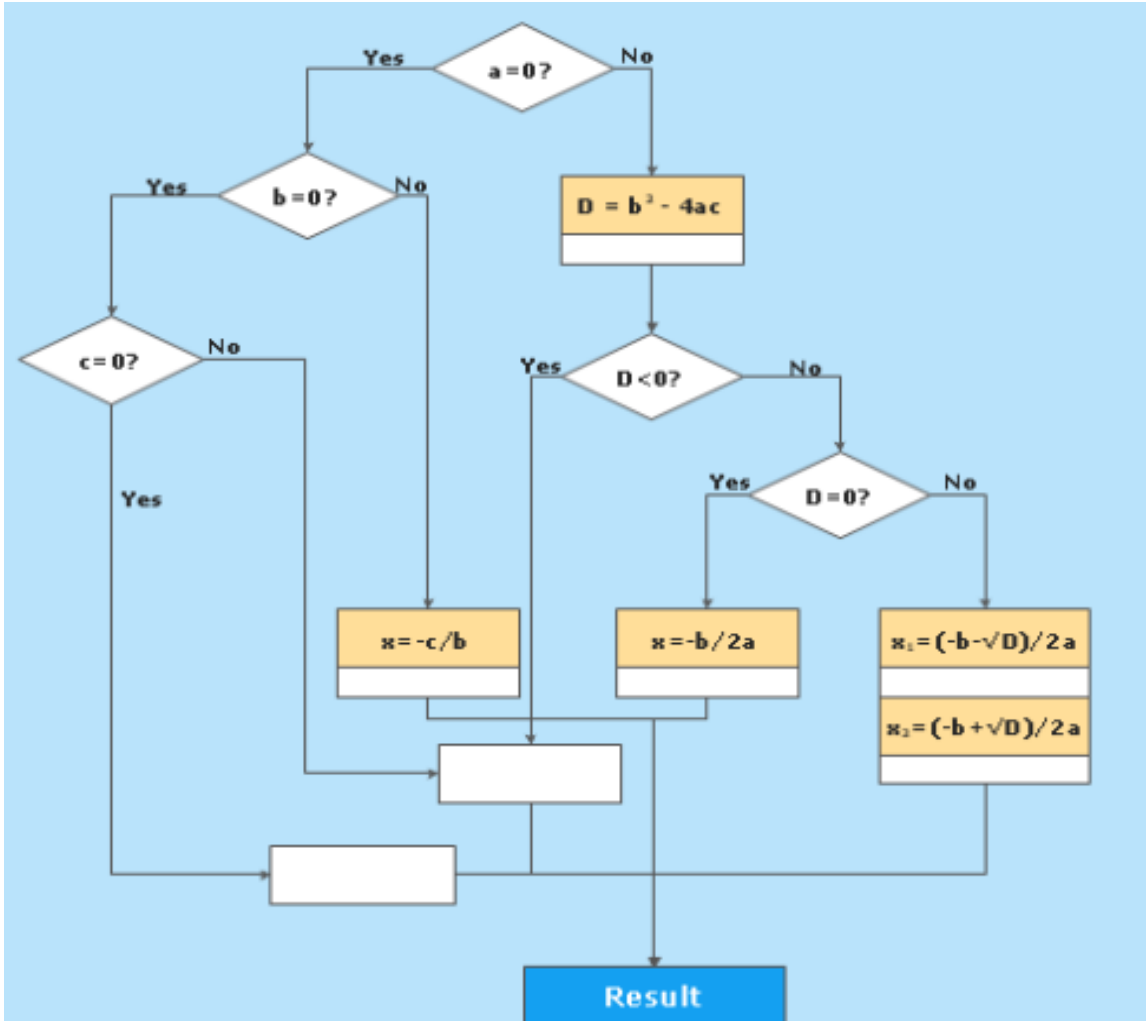
3. Flowcharts

- It is a visual diagram that shows the logic of the program.
- It will help you break down the problem, and very important for teamwork.
- Common symbols used in flowcharts:



3. Flowcharts -

Example: ***Flowchart to find roots of the equation aX^2+bX+c =0.***



4. Pseudocode

- It is another tool to describe the way to arrive at a solution.
- It is expressed based on the program language..

Example: ***Pseudocode to find roots of quadratic equation aX^2+bX+c =0.***

```
Input a, b, c
If (a = 0)
    call fuction_SolvingLinearEquation (b , c)
Else
    Calculate discriminant D ← b2-4ac
    If D < 0
            return No solutions
    Else
            r1←(-b+√D)/2a
            r2←(-b-√D)/2a
    EndIf
EndIf
```

PDLC # *Coding the Program*

- Coding the program means translating an algorithm into specific programming language.
- That means we write the actual program to solve the given problem!
- The technique of programming using only well defined control structures is known as *Structured programming*.
- Programmer must follow the language rules, violation of any rule causes *error*. These errors must be eliminated before going to the next step.

PDLC # *Testing & Debugging the Program*

- After removal of syntax errors, the program will execute. However, the output of the program may not be correct. This is because of logical error in the program.
- A logical error is a mistake that the programmer made while designing the solution to a problem. So the programmer must find and correct logical errors by carefully examining the program output using *Test data*.
- Syntax error and Logical error are collectively known as ***Bugs***. The process of identifying errors and eliminating them is known as ***Debugging***.

PDLC # *Documenting the Program*

- After testing, the software project is almost complete. The *structure charts*, *pseudocodes*, *flowcharts*, *decision tables* developed during the design phase become documentation for others who are associated with the software project.
- This phase ends by writing a manual that provides an overview of the program's functionality, tutorials for the user, explanations of major program features, reference documentation of all program commands and a description of the error messages generated by the program.

PDLC # *Deploying and Maintaining the Program*

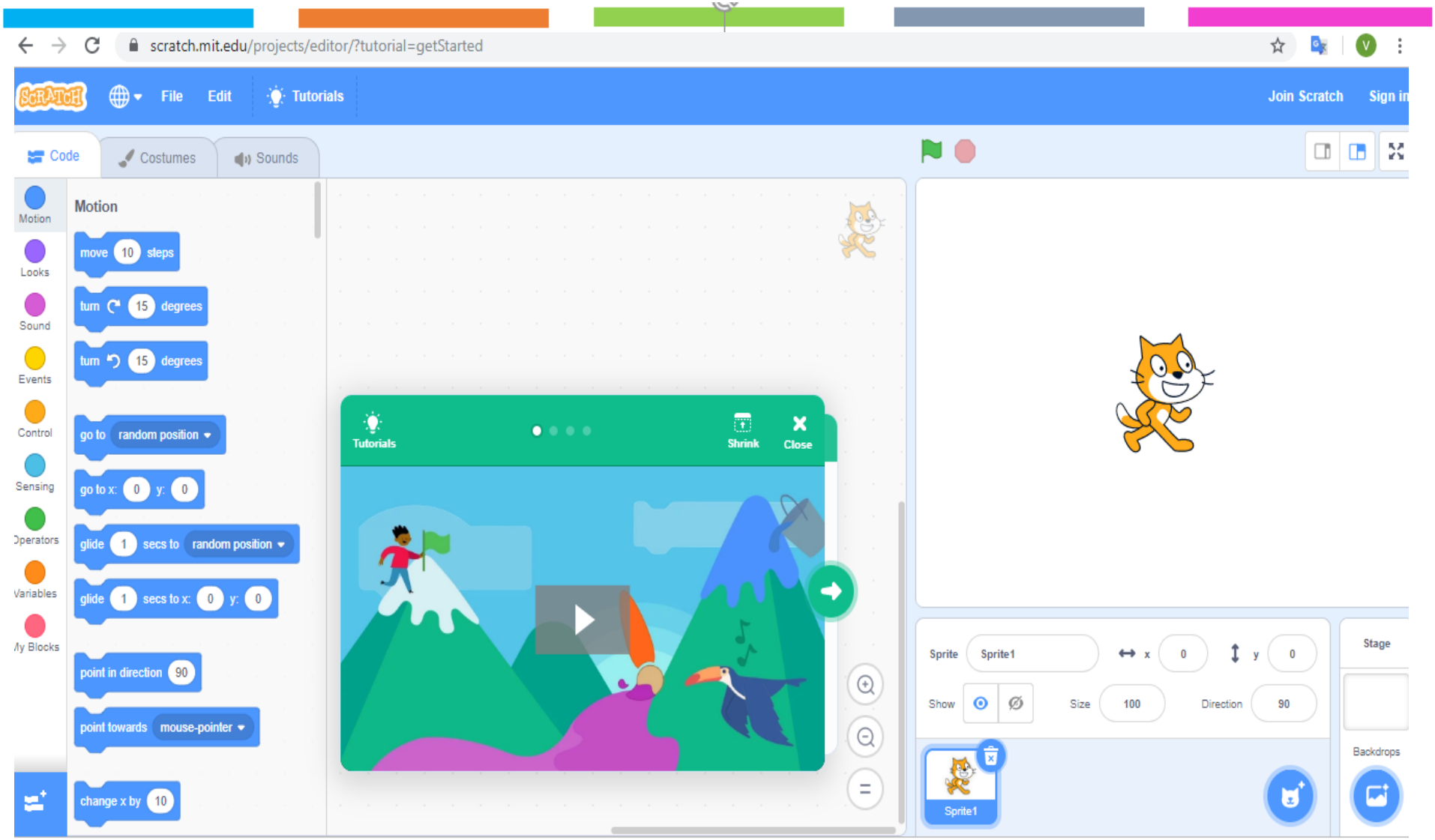
- In the final phase, the program is deployed (installed) at the user's site. Here also, the program is kept under watch till the user gives a green signal to it.
- Even after the software is completed, it needs to be maintained and evaluated regularly. In software maintenance, the programming team fixes program errors and updates the software.

Examine Programs

IDE - Integrated Development Environment

- IDE is an application that makes programming easier!
- IDEs increase programmer productivity by combining common activities of writing software : **editing source code**, **building executables**, and **debugging**.
  - syntax highlighting, autocomplete, autoXX
  - automated build process, compiling, executing code is abstracted away
  - debugging tools
  - hints while coding to prevent errors
  - allows you to code and run in virtual environments.
- The most popular IDEs: [https://www.infoworld.com/article/3217008/the-most-popular-ides-visual-studio-and-eclipse.html]
  - **Visual Studio**, 22.4%
  - Eclipse, 20.38%
  - Android Studio, 9.87%
  - Vim, 8.02 %
  - NetBeans, 4.75%

Scratch


scratch.mit.edu/projects/editor/?tutorial=getStarted
File
Edit
Tutorials
Join Scratch
Sign in
Code
Costumes
Sounds
Motion
Looks
Sound
Events
Control
Sensing
Operators
Variables
My Blocks
move 10 steps
turn 15 degrees
turn 15 degrees
go to random position
go to x: 0 y: 0
glide 1 secs to random position
glide 1 secs to x: 0 y: 0
point in direction 90
point towards mouse-pointer
change x by 10
Tutorials
Shrink
Close
Sprite
Sprite1
x
0
y
0
Show
Size
100
Direction
90
Stage
Backdrops
Sprite1

Scratch

- Scratch is a block-based visual programming language, developed by MIT. In Scratch you can create your own interactive stories, games, and animations **very easy**!
- You can drag and combine code blocks to make a range of programs, including animations, stories, musical instruments and games. It's a bit like the programming equivalent of LEGO!
- Try Scratch at https://scratch.mit.edu/
- Download, Install and Try Scratch Offline.
- Create some Scratch projects like [Demo]

Examine C/C++ Programs – using Online IDE

- Online IDE is Fast (*source code is compiled and run on the server*), Powerful (*applies for many programming languages*), and Free (*no charge to use*)
- Some to try:
  - https://www.codechef.com/ide
  - https://www.onlinegdb.com/
  - https://repl.it/languages/c

Ex01: Edit and Run the following source code:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello World";

    return 0;
}
```

Examine C/C++ Programs – using IDE Apps for Mobile

- Fast (compile and run on server), Powerful, many Free Apps!
- Some to try:
  - Mobile C (C/C++ Compiler) - Android Apps on Google Play
  - C Compiler App for IOS
  - C 4 Droid
  - CppDroid
  - Dcoder

Ex02: Run this program on **your phone.** Investigate what happened

```
#include <iostream>
using namespace std;
int main() {
    cout << "operator test: \n";
    cout<<1+2 <<endl <<1-2 <<endl <<1*2 <<endl <<1/2;
    return 0;
}
```

Examine C/C++ Programs – Using Visual Studio IDE

- VS is still the Best IDE for Windows (*for more than 2decades*)
- There are many versions of VS. You can use any of them (*from 2010 to 2019 – VS Community Edition is free*)

Ex03: Create a solution /project *Win32 Console Application* , Edit, Build and Run the following source code:

```
#include <iostream>
using namespace std;
int main() {
    int firstNumber, secondNumber, sumOfTwoNumbers;

    cout << "Enter two integers: ";
    cin >> firstNumber >> secondNumber;
    // sum of two numbers in stored in variable sumOfTwoNumbers
    sumOfTwoNumbers = firstNumber + secondNumber;
    // Prints sum
    cout << firstNumber << " + " <<  secondNumber << " = " << sumOfTwoNumbers;
}
```

Examine C/C++ Programs – Using Visual Studio IDE

Ex04: Add a project to the previous solution, with the following program:

```
#include<iostream>
using namespace std;
int main() {
   float a, b, c, d, X;
   cout<<"The form of the linear equation in one variable is: aX + b = cX + d"<<endl;
   cout<<"Enter the values of a, b, c, d : "<<endl;
   cin>>a>>b>>c>>d;
   cout<<"The equation is "<<a<<"X + "<<b<<" = "<<c<<"X + "<<d<<endl;

   if(a==c && b==d)
      cout<<"There are infinite solutions possible for this equation"<<endl;
   else if(a==c)
      cout<<"This is a wrong equation"<<endl;
   else {
      X = (d-b)/(a-c);
      cout<<"The value of X = "<< X <<endl;
   }
}
```

Examine C/C++ Programs – Using Visual Studio IDE

Ex05: Add the 3rd project to solve the linear equation aX+b=0 - try it yourseft in 5 minutes!

Reference

- **Thinking in C**, Bruce Eckel, E-book, 2006.
- **Theory and Problems of Fundamentals of Computing with C++**, John R.Hubbard, Schaum's Outlines Series, McGraw-Hill, 1998.