

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



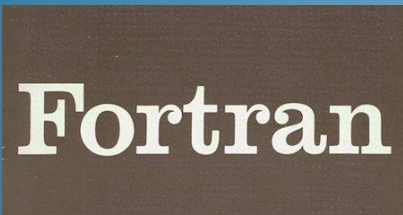
Scripting



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

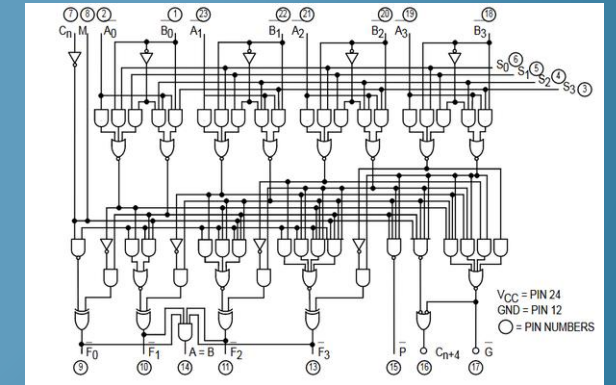
- Machine language: Binary digits (1 and 0) representing on/off state of wires
- Encodes basic instructions for computer (CPU): read, write, add, subtract, etc.
- Each CPU could have its own, but most follow a standard: either x86 or ARM



000000110011011

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

Assembly language

```
pushq %rbx
movq  %rdx, %rbx
call  plus
movq  %rax, (%rbx)
popq  %rbx
```

Assembler

```
0000001110011011 1100101001011000
0010110010101101 1100010011010011
```

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

High-level language: C

```
void sumstore(int x, int y,  
int *dest) {  
    int res = plus(x, y);  
    *dest = res;  
}
```

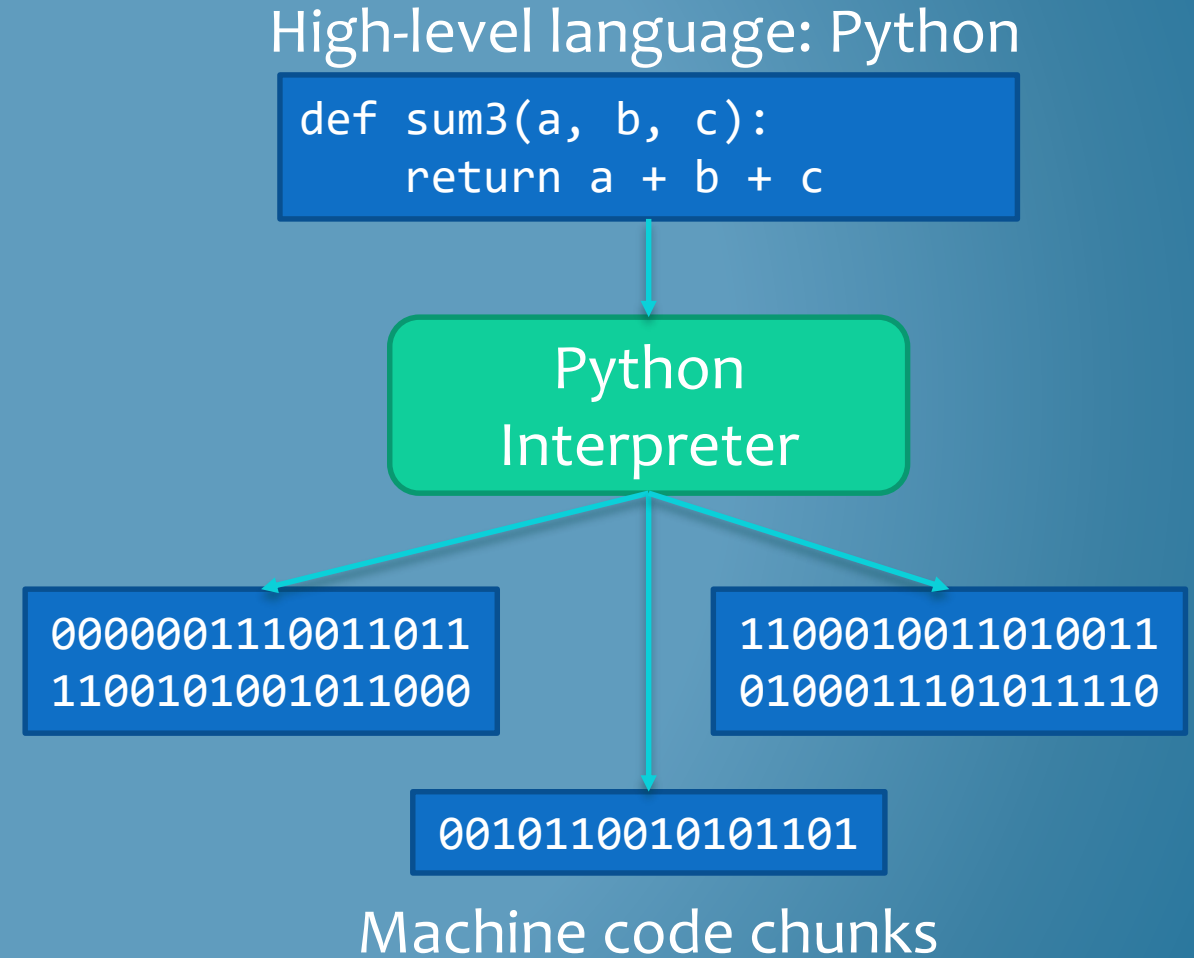
C Compiler

```
0000001110011011 1100101001011000  
0010110010101101 1100010011010011
```

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

High-level language: C#

```
static void SayHi() {  
    Console.WriteLine("Hi");  
}
```

C# Compiler

CIL (Common
Intermediate Language)

```
.maxstack 8  
IL_0000: nop  
IL_0001: ldstr "Hi"
```

.NET Runtime (CIL
Interpreter)

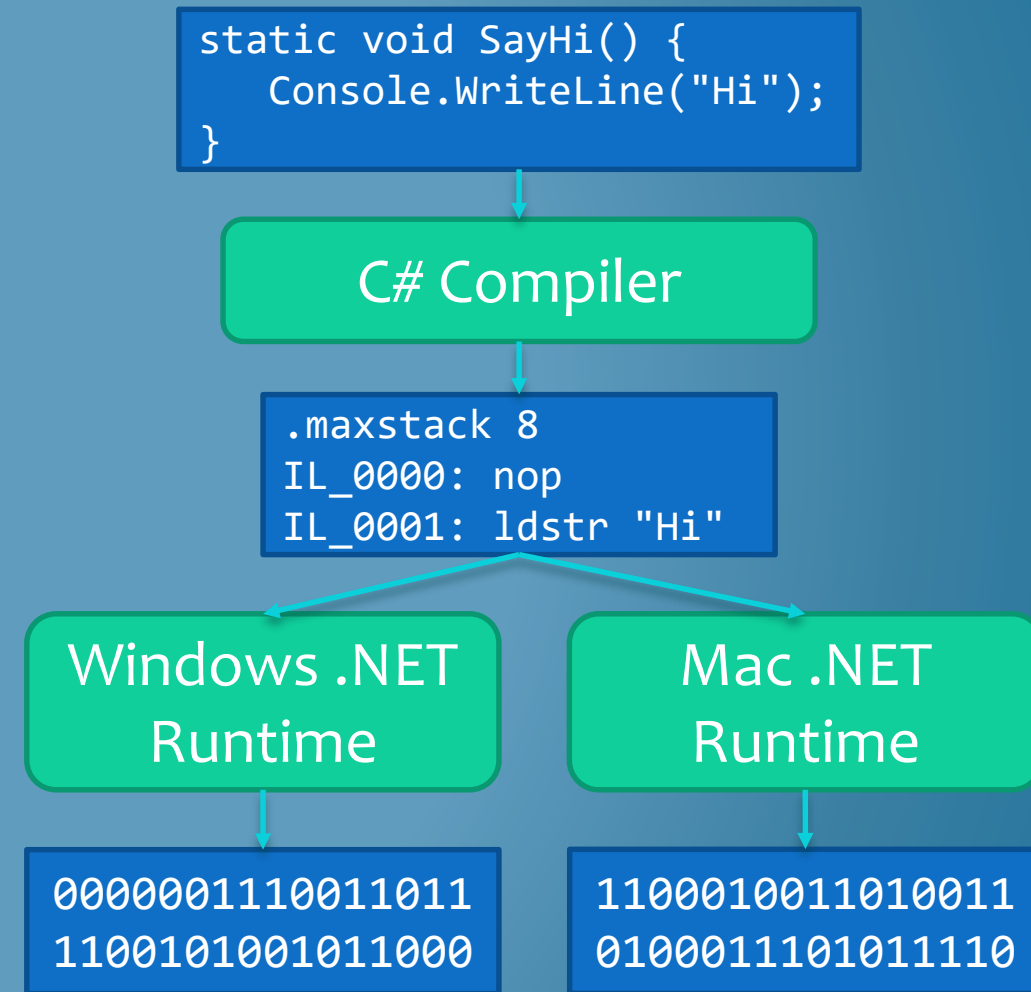
Machine code
chunks

```
0000001110011011  
1100101001011000
```

```
1100010011010011  
0100011101011110
```

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

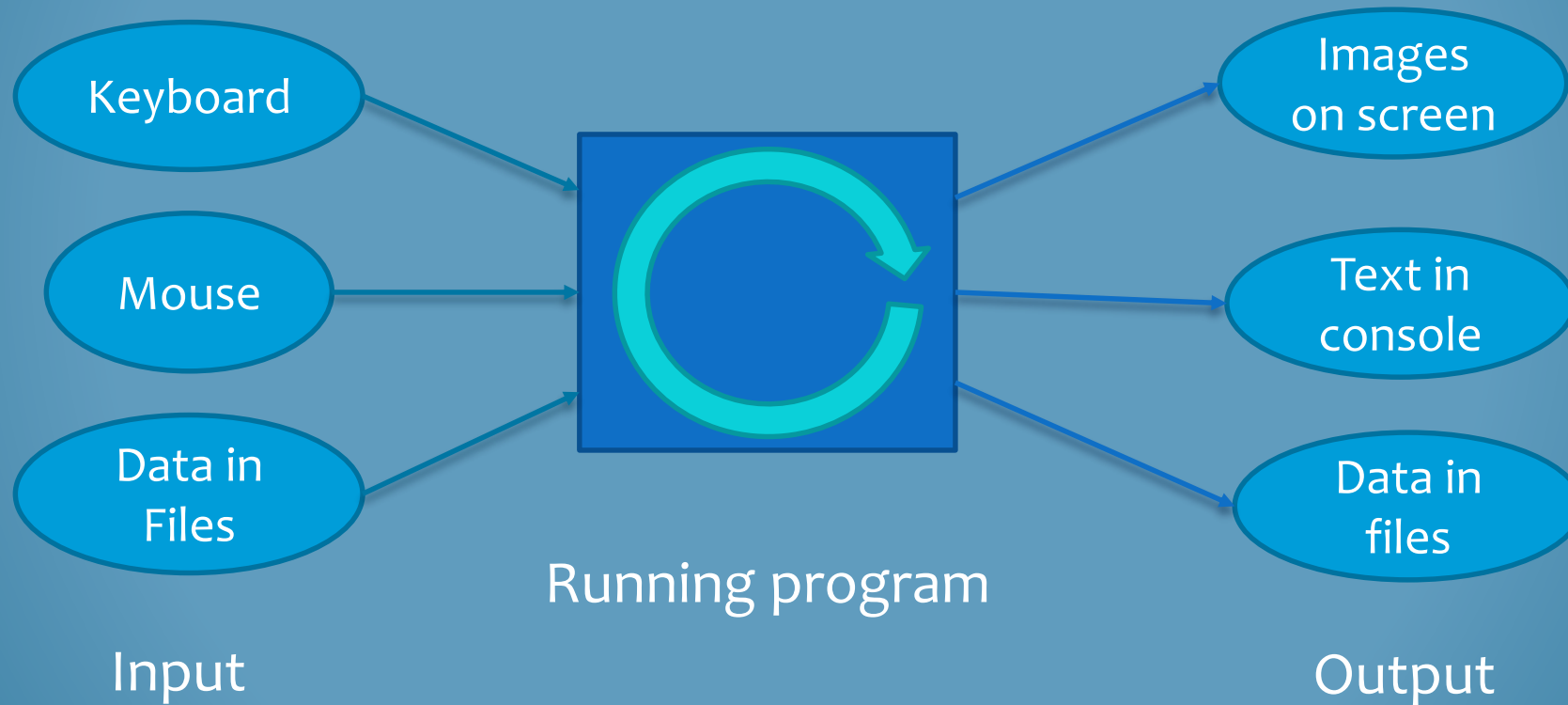


Outline

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

Software Concepts

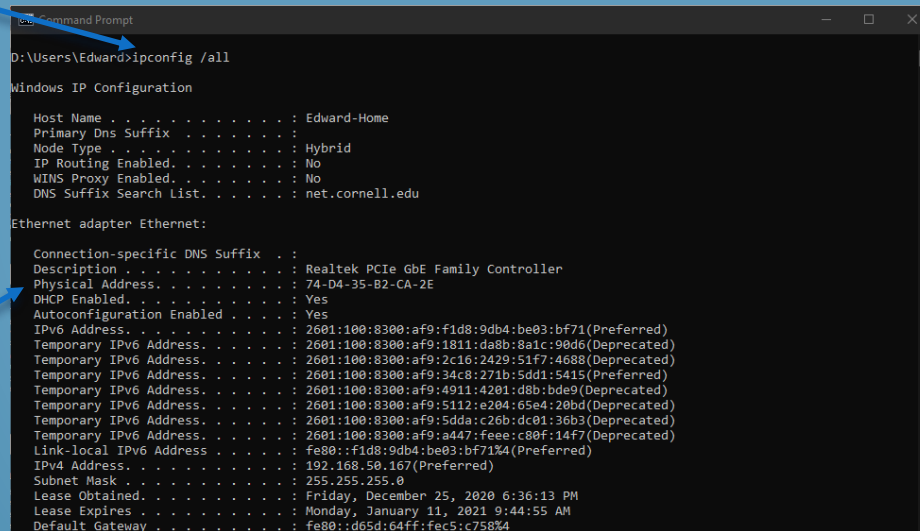
- Programs receive **input** and produce **output**



Software Concepts

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

Type input at
command
prompt



```
Command Prompt
D:\Users\Edward>ipconfig /all

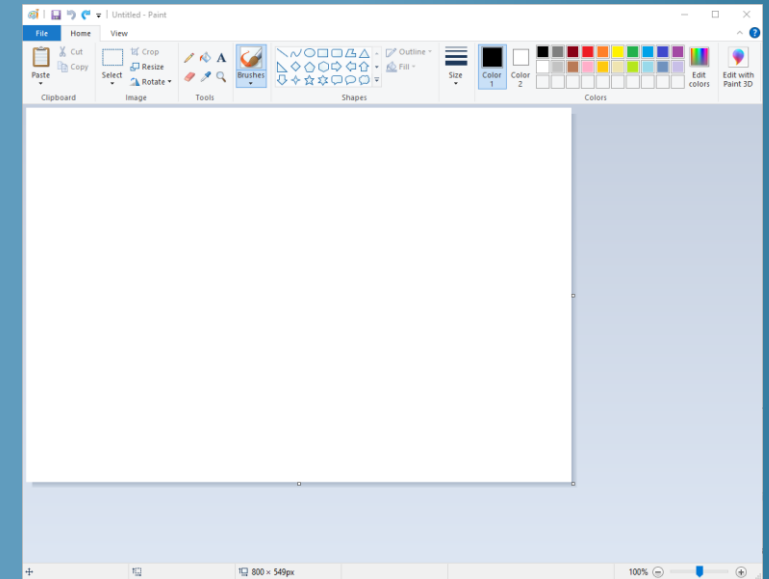
Windows IP Configuration

Host Name . . . . . : Edward-Home
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : net.cornell.edu

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . . :
Description . . . . . : Realtek PCIe GbE Family Controller
Physical Address. . . . . : 74-D4-35-B2-CA-2E
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
IPv6 Address. . . . . : 2601:100:8300:af9:f1d8:9db4:be03:bf71(Preferred)
Temporary IPv6 Address. . . . . : 2601:100:8300:af9:1811:da8b:8a1c:90d6(Deprecated)
Temporary IPv6 Address. . . . . : 2601:100:8300:af9:2c16:2429:51f7:4688(Deprecated)
Temporary IPv6 Address. . . . . : 2601:100:8300:af9:34c8:271b:5dd1:5415(Prefere
Temporary IPv6 Address. . . . . : 2601:100:8300:af9:4911:4201:d8b:bde9(Deprecated)
Temporary IPv6 Address. . . . . : 2601:100:8300:af9:5112:e204:65e4:20bd(Deprecated)
Temporary IPv6 Address. . . . . : 2601:100:8300:af9:5dda:c26b:dc01:36b3(Deprecated)
Temporary IPv6 Address. . . . . : 2601:100:8300:af9:a447:feee:c80f:14f7(Deprecated)
Link-local IPv6 Address . . . . . : fe80::f1d8:9db4:be03:bf71%4(Prefere
IPv4 Address. . . . . : 192.168.50.167(Prefere
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Friday, December 25, 2020 6:36:13 PM
Lease Expires . . . . . : Monday, January 11, 2021 9:44:55 AM
Default Gateway . . . . . : fe80::d65d:64ff:fec5:c758%4
```

CLI



GUI

Why CLI?

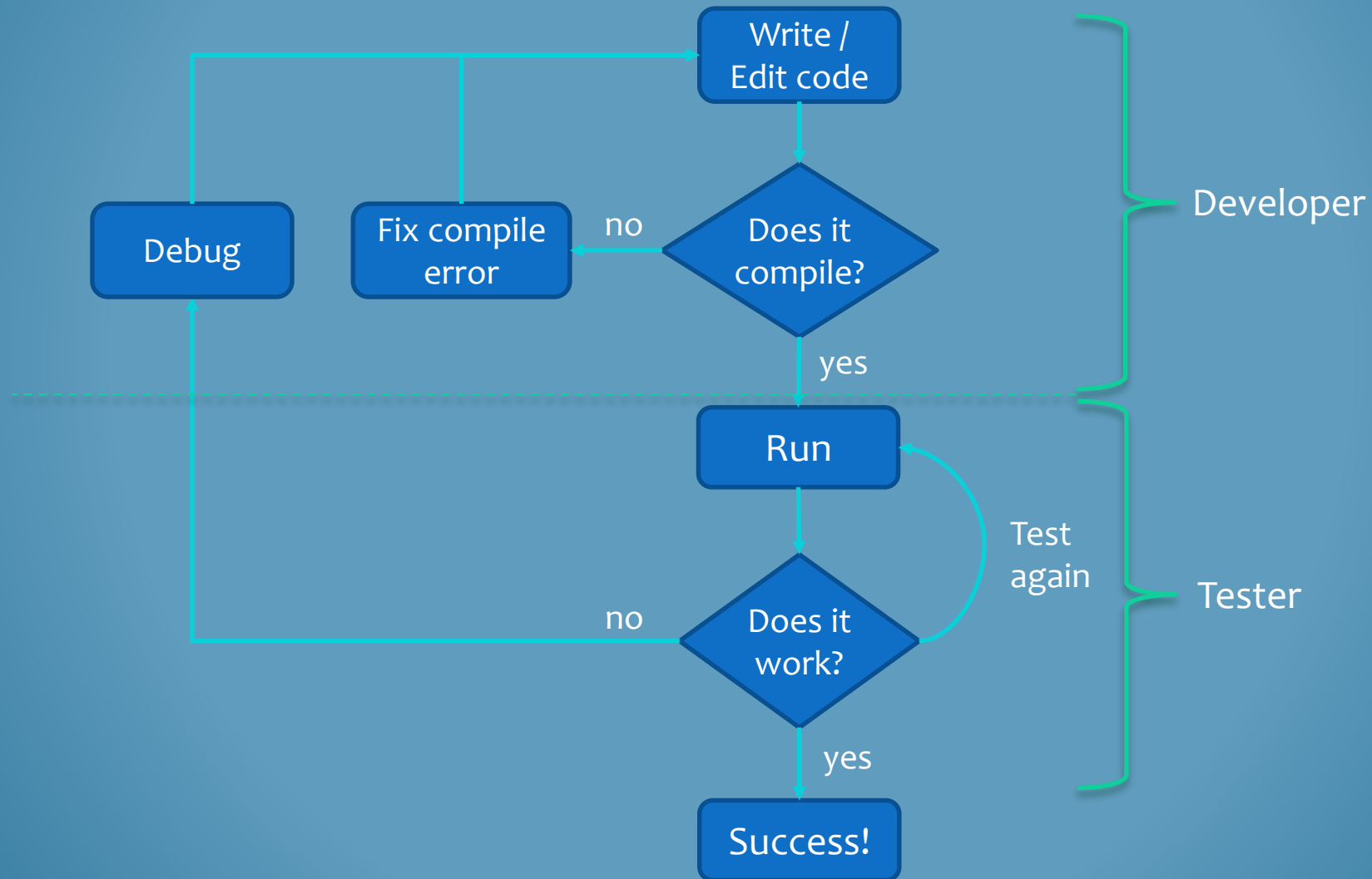
- Same interface as those old “text-only” computers
- Very portable: All computers support it
- GUI depends on OS to help draw “windows” on screen
- Simple and easy to work with, lets you focus on your program’s logic instead of managing input/output with graphics



Outline

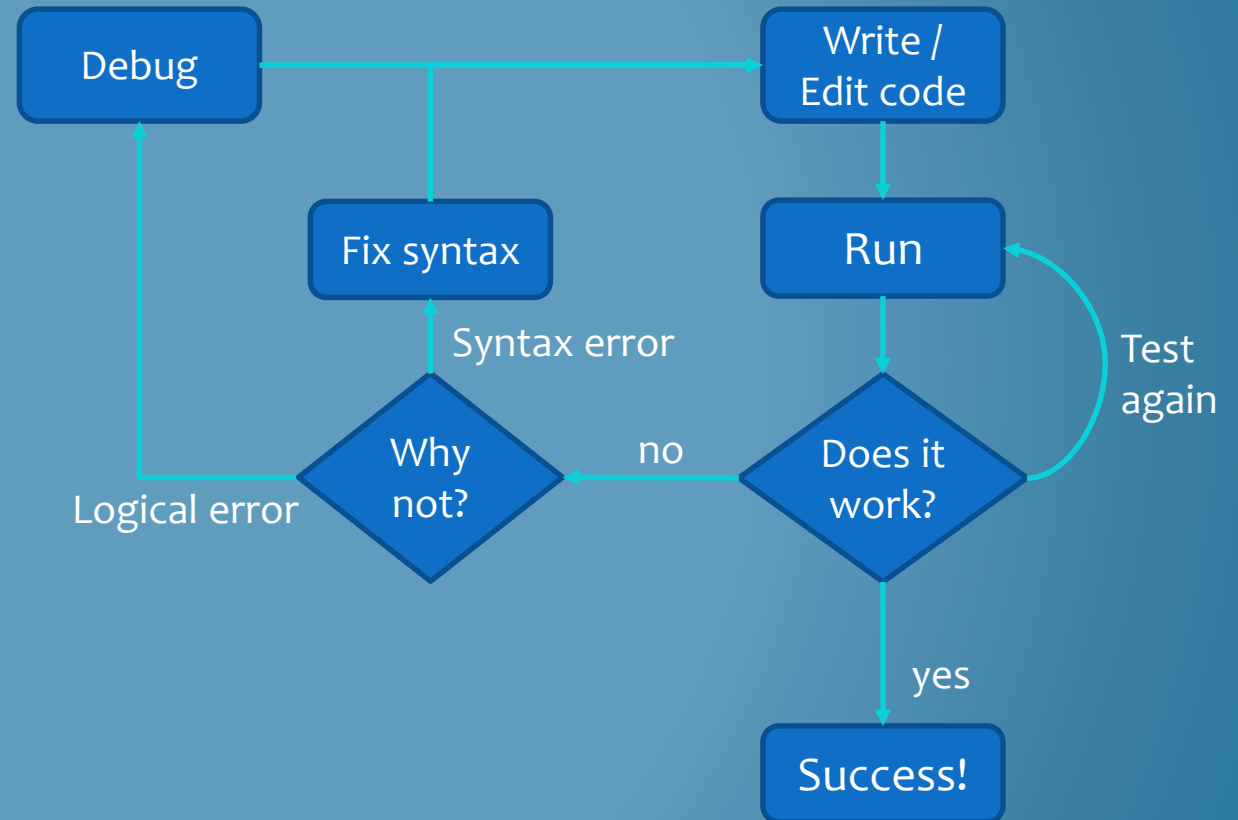
- Principles of Computer Programming
- Computer language concepts
- Software concepts
- **Programming workflow**
- Visual Studio concepts

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?

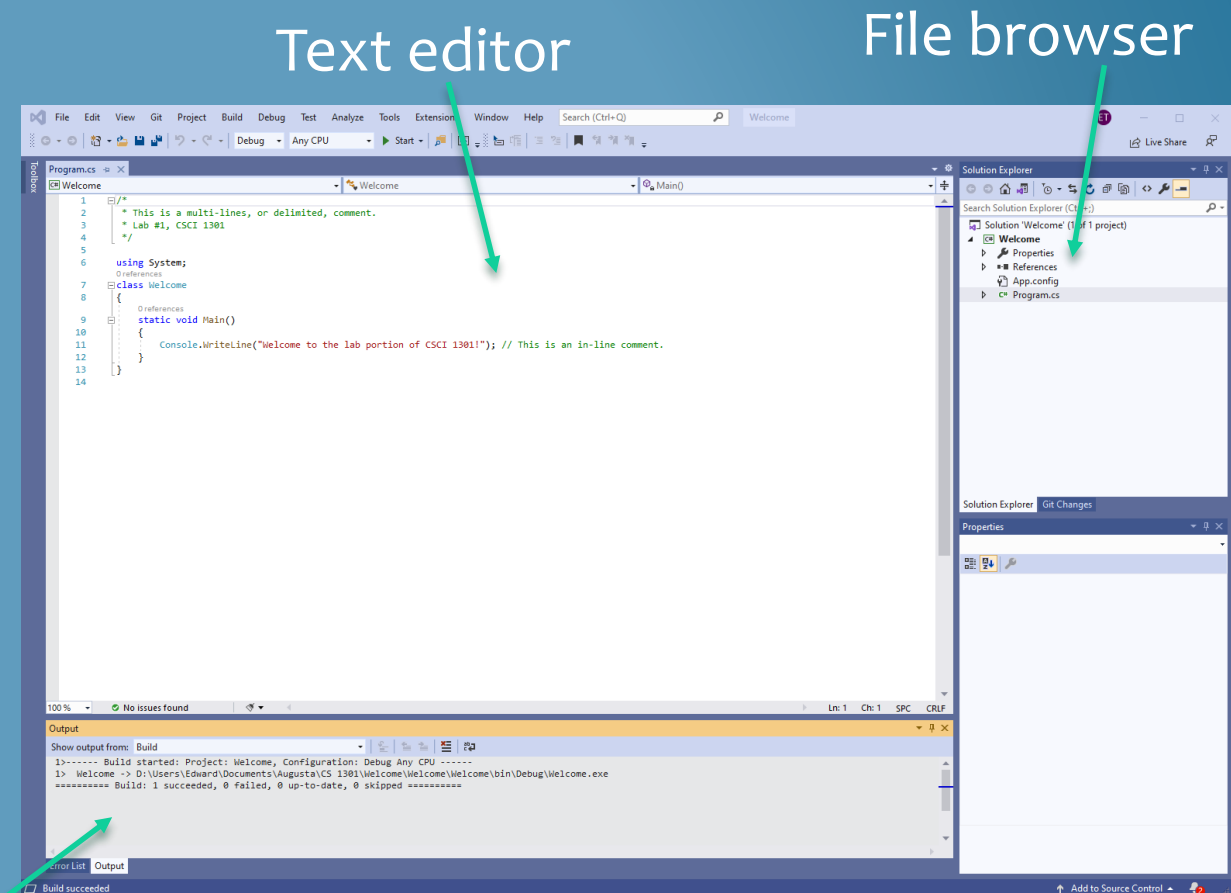


Outline

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

Visual Studio: An IDE

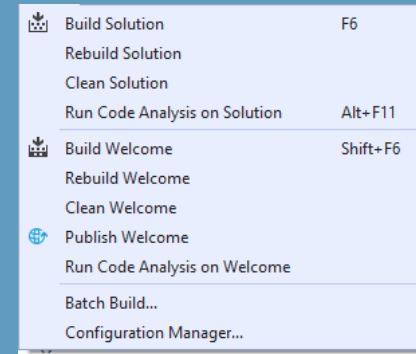
- Integrated Development Environment: A text editor, compiler, debugger, file browser, other tools
- Can write, compile, run code without leaving Visual Studio
- Tools to help you navigate and edit code



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

