What is Programming?

- Planning or scheduling the performance of a task.
- Consciously thinking about each step
- Example: Accelerating in a car
  1. Move right foot to gas pedal
  2. Apply pressure to gas pedal with right foot
  3. If speed is too high, apply less pressure.
  4. If speed is too low, apply more pressure.
Are Computers Intelligent?

- Do we really need to be involved in programming computers?
  - They have beaten world chess champions.
  - They help predict weather patterns.
  - They can perform arithmetic quickly.
- So, a computer has an IQ of _____.

What Do We Have To Do?

- Computers cannot analyze problems and devise solutions.
- Humans (that’s us) must
  - Analyze and understand a problem;
  - Devise a sequence of steps to solve the problem;
  - Translate the steps into a computer language.
Basic Computer Components

Central Processing Unit (CPU)
- Arithmetic/Logic Unit
- Control Unit
- Memory Unit

Input Devices
Output Devices

Introduction

Central Processing Unit (CPU)
- Executes stored *instructions*
- Arithmetic/Logic Unit (ALU)
  - Performs arithmetic and logical operations
- Control Unit
  - Controls the other components
  - Guarantees instructions are executed in sequence
### Memory Unit

- Ordered sequence of *cells* or *locations*
- Stores instructions and data in *binary*
- Types of memory
  - Read-Only Memory (ROM)
  - Random Access Memory (RAM)

<table>
<thead>
<tr>
<th>Address</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td></td>
</tr>
<tr>
<td>1003</td>
<td></td>
</tr>
<tr>
<td>1004</td>
<td></td>
</tr>
<tr>
<td>1005</td>
<td></td>
</tr>
<tr>
<td>1006</td>
<td></td>
</tr>
<tr>
<td>1007</td>
<td></td>
</tr>
<tr>
<td>1008</td>
<td></td>
</tr>
<tr>
<td>1009</td>
<td></td>
</tr>
</tbody>
</table>

### Input and Output Devices

- Interaction with humans
- Gathers data (Input)
- Displays results (Output)
What is Computer Programming?

- Planning or scheduling a sequence of steps for a computer to follow to perform a task.
- Basically, telling a computer what to do and how to do it.

What Is A Computer Program?

- A sequence of steps to be performed by a computer.
- Expressed in a computer language.
A Computer Program In C++

// This program converts miles to kilometers.
// From Problem Solving, Abstraction, & Design Using C++
// by Frank L. Friedman and Elliot B. Hoffman
#include <iostream>
using namespace std;

int main() {
    const float KM_PER_MILE = 1.609; // 1.609 km in a mile
    float miles, // input: distance in miles
         kms; // output: distance in kilometers
    // Get the distance in miles
    cout << "Enter the distance in miles: ";
    cin >> miles;
    // Convert the distance to kilometers and display it.
    kms = KM_PER_MILE * miles;
    cout << "The distance in kilometers is " << kms << endl;
    return 0;
}

Computer Languages

- A set of
  - Symbols (punctuation),
  - Special words or keywords (vocabulary),
  - And rules (grammar)
used to construct a computer program.
Differences In Computer Languages

- Languages differ in
  - Size (or complexity)
  - Readability
  - Expressivity (or writability)
  - "Level"
    - closeness to instructions for the CPU

Machine Language

- Binary-coded instructions
- Used directly by the CPU
- Lowest level language
- Every program step is ultimately a machine language instruction

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>2034</td>
<td>10010110</td>
</tr>
<tr>
<td>2035</td>
<td>11101010</td>
</tr>
<tr>
<td>2036</td>
<td>00010010</td>
</tr>
<tr>
<td>2037</td>
<td>10101010</td>
</tr>
<tr>
<td>2038</td>
<td>10010110</td>
</tr>
<tr>
<td>2039</td>
<td>11101010</td>
</tr>
<tr>
<td>2040</td>
<td>11111111</td>
</tr>
<tr>
<td>2041</td>
<td>01010101</td>
</tr>
<tr>
<td>2042</td>
<td>10101101</td>
</tr>
</tbody>
</table>
Assembly Language

- Each CPU instruction is labeled with a mnemonic.
- Very-low level language
  - Almost 1 to 1 correspondence with machine language
- Translated to machine language by an assembler.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>10010011</td>
</tr>
</tbody>
</table>

Sample Program

- MUL X, 10
- ADD X, Y
- STO Z, 20
- SUB X, Z

High-Level Languages

- Closer to natural language
- Each step maps to several machine language instructions
- Provides support for abstractions
  - Easier to state and solve problems
  - Example: Colors
Examples of High-Level Languages

<table>
<thead>
<tr>
<th>Language</th>
<th>Primary Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pascal</td>
<td>Learning to program</td>
</tr>
<tr>
<td>C++</td>
<td>General purpose</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>Scientific programming</td>
</tr>
<tr>
<td>PERL</td>
<td>Web programming, text processing</td>
</tr>
<tr>
<td>Java</td>
<td>Web programming, application</td>
</tr>
<tr>
<td></td>
<td>programming</td>
</tr>
<tr>
<td>COBOL</td>
<td>Business</td>
</tr>
</tbody>
</table>

Basic Programming Language Structures

- **Sequence**
  - Execute steps or *statements* in the language one after another.
**Basic Programming Language Structures**

- **Selection (AKA *branch* or *decision*)**
  - Selectively execute statements based on some condition being true or false.
  - `IF condition THEN statement1 ELSE statement2`

- **Loop (AKA *repetition* or *iteration*)**
  - Repeat a statement several times until a specified condition is false.
  - `WHILE condition DO statement1`
Basic Programming Language Structures

- Subprogram (AKA procedure, function, method, or subroutine)
  - A collection of the previous structures that accomplishes some smaller task.

SUBPROGRAM1

SUBPROGRAM1
Accomplishes some small part of a larger task.