Computers and Programming

Principles of Computer Programming I

Spring/Fall 20XX

AUGUSTA
UNIVERSITY

Outline

- Principles of Computer Programming
- Computer language concepts
- Software Concepts
- Programming workflow
- Visual Studio concepts

Computer Programming

- Computers: frequently changing hardware

- Rapid increases in capability (storage, speed, graphics, etc.)

Computer Programming

- Programming: Many different languages for instructing computers what to do, different tools for different jobs

OS and drivers

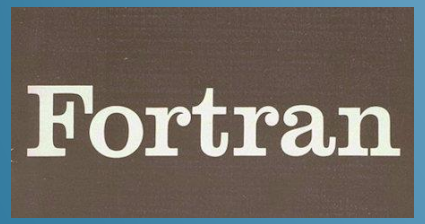
Graphics, video games, server software

Desktop and web apps

Scripting

New server software

Interactive web pages



Scientific computation

Principles of Computer Programming

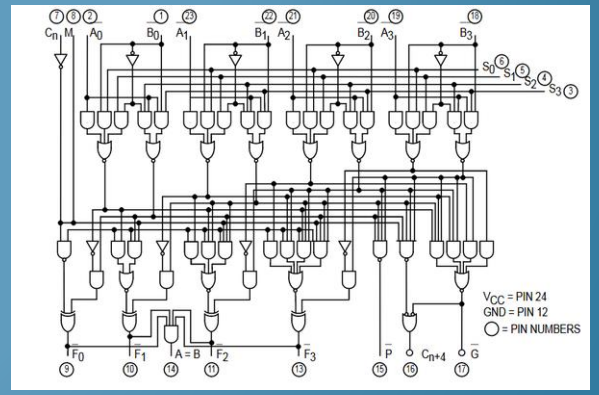
- Learn common principles behind all languages
  - How to organize and structure data
  - How to express logical conditions and relations
  - How to solve problems with programs
- Won't change even as languages and hardware change
- Mathematical theory: All programming languages are equivalent

Outline

- Principles of Computer Programming
- **Computer language concepts**
- Software concepts
- Programming workflow
- Visual Studio concepts

From Hardware to Software

- Machine language: Binary digits (1 and 0) representing on/off state of wires
- Encodes basic instructions for computer (CPU): read, write, add, subtract, etc.
- Each CPU could have its own, but most follow a standard: either x86 or ARM



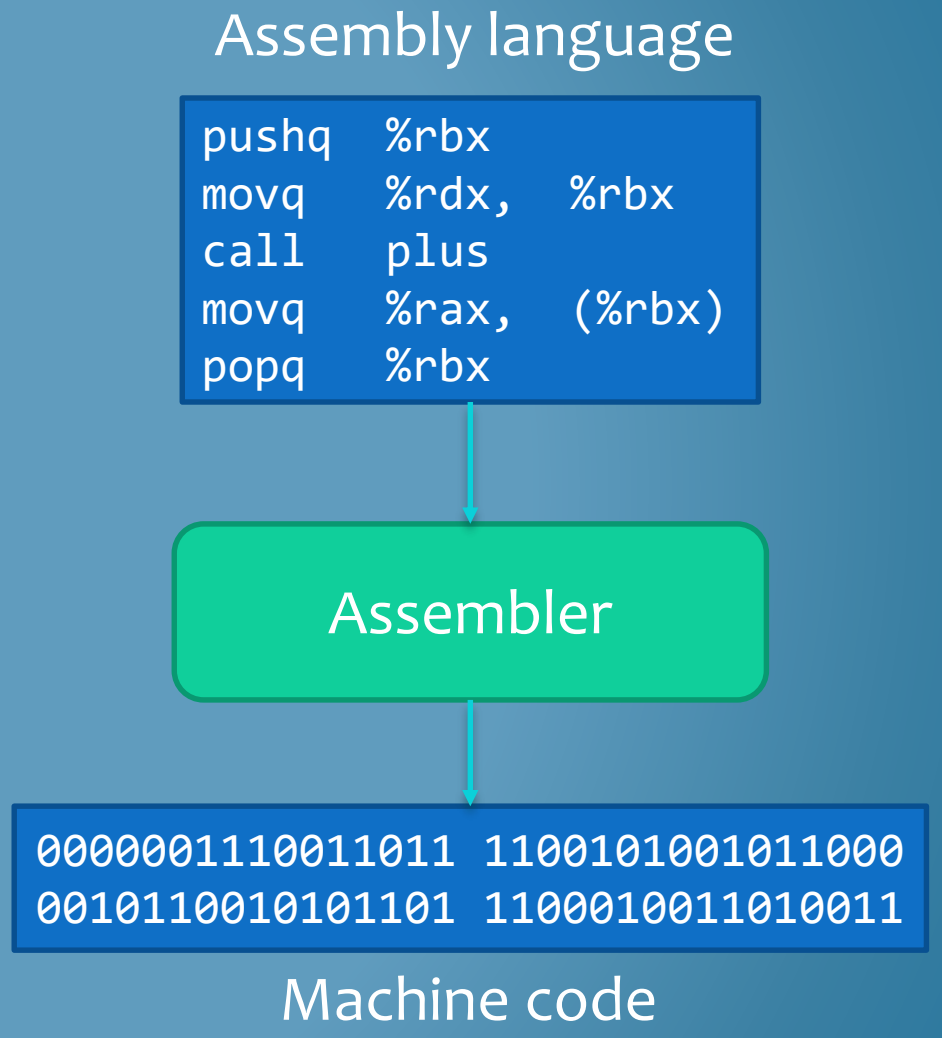

```
0000001100111011
```

set these wires to OFF

set these wires to ON

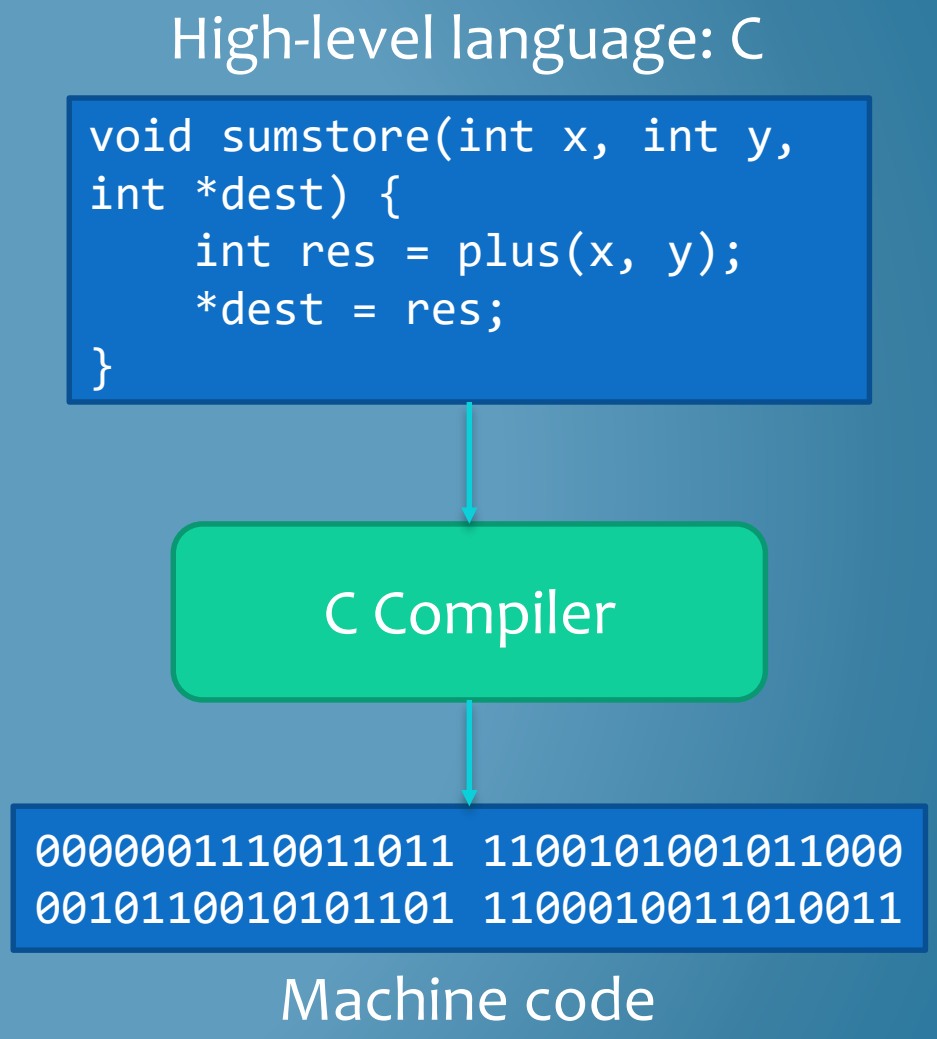
Assembly Language

- "Human-readable" (kind of) representation of machine instructions – letters and symbols instead of bits
- **Assembler:** translates assembly language to machine code
- Each statement produces one machine instruction
  - x86 assembly → x86 machine code



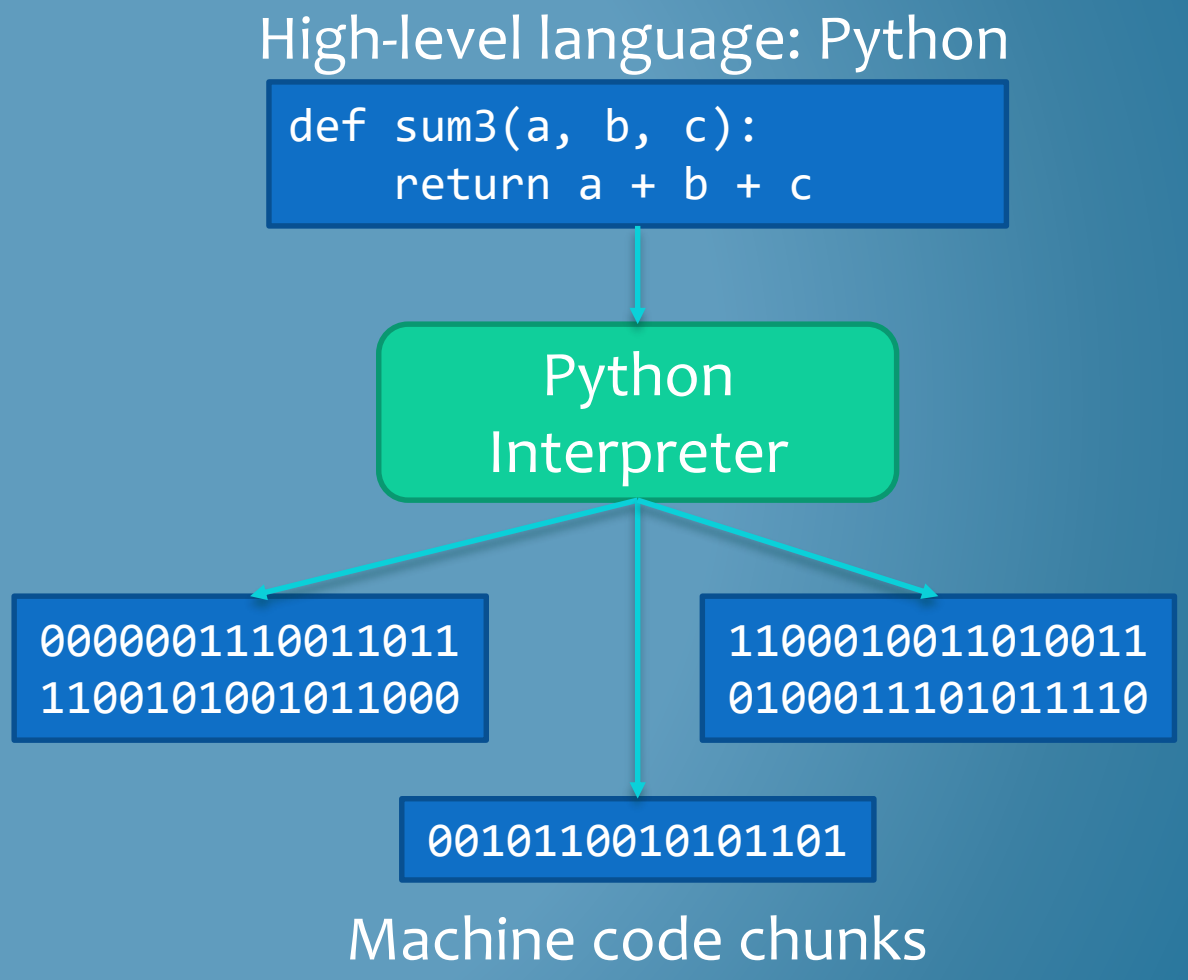
High-Level Language

- Much more human-readable
- Statements don't correspond directly to machine instructions
- **Compiler:** Translates high-level language to machine code
  - Each high-level language and target machine needs its own compiler
  - Incorporates code libraries
- After compiling, **run** the machine code



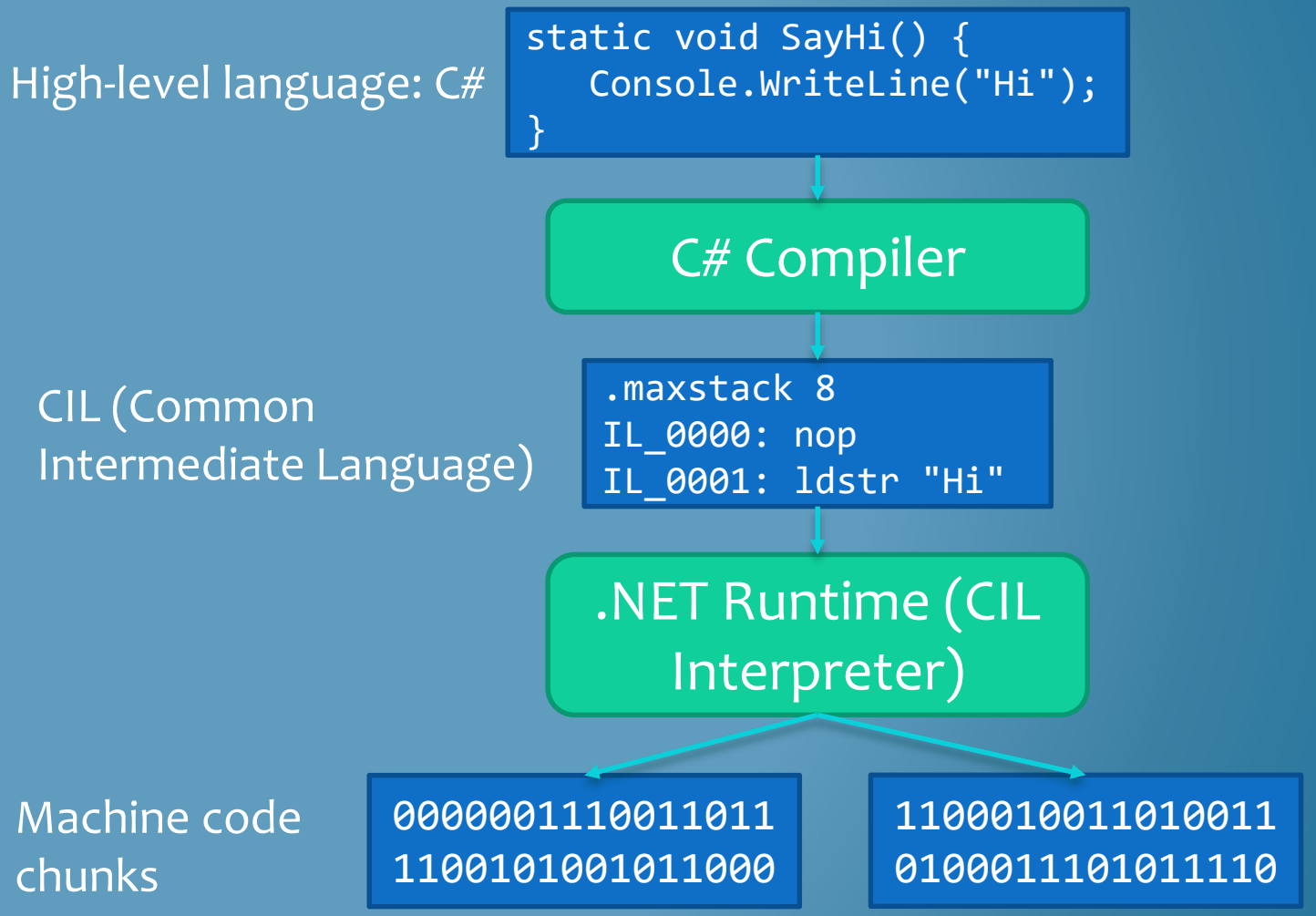
Compiled vs. Interpreted

- **Interpreter:** Directly executes high-level code by translating one instruction at a time
- Less waiting before you can run your program
- Runs slower than compiled machine code



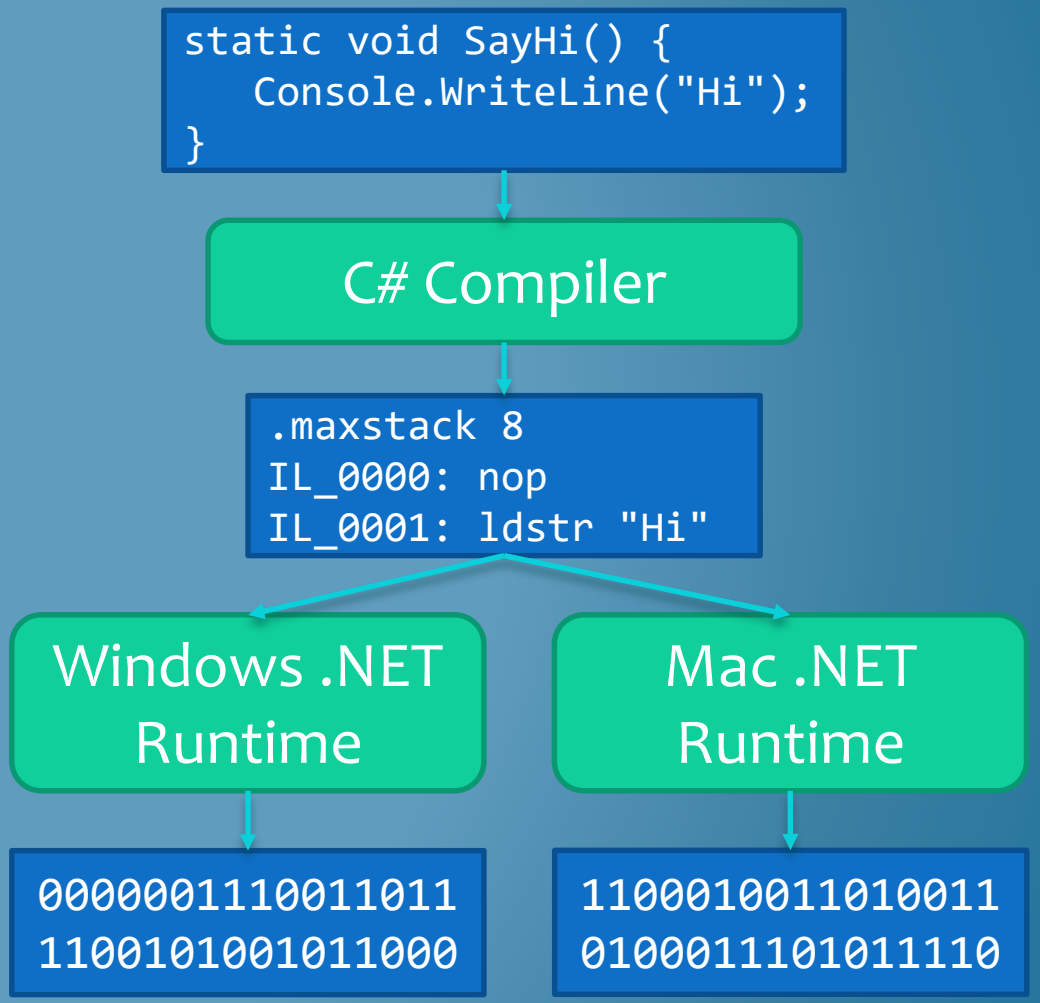
Managed High-Level Languages

- Some features of both compiled & interpreted
- Compiler produces an **intermediate language**
  - Like assembly, but not tied to any CPU
- **Runtime:** an interpreter for the intermediate language



Why Managed Languages?

- **Platform:** Combination of OS (Windows, Mac, Linux) and hardware
- Compiled code only works on one platform
  - Must write a new compiler for each platform and re-compile all code
- Interpreter can generate code for the current platform *without* recompiling
- Interpreting IL much faster than interpreting high-level language

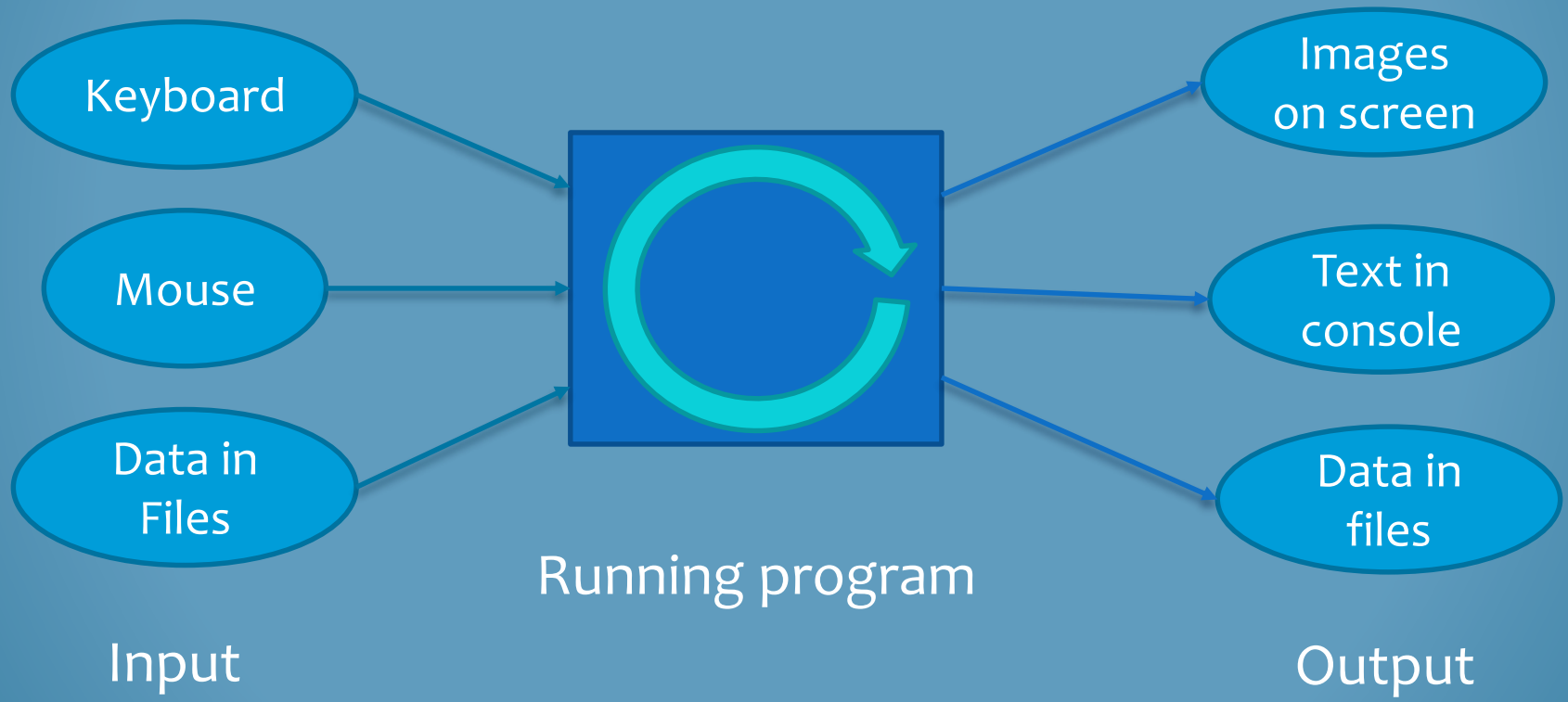


Outline

- Principles of Computer Programming
- Computer language concepts
- **Software concepts**
- Programming workflow
- Visual Studio concepts

Software Concepts

- Programs receive **input** and produce **output**

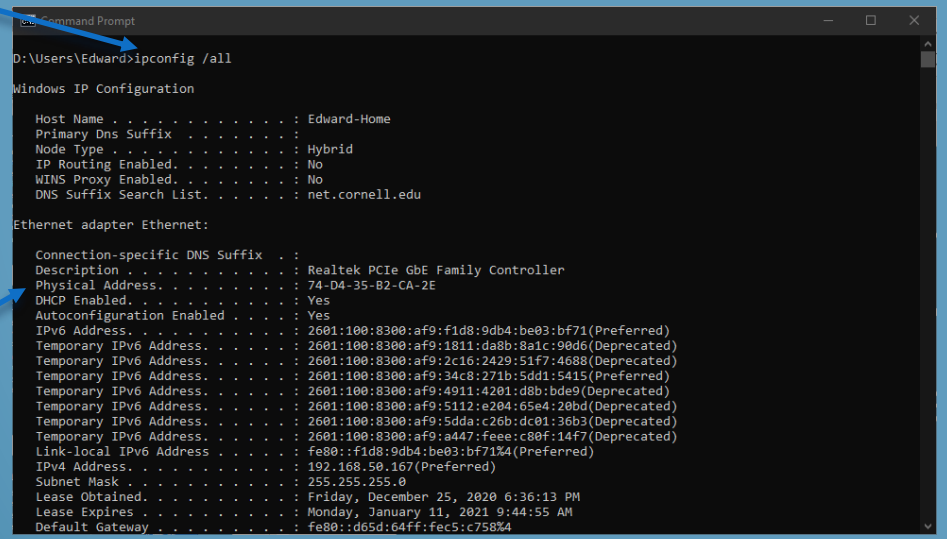


Software Concepts

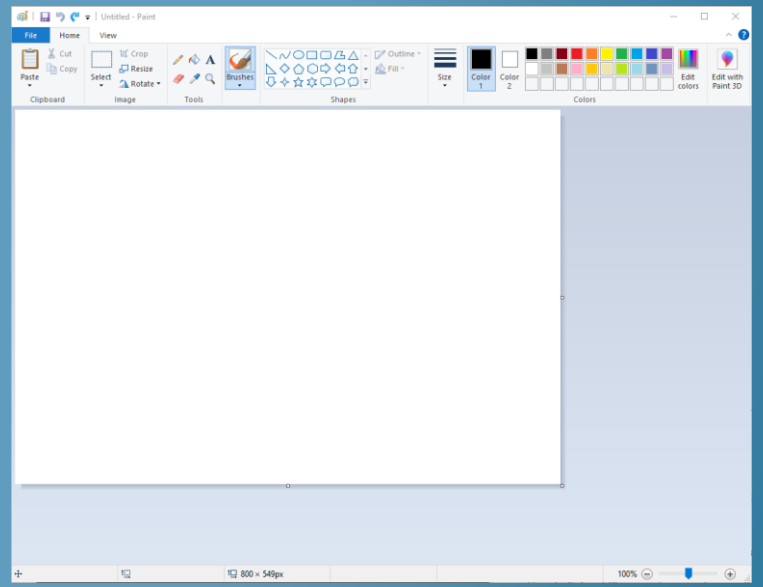
- Two ways of interacting with a program: Command-Line Interface (CLI) and Graphical User Interface (GUI)

Type input at command prompt

Output printed to console



CLI



GUI

Why CLI?

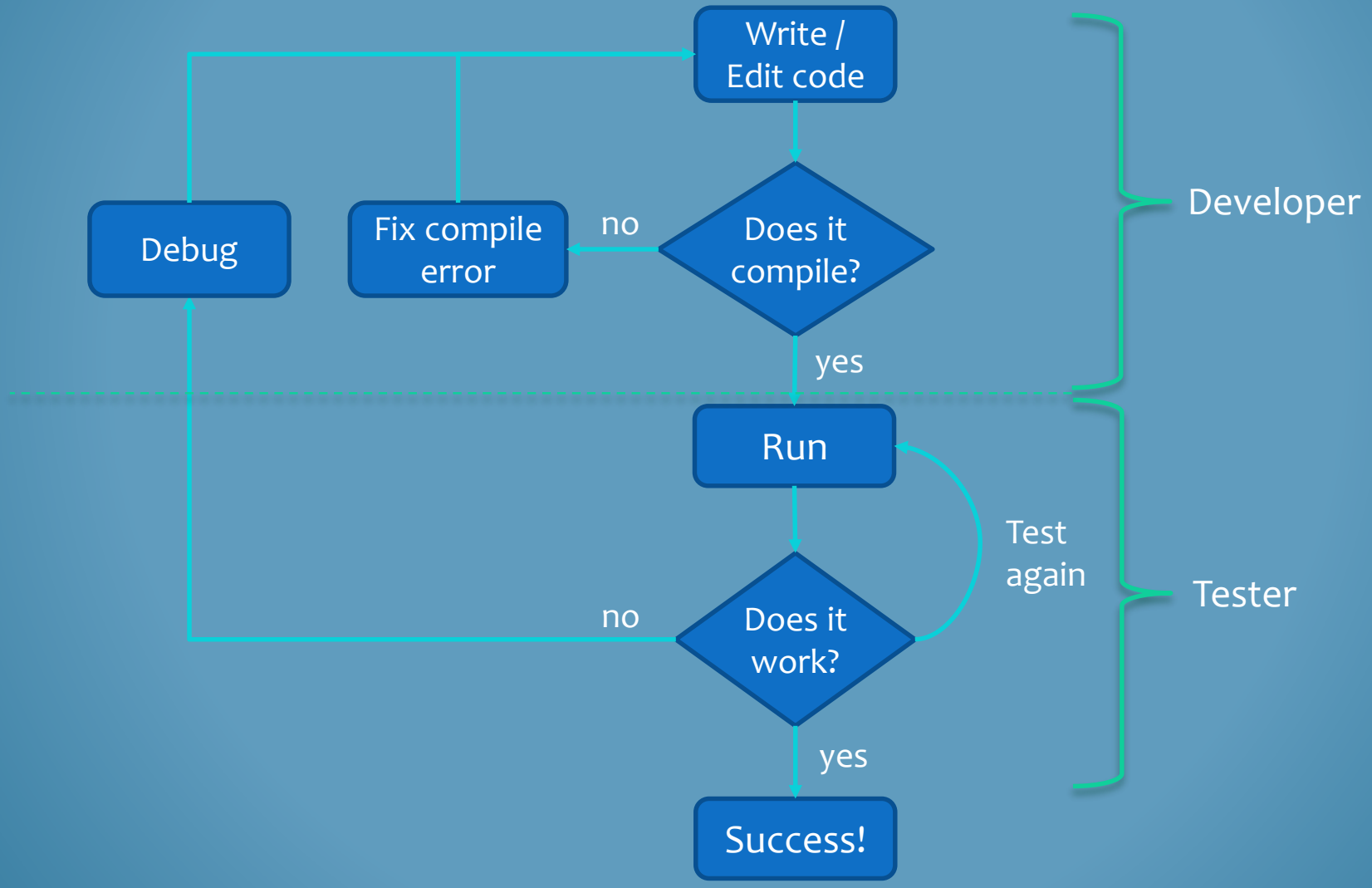
- Same interface as those old “text-only” computers
- Very portable: All computers support it
- GUI depends on OS to help draw “windows” on screen

- Simple and easy to work with, lets you focus on your program’s logic instead of managing input/output with graphics

Outline

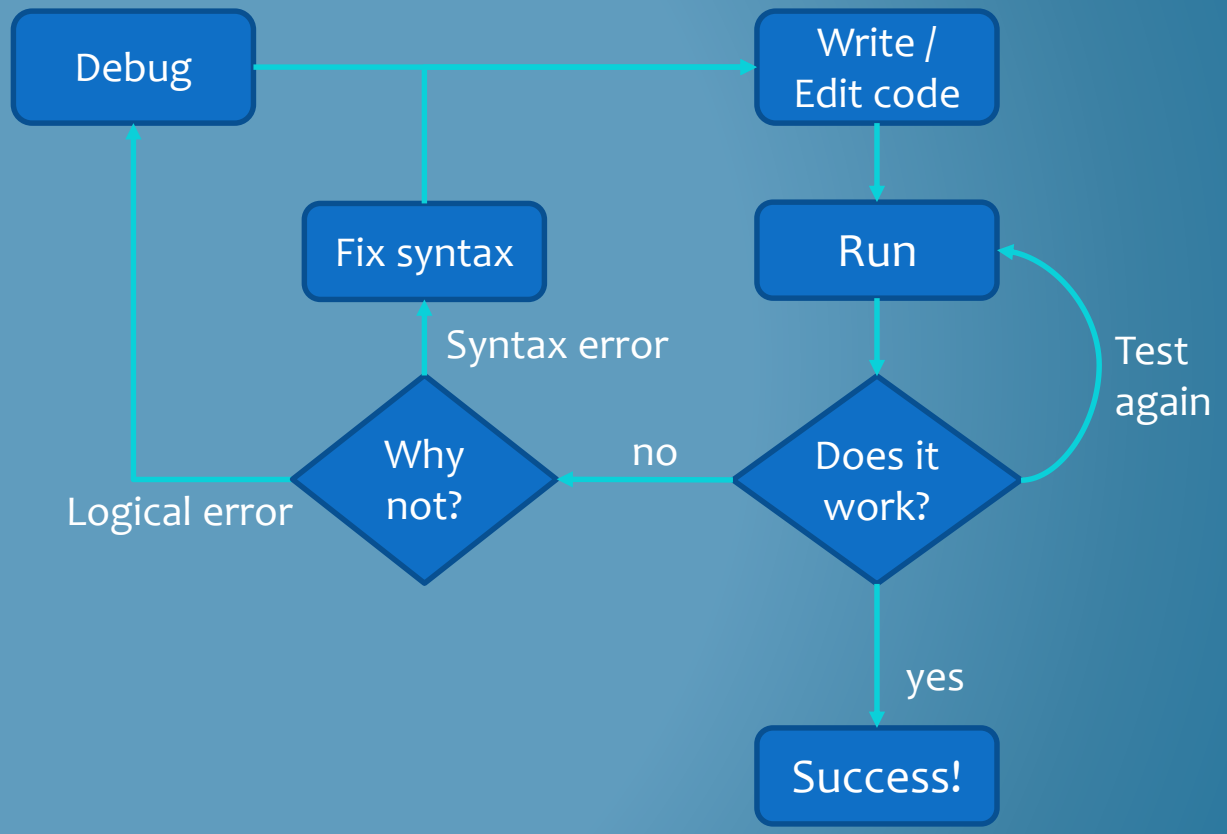
- Principles of Computer Programming
- Computer language concepts
- Software concepts
- **Programming workflow**
- Visual Studio concepts

Programming Workflow



Interpreted Language Workflow

- For comparison: workflow for a language like Python
- All errors happen when you run the program, even syntax mistakes that would be "compile errors"
- Why might you prefer a compiled language?

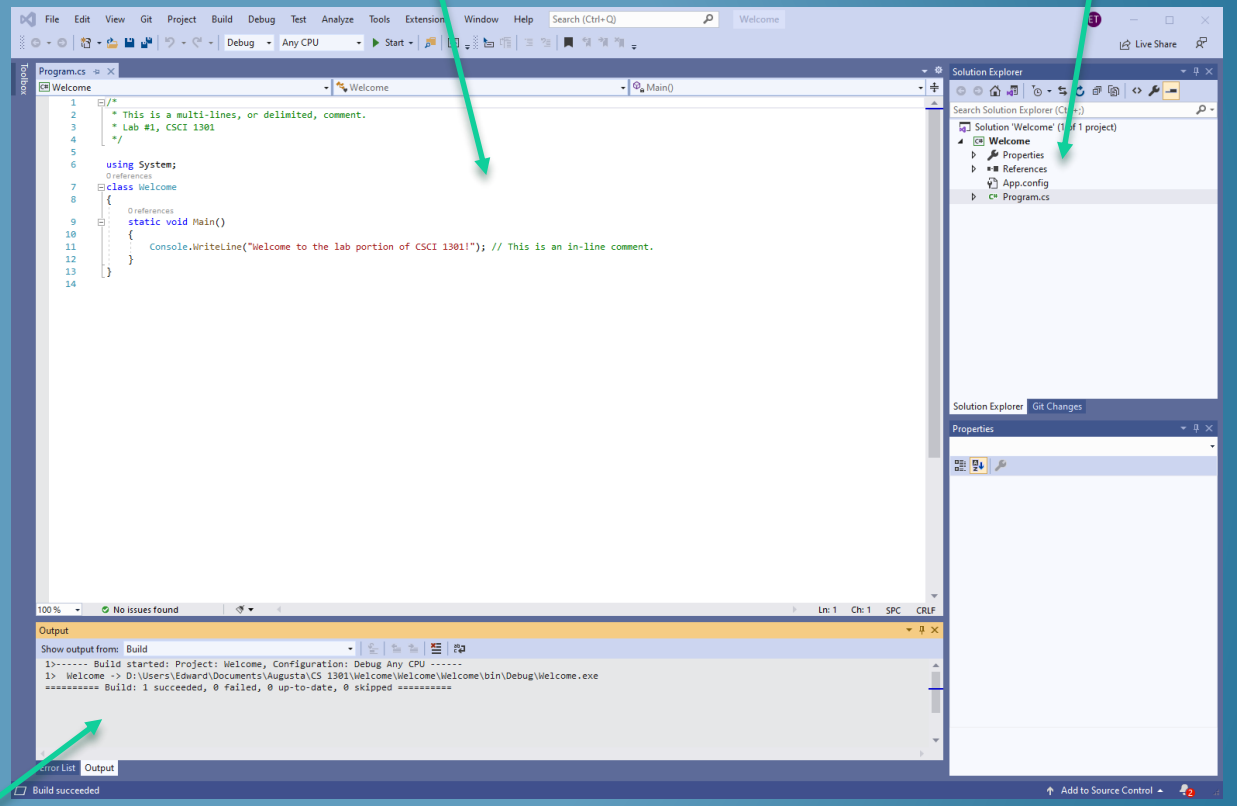


Outline

- Principles of Computer Programming
- Computer language concepts
- Software concepts
- Programming workflow
- **Visual Studio concepts**

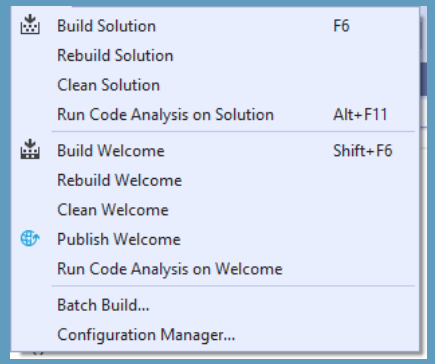
Visual Studio: An IDE

- **I**ntegrated **D**evelopment **E**nvironment: A text editor, compiler, debugger, file browser, other tools
- Can write, compile, run code without leaving Visual Studio
- Tools to help you navigate and edit code



Terminology Quirks

- **Solution:** A software project in Visual Studio, including source code, metadata, data files, images, etc.
- "Build solution" = compile code



- "Start without debugging" = run application