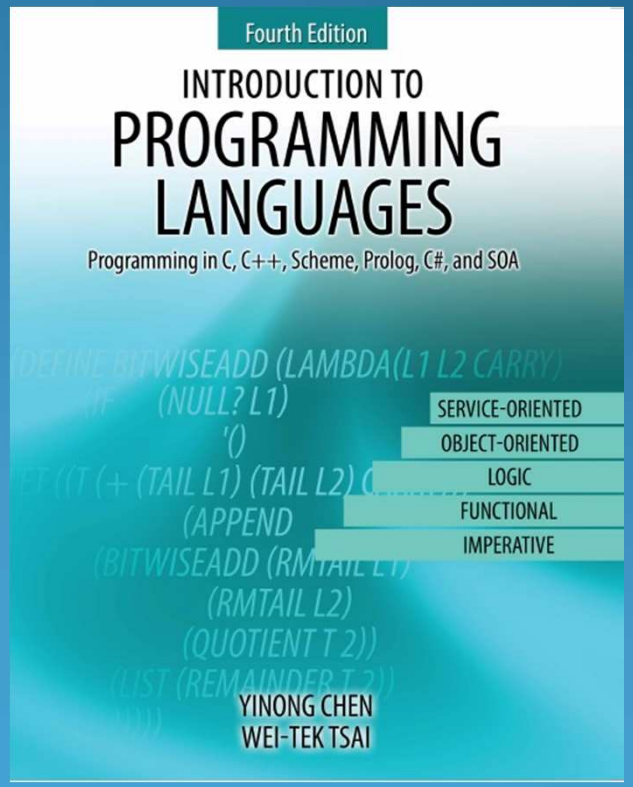
Introduction to Engineering Using Robotics Experiments

Programming Languages



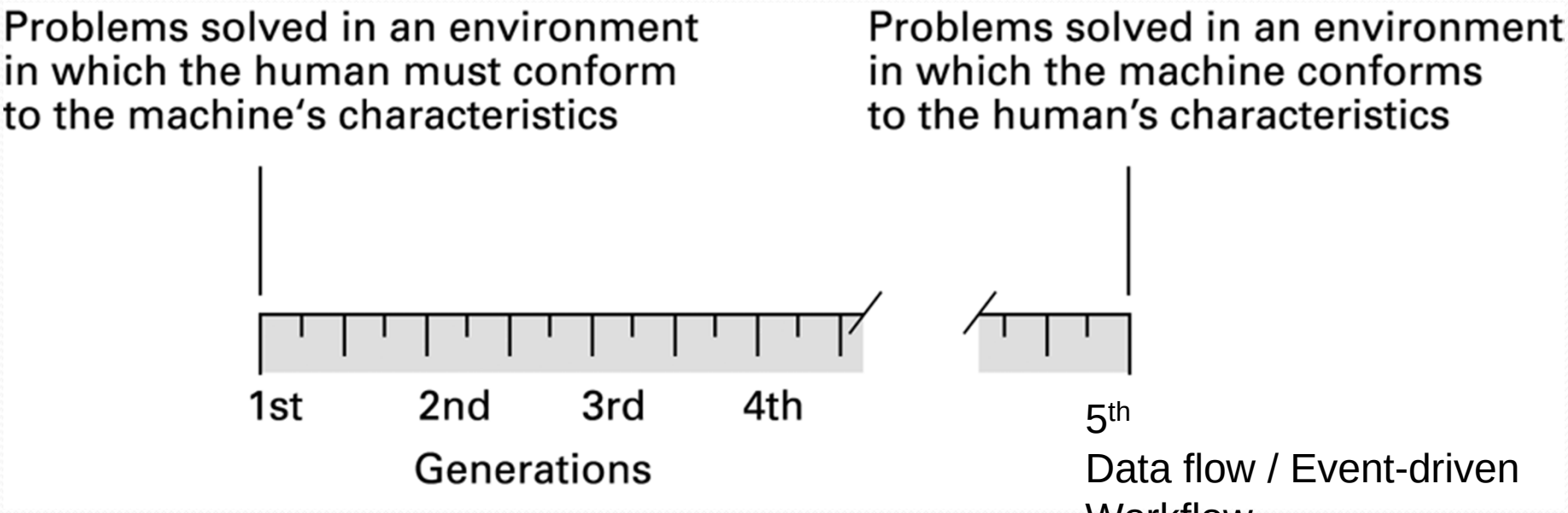
Dr. Yinong Chen

IRA A. FULTON SCHOOLS OF engineering
school of computing, informatics, & decision systems engineering

Outline

- Historical Perspective
- Programming Language Generations
- Programming Language Paradigms
- Imperative Programming Paradigm
- Writing **Imperative Programs** in C#
- Console Interface vs. Graphic User Interface
- Next…
  - Object-Oriented Computing
  - Service-Oriented Computing
  - Web-based Computing
  - Mobile Computing

Generations of Programming Languages Classification by Time



5th

Data flow / Event-driven

Workflow

Visual Programming,

- Alice
- VPL
- Phone Inventor

First-Generation: Machine language

- Operator and operands are coded in to binary
- Each instruction is a binary (or hexadecimal) number

Second-Generation: Assembly language

- A mnemonic system for representing machine instructions
- One-to-one correspondence (table) between machine instructions and assembly instructions
- Converted to machine language by a program called an **assembler**: look up the table

Program Examples

Machine language in HEX		Assembly language
156C	⟷	LD R5, Price
166D	⟷	LD R6, ShippingCharge
5056	⟷	ADDI R0, R5 R6
30CE	⟷	ST R0, TotalCost
C000	⟷	HLT

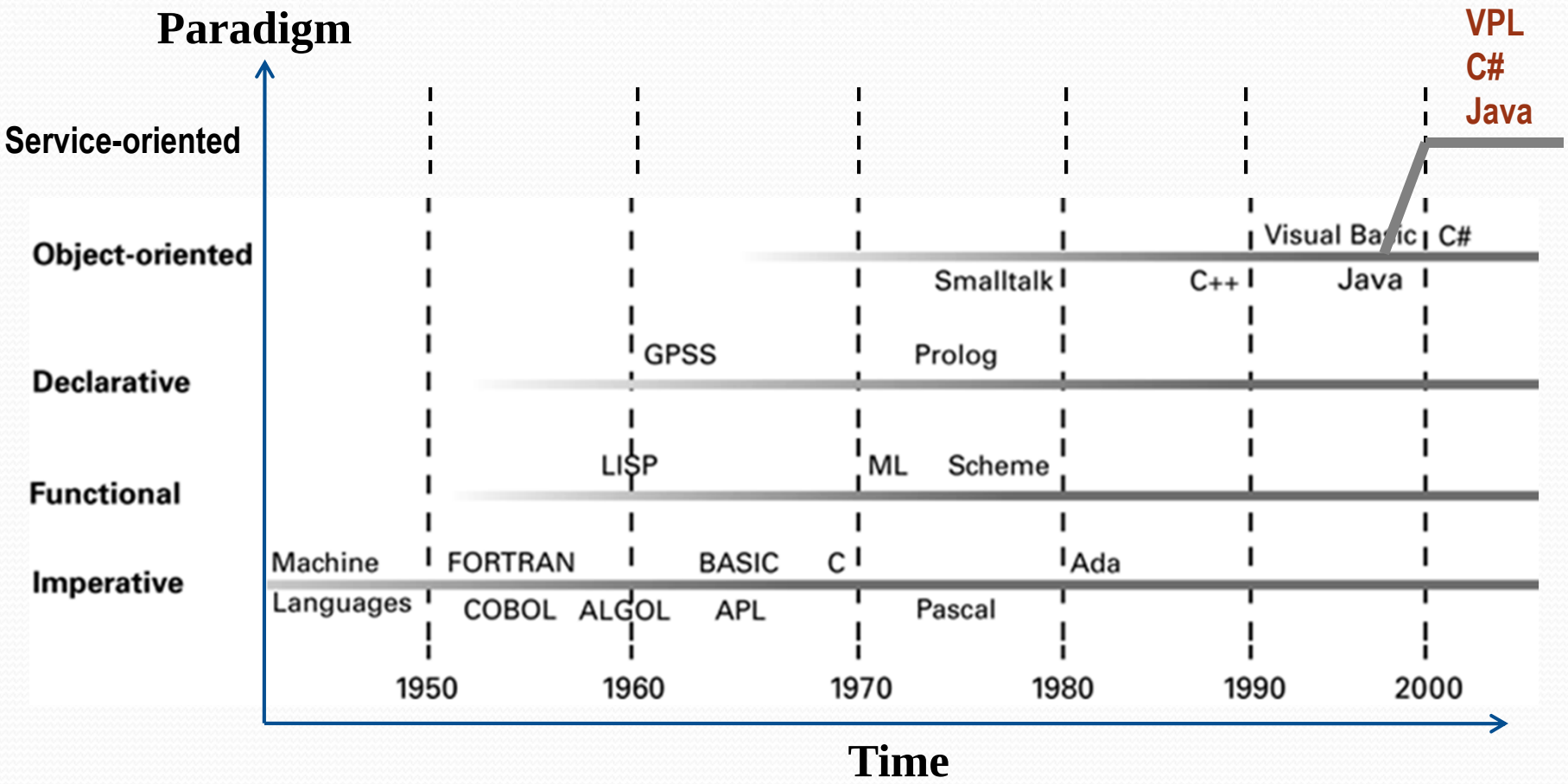
Machine code

Assembly code

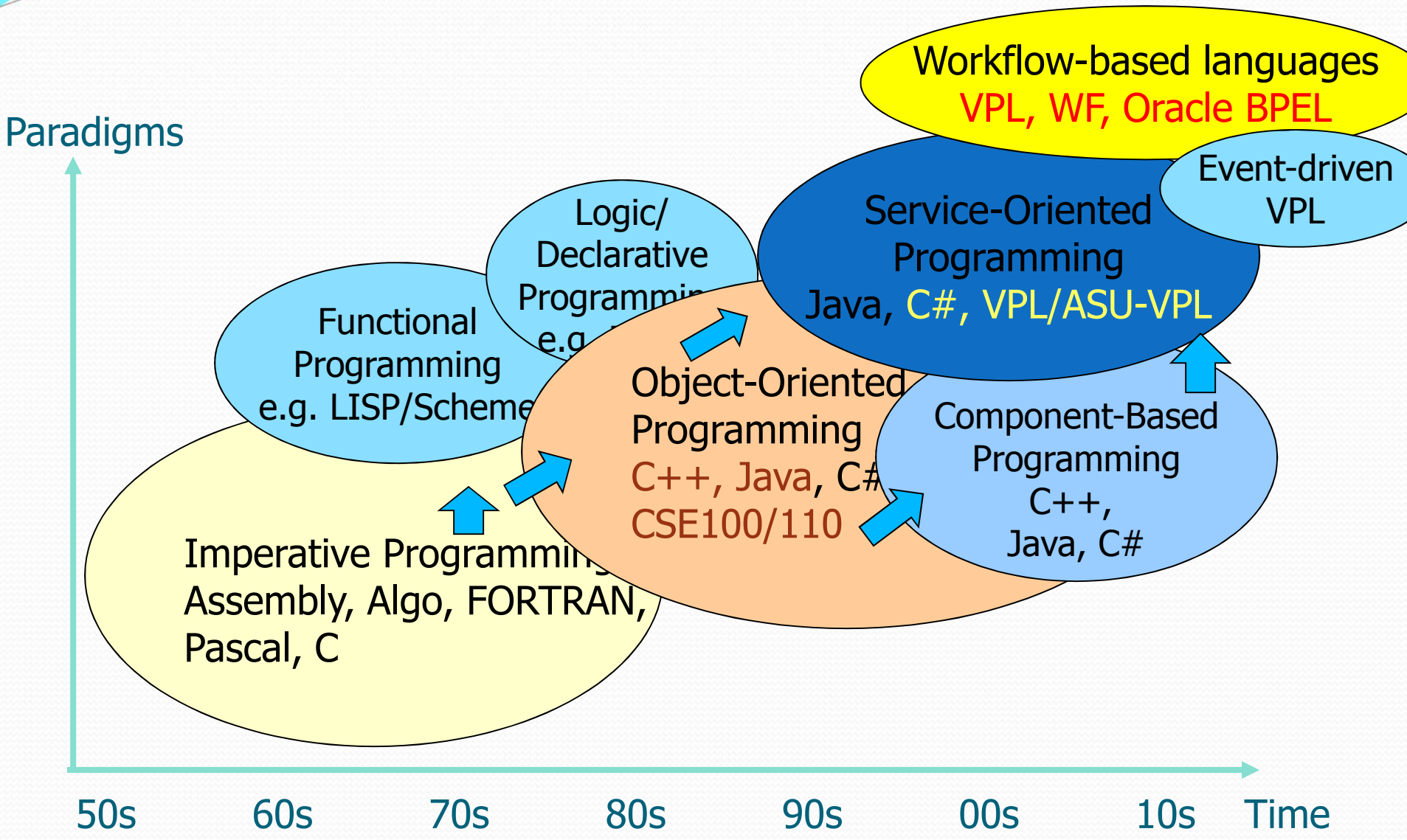
Third Generation Language

- Uses high-level imperative languages
  - Similar to our pseudo-code
- Machine independent (mostly)
- Examples: FORTRAN, COBOL, C
- Each primitive corresponds to a sequence of machine language instructions
- Converted to machine language by a program called a **compiler**

Evolution of Programming Languages



Spectrum of Programming Languages



What is a Program?

Program

Algorithm

Process (steps) of data manipulation

Emphasis: Imperative / Procedural Paradigms

Data structure

Objects of the manipulation

Emphasis: Object-Oriented Paradigm

Imperative Programming Paradigm

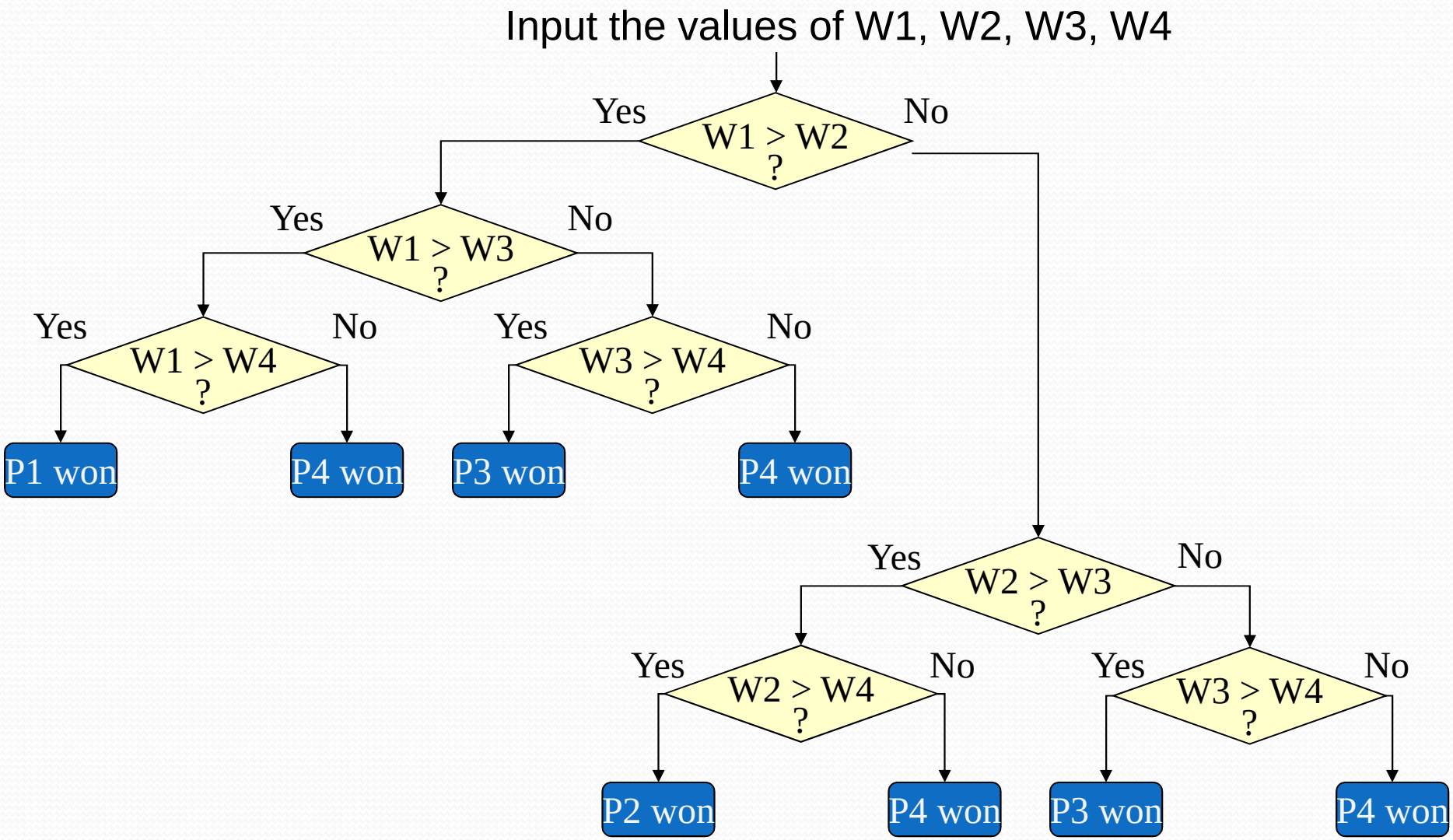
Fully specified and fully controlled manipulation of named data in a step-wise fashion.

- Developed as abstractions of von Neumann machine (stored program concept).
- Programs are algorithmic in nature: do this, then that, then repeat this ten times -- focuses on how rather than what.

Why popular?

- Performance – match the machine
- Culture – reading manuals
- Foundation of object-oriented programming

Flowchart


Input the values of W1, W2, W3, W4
W1 > W2 ?
Yes
No
W1 > W3 ?
Yes
No
W1 > W4 ?
Yes
No
W3 > W4 ?
Yes
No
P1 won
P4 won
P3 won
P4 won
W2 > W3 ?
Yes
No
W2 > W4 ?
Yes
No
W3 > W4 ?
Yes
No
P2 won
P4 won
P3 won
P4 won

Basics of Programming

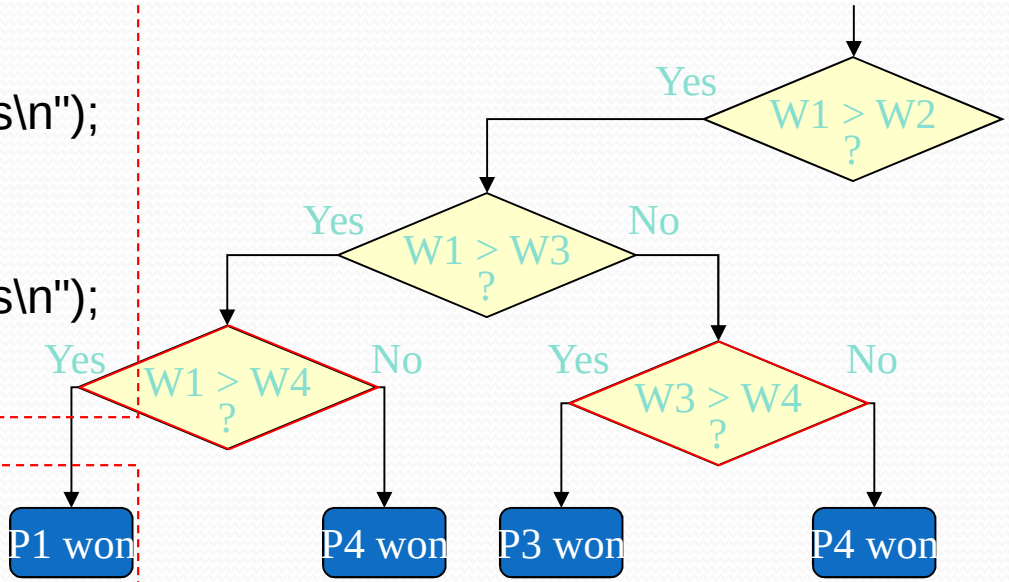
1. List the library packages to be used, e.g., I/O package;
2. Declare variables
3. Initialize variables
   - From outside (keyboard, sensors, networks);
   - Hardcoded assignment, x = “Hello World”; y = 7;
4. Manipulate variables (computing)
   - One time modification, e.g., x = x + 1; z = x + y;
   - Multiple modifications using a loop;
5. Selections
   - Select one out of two: if-then-else;
   - Select one out of multiples: switch;
6. Loops
   - For-loop: Iterate a fixed number of times
   - While-loop: Iterate until a condition is met

Implementation: Map the Problem to Program

```
using System;              // It includes the most frequently used lib functions
class weightLift {         // main class
    static void Main() {
   // Declare variables (memory spaces)
     Int32 W1, W2, W3, W4;    // for holding the weight lifted by each player
     string str;              // for temporarily holding the input from keyboard;

     // Enter the weights lifted by each player
     Console.WriteLine("Please enter the weight lifted by Player 1\n");
     str = Console.ReadLine();                    // read a string of characters
     W1 = Convert.ToInt32(str);                   // Convert the string to an integer

     Console.WriteLine("Please enter the weight lifted by Player 2\n");
     str = Console.ReadLine();                    // read a string of characters
     W2 = Convert.ToInt32(str);                   // Convert the string to an integer

     Console.WriteLine("Please enter the weight lifted by Player 3\n");
     str = Console.ReadLine();                    // read a string of characters
     W3 = Convert.ToInt32(str);                   // Convert the string to an integer

     Console.WriteLine("Please enter the weight lifted by Player 4\n");
     str = Console.ReadLine();                    // read a string of characters
     W4 = Convert.ToInt32(str);                   // Convert the string to an integer
```

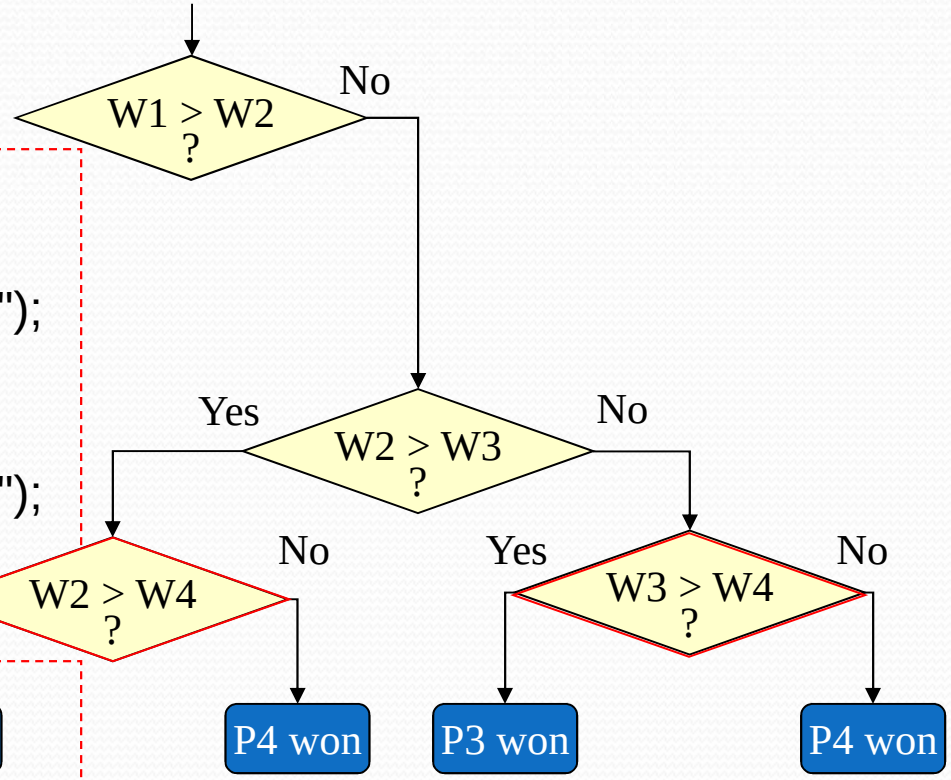
Map the Problem to Program (contd.)

```
if (W1 > W2)
    {
        if (W1 > W3)
        {
            if (W1 > W4)
            {
                Console.Write("Player 1 Wins\n");
            }
            else
            {
                Console.Write("Player 4 Wins\n");
            }
        }
        else
        {
            if (W3 > W4)
            {
                Console.Write("Player 3 Wins\n");
            }
            else
            {
                Console.Write("Player 4 Wins\n");
            }
        }
    }
```



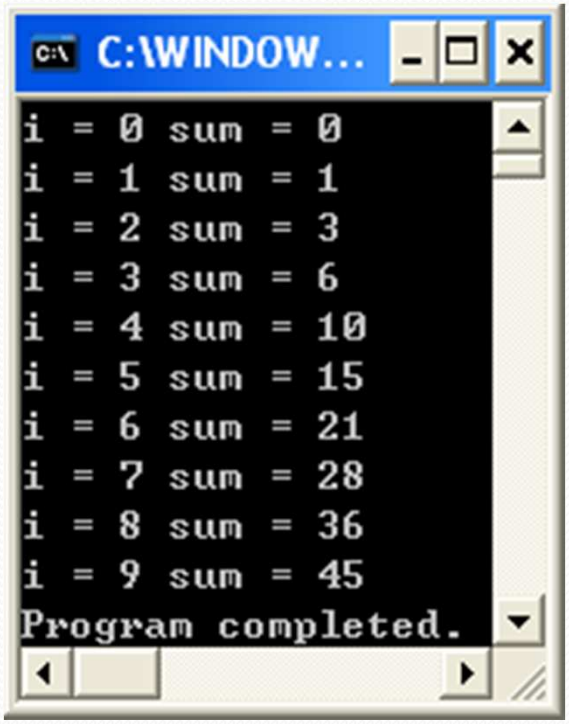
Map the Problem to Program (contd.)

```
else
    {
        if (W2 > W3)
        {
            if (W2 > W4)
            {
                Console.Write("Player 2 Wins\n");
            }
            else
            {
                Console.Write("Player 4 Wins\n");
            }
        }
        else
        {
            if (W3 > W4)
            {
                Console.Write("Player 3 Wins\n");
            }
            else
            {
                Console.Write("Player 4 Wins\n");
            }
        }
    }
}
}
```



Add and Display a List of Numbers in a While-Loop

```
using System;
class addList
{
   static void Main()
   {
      Int32 i = 0;
      Int32 sum = 0;
      while (i<10)
      {
         sum = sum + i;
         Console.WriteLine("i = {0} sum = {1} ", i, sum);
         i++;
      }
      Console.WriteLine("Program completed.");
   }
}
```



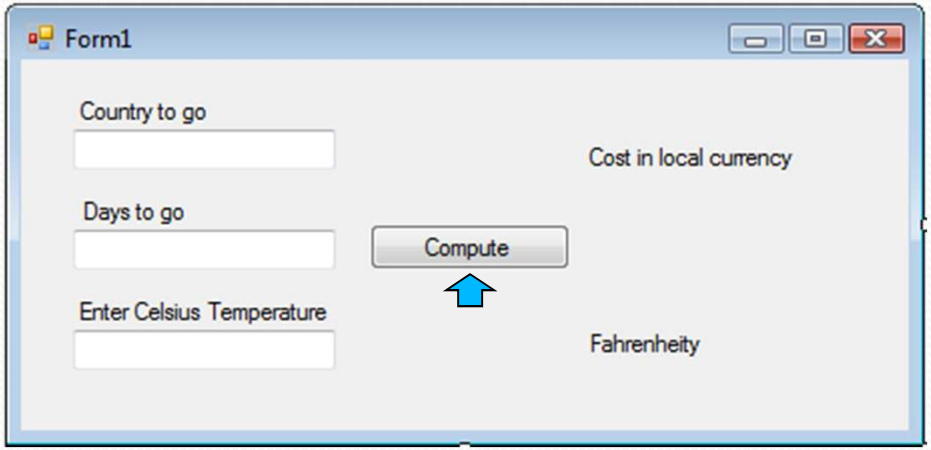
Interface to Users

- Console input and output
  - Minimum effort on the interface design
  - Used by developers in the development stage
- Windows-based GUI (Graphic User's Interface)
  - The application is running on the operating system of the computer
  - Example: Install a game and play on your computer
- Web-based GUI
  - The application is running on a remote server;
  - User access the application through a Web browser;
  - Example: Play an internet game
- Mobile Device GUI
  - Windows Phone, iPhone, Android Phone

Console Interface versus Graphic User Interface

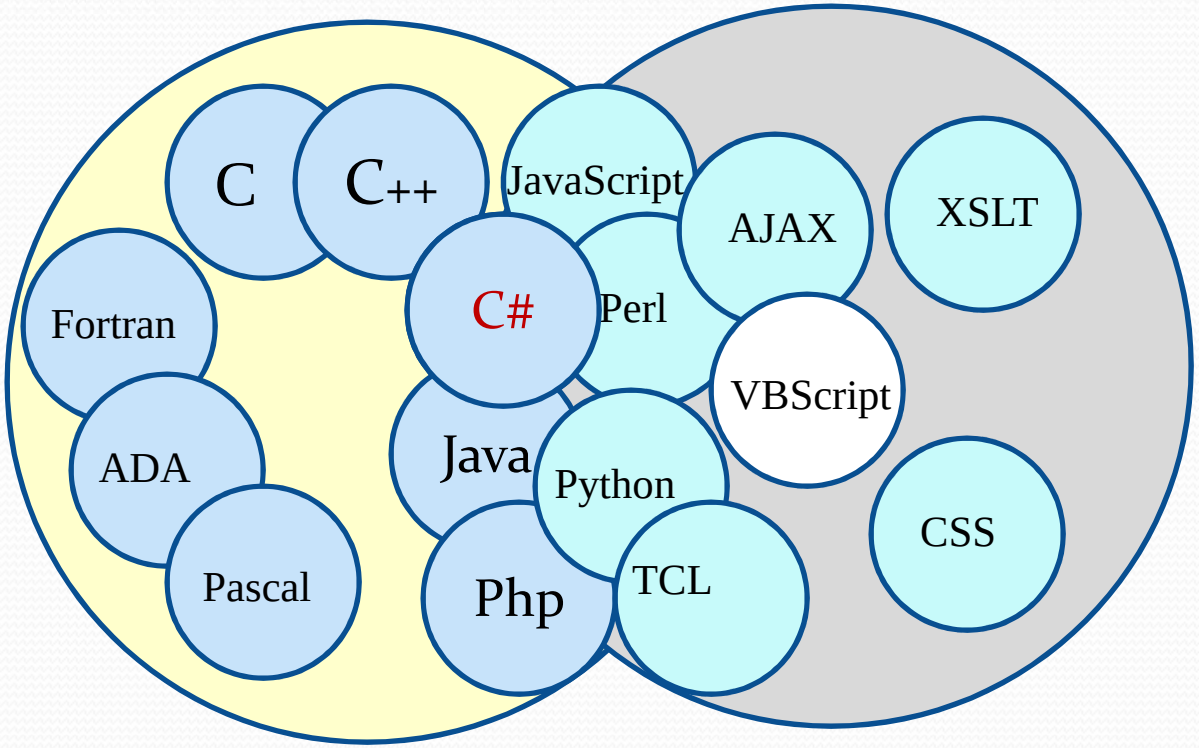
C:\Windows\system32\cmd.exe

```
Please enter the number of days you will travel
7
Please enter the country name you will travel to
UK
The amount of local currency needed is: 514.5
Please enter the temperature in Celsius
5
The local temperature in Farenheit is: 41
Press any key to continue . . .
```



Traditional Programming Languages vs. Scripting Languages

- They come and go
  - Fortran
  - Cobol
  - Pascal
  - ADA
  - Lisp
  - Scheme
  - Prolog
  - Java
  - C
  - C++
  - Php
  - Ruby on Rails
  - SQL
  - Java
  - C#
- But basic concepts remain



Traditional programming languages

Scripting languages

Scripting Languages

- Job control languages and shells:
  - IBM JCL
  - Unix Script
  - AppleScript
- Visual programming languages/Workflow:
  - Alice
  - National Instrument LabView
  - EV3 Programming Language
  - Microsoft VPL / ASU - VPL
- Application-specific languages:
  - QuickC
  - Emacs Lisp
  - Parallax C for robotics programming
- Extension/embeddable languages
  - SpiderMonkey embedded in Yahoo Widget Engine
  - Adobe Flash (ActionScript)
  - TCL
  - Perl
  - Python
- Web client-side scripting:
  - AJAX
  - CSS, XSLT
  - JavaScript
  - VBScript, C#
  - ECMAScript
- Dynamic languages and server-side Scripting & Computing:
  - Java
  - PHP
  - C# on ASP .Net
  - Ruby on Rails