C++ Programming

Instructor: Rita Kuo
Office: CS 520E
Phone: Ext. 4405
E-mail: rita.kuo@uvu.edu

Mapping zyBooks Chapters

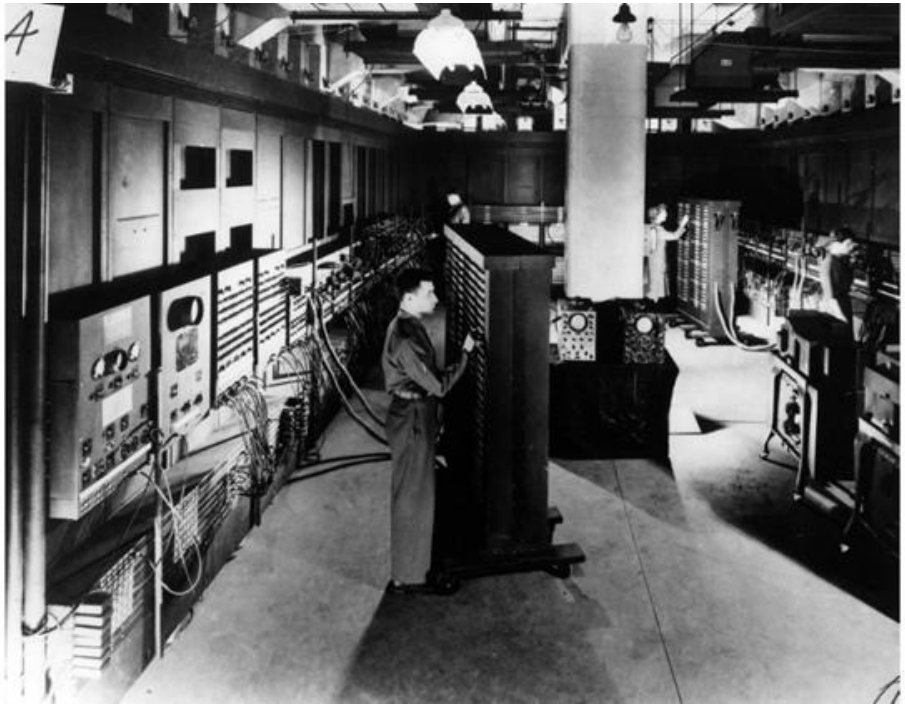
Topics on the slides	Chapters in zyBooks
Computer Language History	1.1
Computer Program Basics	
Programming Basics in C++	1.3
A Simple C++ Program Form	1.3
Basic Input/Output	1.3
Comments, Formatting, and Debugging	1.4, 1.5, 2.23

Not in the slides but required in the PA assignment: 1.6, 1.7

Introduction to C++

Computer History

- First Generation (1951 ~ 1959) - Hardware
  - Built using vacuum tubes to store information



ENIAC

Computer History

- First Generation (1951 ~ 1959) - Input and Output Devices
  - Input Device: IBM card (descendant of Hollerith card)

  - Output Device: punch card or line printer

Programming Language

- What is Language
  - The ability to acquire and use complex systems of communication

    ~ Wikipedia
  - Examples: English, Spanish, Mandarin, Japanese, etc.
- Constructed Language
  - A language whose phonology, grammar, and vocabulary have been consciously devised for human or human-like communication

    ~ Wikipedia
  - Examples: Klingon (Star Trek), Quenya (LOTR), etc.
- Computer Language
  - A formal constructed language designed to communicate instructions to a machine

    ~ Wikipedia

Types of Programming Language

- Machine Language

```
4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00
B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

- Assembly Language

```
;****************************************
code    segment
        assume  cs:code,ds:code
        org     100h
;----------------------------------------
start:  jmp     begin
mes     db      'Hi, I learn assembly.$'
begin:  mov     dx,offset mes
        mov     ah,9
        int     21h
        mov     ax,4c00h
        int     21h
```

- Higher Level Language
  - C, Basic, Java, Python

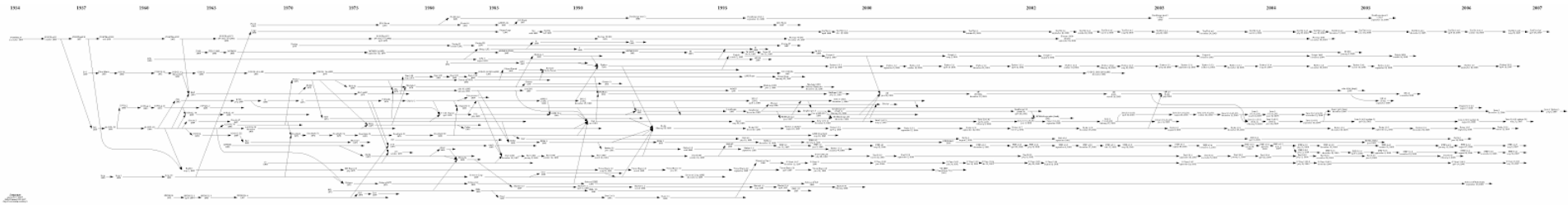
```
Public Function Add(a As Double, b As Double) As Double
    Add = a + b
End Function

Public Function Minus(a As Double, b As Double) As Double
    Minus = a - b
End Function
```

```
#include <stdio.h>
main( )
{
    int i=0 ;
    i=i+1;
    printf("i=%d\n",i) ;
}
```

History of Higher Level Language

- Computer Languages History
  - http://www.levenez.com/lang/



- Important Languages
  - FORTRAN: FORmula TRANslator
  - LISP: LISt Processor
- Popular Languages: Basic, C, C++, Java, etc.
  - https://www.tiobe.com/tiobe-index/

History of Higher Level Language

- Con
  -
- Imp
  -
  -
- Pop
  -

TIOBE Index - TIOBE

tiobe.com/tiobe-index/

TIOBE (the software quality company)

About us | Knowledge | News | Coding Standards | TIOBE Index | Contact

Products | Quality Models | Markets | Schedule a demo

Aug 2023	Aug 2022	Change	Programming Language	Ratings	Change
1	1		Python	13.33%	-2.30%
2	2		C	11.41%	-3.35%
3	4	^	C++	10.63%	+0.49%
4	3	v	Java	10.33%	-2.14%
5	5		C#	7.04%	+1.64%
6	8	^	JavaScript	3.29%	+0.89%
7	6	v	Visual Basic	2.63%	-2.26%
8	9	^	SQL	1.53%	-0.14%
9	7	v	Assembly language	1.34%	-1.41%
10	10		PHP	1.27%	-0.09%
11	21	^^	Scratch	1.22%	+0.63%

History of C++

- Unix and Unix-Like Systems
  - Developed at Bell Laboratories in 1969
  - Was written in assembly language
    → painful to debug and hard to enhance
  - Language B was used for Unix development
    → not well-suited to the machine
- C Language
  - A by-product of the UNIX operating system
  - Was extended from B
  - Was stable enough by 1973 that Unix could be rewritten in C
  - In 1978, Brian Kernighan and Dennis Ritchie at AT&T Bell Labs published a book describing a new high-level language with the simple name C

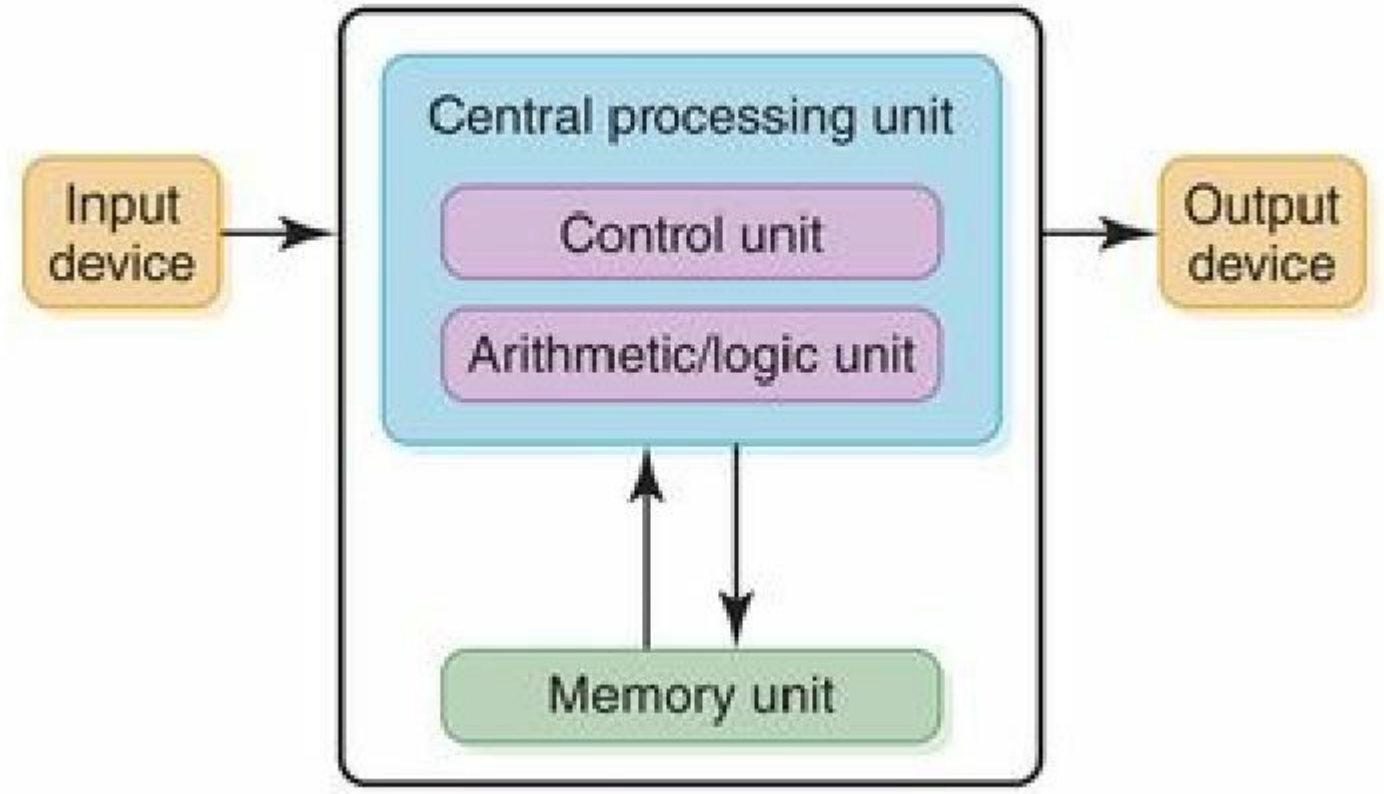
History of C++

- Object-Oriented Programming
  - How to enhance the reusability and code maintenance in large-scale program
  - The important concepts are introduced in Simula in 1960s
- C++ Language
  - In 1985, Bjarne Stroustrup published a book describing a C-based language called C++
  - Adding constructs to support object-oriented programming, along with other improvements
  - ++: comes from ++ being an operator in C that increases a number → an increase or improvement over C

Computer Program Basics

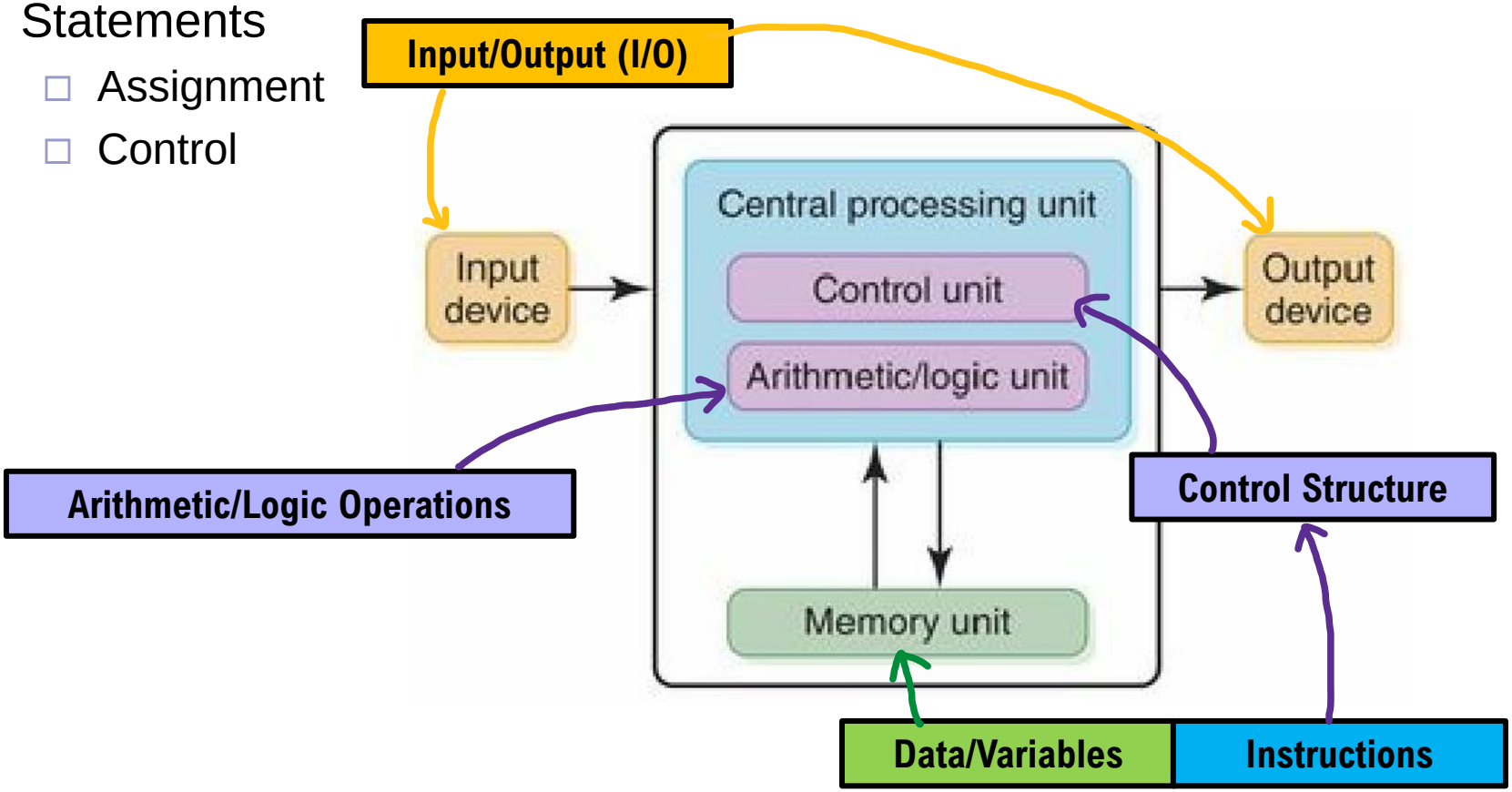
The von Neumann Architecture

- Describes a design architecture for an electronic digital computer with these components



Elements in Programming Language

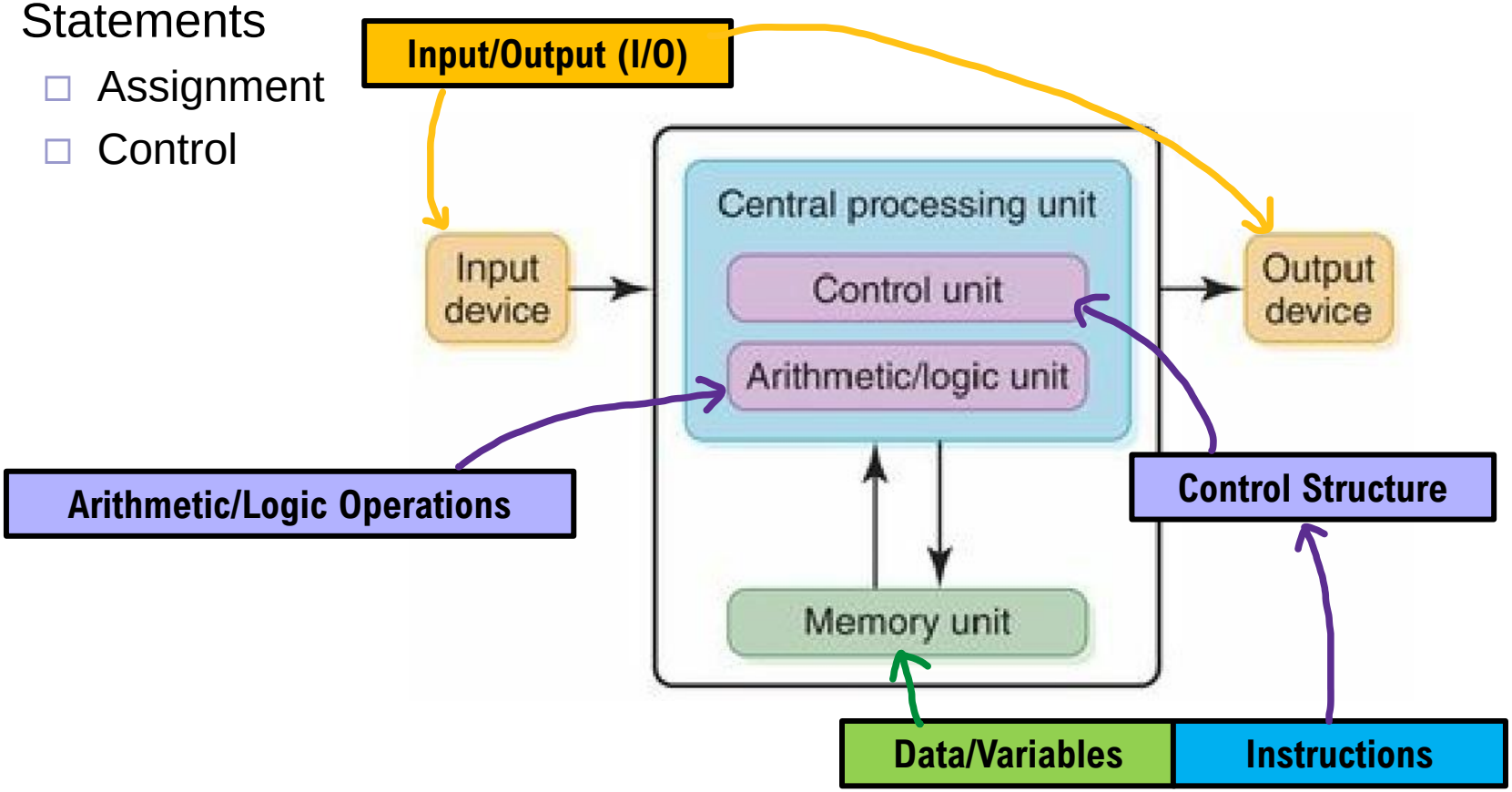
- Input/Output (I/O)
- Variables
- Expression
- Statements
  - Assignment
  - Control



Elements in Programming Language

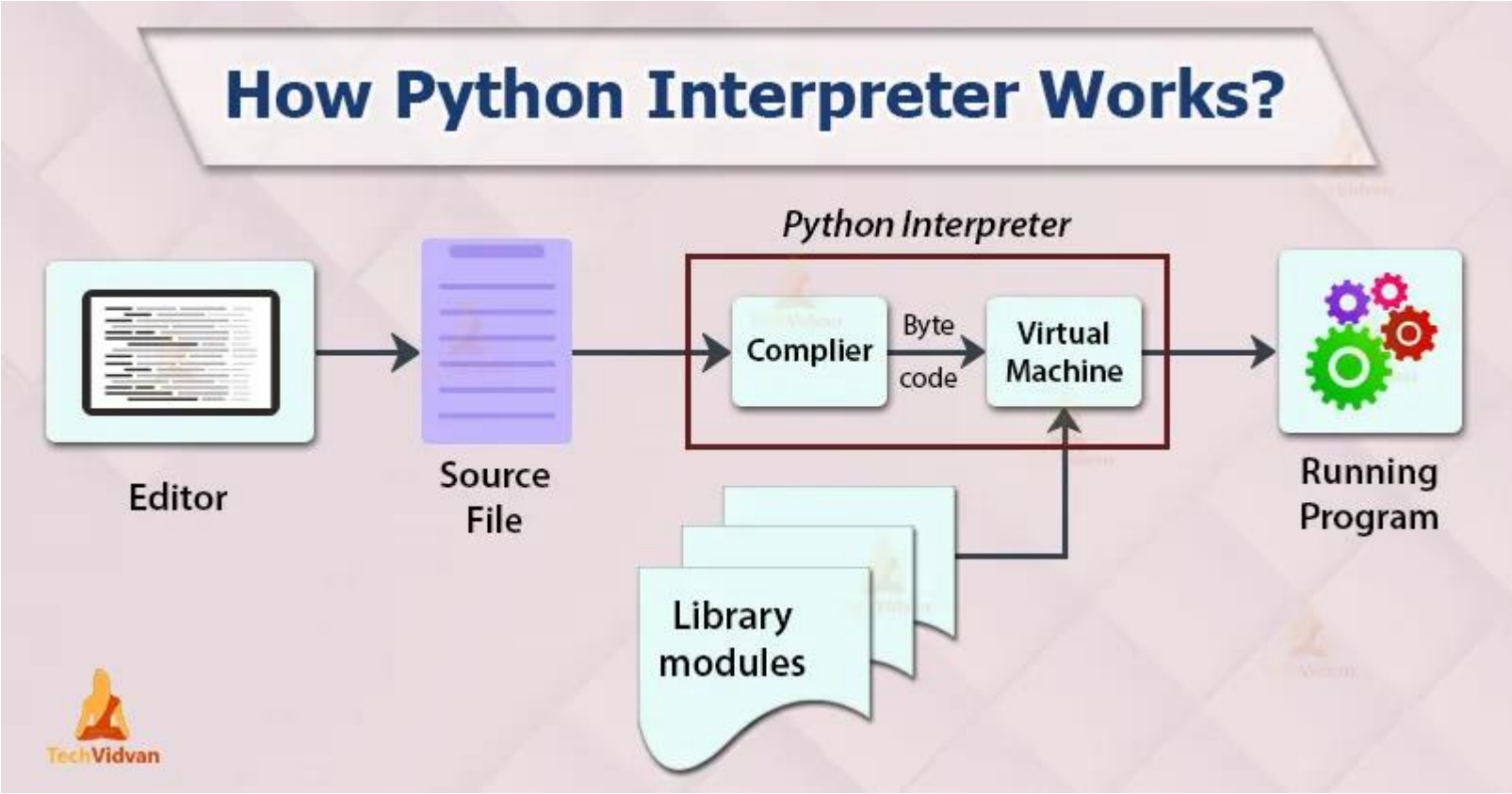
- Input/Output (I/O)
- Variables
- Expression
- Statements
  - Assignment
  - Control

Review in Python



Programming Basics in C++

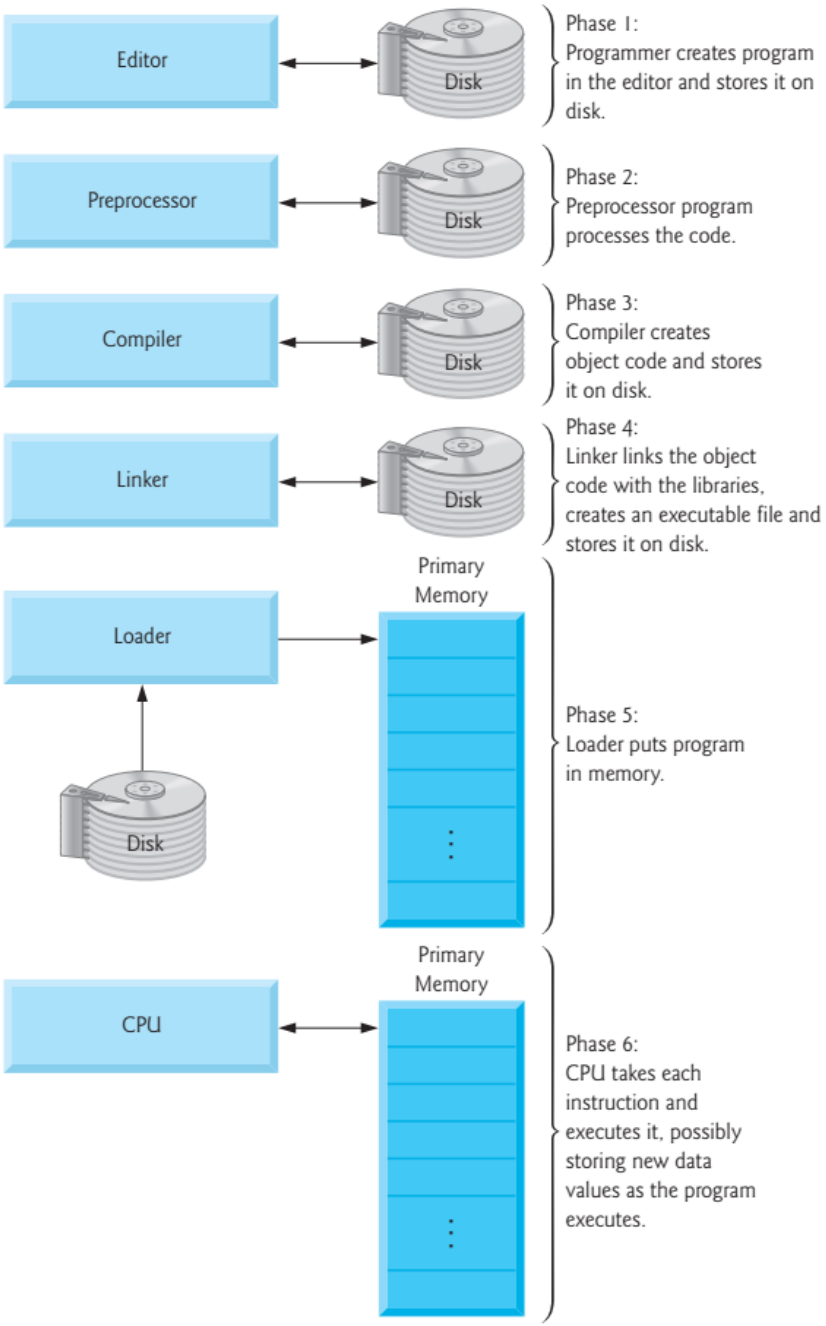
Review – How Python Interpreter Works



https://techvidvan.com/tutorials/python-interpreter/

C++ Working Environment

- A text-editor - write source code
  - VSCode, Atom, etc.
- A compiler - translate source code into machine language
  - Linux: GNU C++ Compiler
  - Windows: Visual Studio, MinGW
  - Mac: Xcode
- A shell - a way to interact with the kernel; a means to execute the program (Unix)

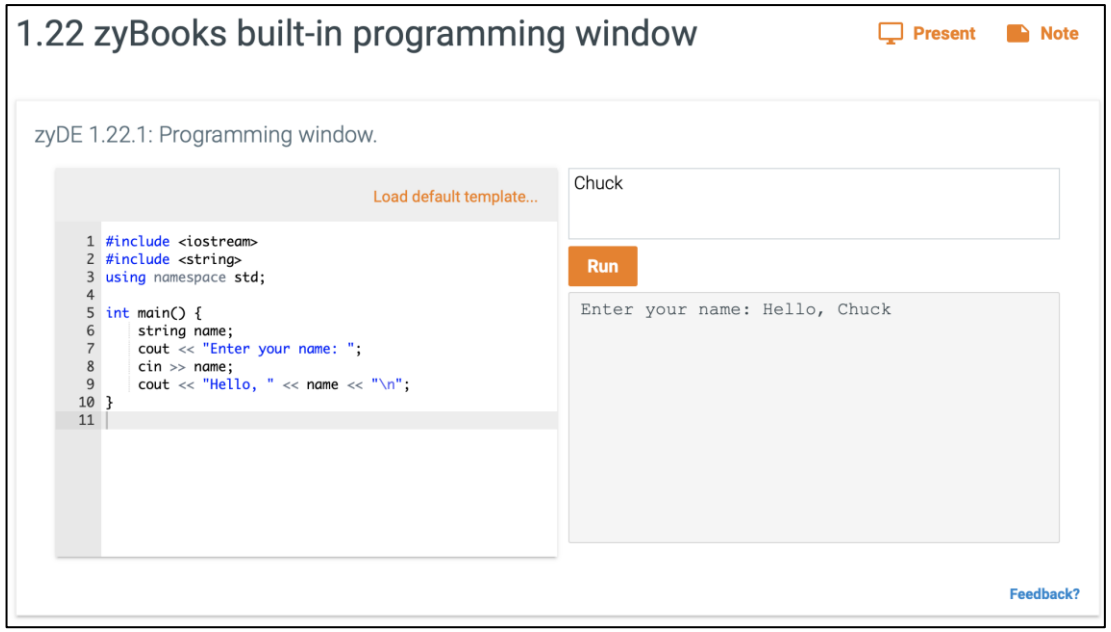


Online C++ Working Environment

- Repl.it
  - Go to https://replit.com
  - Create an account
  - Be sure to make your projects private
  - Click + New repl
- OnlineGDB (https://www.onlinegdb.com/)
- https://www.codiva.io/
- More on https://arnemertz.github.io/online-compilers/

zyBooks

- Method
  - Click any zyBooks assignment link in your learning management system (Do not go to the zyBooks website and create a new account)
  - Subscribe
- A subscription is $89. Students may begin subscribing on Aug 09, 2023 and the cutoff to subscribe is Dec 02, 2023. Subscriptions will last until Dec 30, 2023.



Project 0: Hello World

- Demonstrate that you can build and execute a simple C++ program.
- Scoring:
  - 5 points if you use an online compiler such as replit.com, cpp.sh, or onlinegdb.com
  - 10 points if you install a compiler, such as Visual Studio, clang, or g++. (Note that VSCode is NOT a compiler, it is an editor.)
- Turn in options:
  - 1. Submit a screen shot of your compile and execution.
  - 2. Demonstrate it in class the second day of class.

A Simple C++ Program Form

The First Program

- A simple C++ program form

```
directives

int main()
{
    statements
}
```

- Example

```
#include <iostream>
using namespace std;

int main() {
  int wage;

  wage = 20;

  cout << "Salary is ";
  cout << wage * 40 * 52;
  cout << endl;

  return 0;
}
```

Running the program on zyDE 1.3.1 or other online platforms

The First Program

- A simple C++ program form

```
directives

int main()
{
    statements
}
```

- Example

```
#include <iostream>
using namespace std;

int main() {
  int wage;

  wage = 20;

  cout << "Salary is ";
  cout << wage * 40 * 52;
  cout << endl;

  return 0;
}
```

Directives

- A language construct that specifies how a compiler should process its input
- In a C/C++ program, directives usually begin with a # character, which distinguishes them from other items.

`#include <iostream>`

- The information in `<iostream>` libraries are "included" into the program before it is compiled

`<iostream>`

- Contains information about C++'s console I/O library

The First Program

- A simple C++ program form

```
directives

int main()
{
    statements
}
```

- Example

```
#include <iostream>
using namespace std;

int main() {
  int wage;

  wage = 20;

  cout << "Salary is ";
  cout << wage * 40 * 52;
  cout << endl;

  return 0;
}
```

Namespace

- Allow us to group named entities that otherwise would have global scope into narrower scopes
- Multiple namespace blocks with the same name are allowed.

`std`

- The namespace defined in **`<iostream>`**
- **`cout`** is an object defined in the **`std`** namespace

If without **`using namespace std`**

- **`cout << "Salary is ";`** should be revised to **`std::cout << "Salary is ";`**

https://www.geeksforgeeks.org/namespace-in-c/

The First Program

- A simple C++ program form

```
directives

int main()
{
    statements
}
```

- Example

```
#include <iostream>
using namespace std;

int main() {
  int wage;

  wage = 20;

  cout << "Salary is ";
  cout << wage * 40 * 52;
  cout << endl;

  return 0;
}
```

A program starts in `main()` function

- Execute the statements within braces `{}`
- One statement at a time
- Each statement ends with a semicolon, as English sentences end with a period

Functions

- A C++ program is a collection of functions
- The form of a basic function:

```
return-val name-of-func (args) {
        body of function
}
```

The First Program

- A simple C++ program form

```
directives

int main()
{
    statements
}
```

- Example

```
#include <iostream>
using namespace std;

int main() {
  int wage;

  wage = 20;

  cout << "Salary is ";
  cout << wage * 40 * 52;
  cout << endl;

  return 0;
}
```

`return 0`

- Ends the program with no error
- Return a non-zero value in the main function usually indicate that there is an error occurred in the program
- Is optional in C++.

The First Program

- A simple C++ program form

```
directives

int main()
{
    statements
}
```

- Example

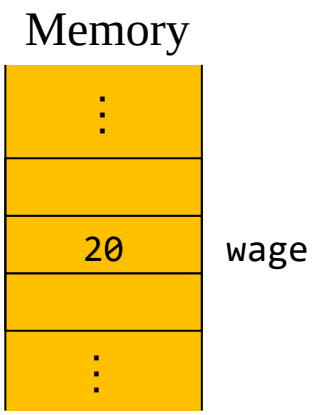
```
#include <iostream>
using namespace std;

int main() {
  int wage;

  wage = 20;

  cout << "Salary is ";
  cout << wage * 40 * 52;
  cout << endl;

  return 0;
}
```



`int wage;`

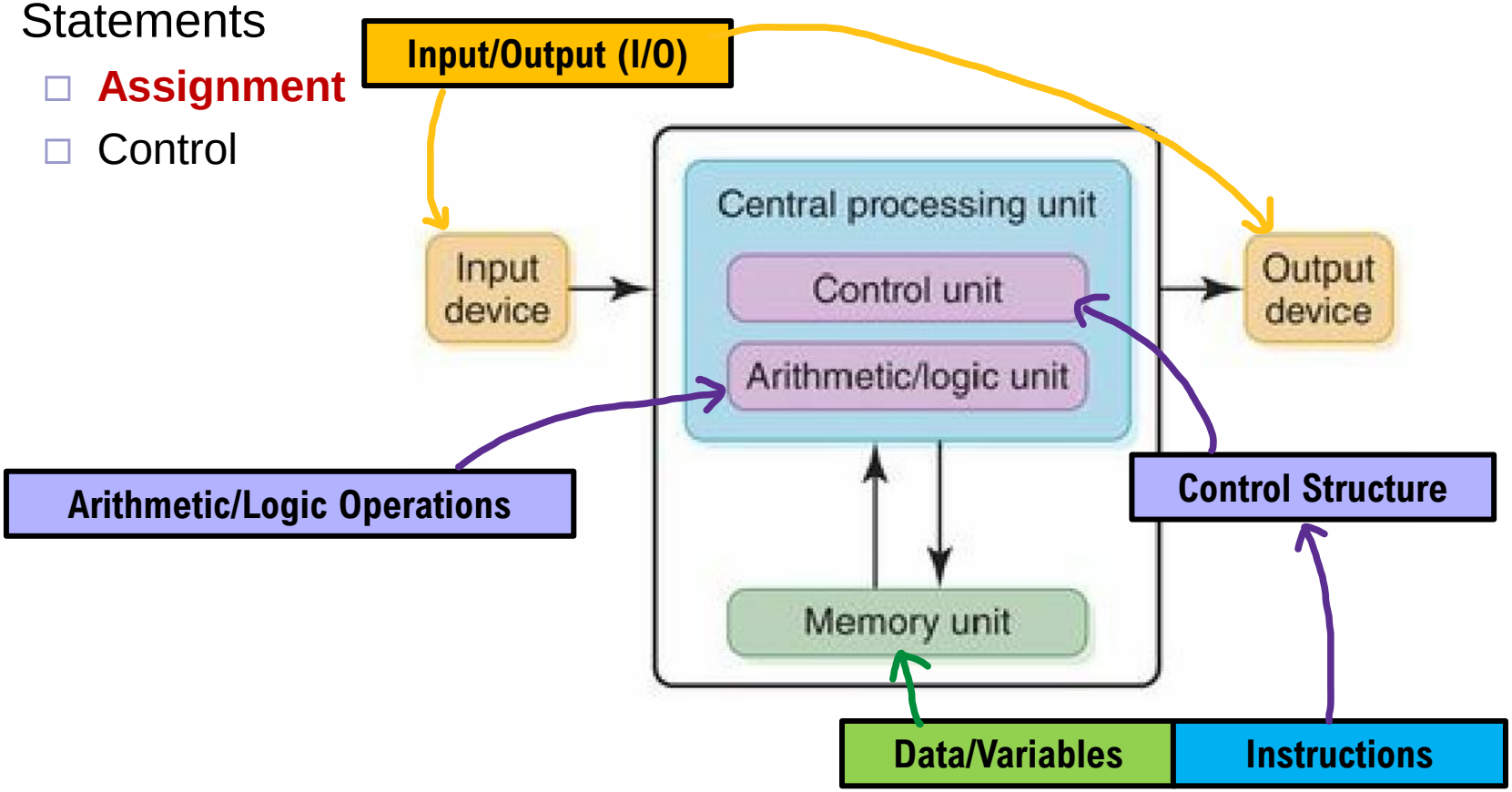
- Declares and integer variable
- The symbolic name of the integer variable is "wage"

`wage = 20;`

- Assign value 20 to variable – wage.

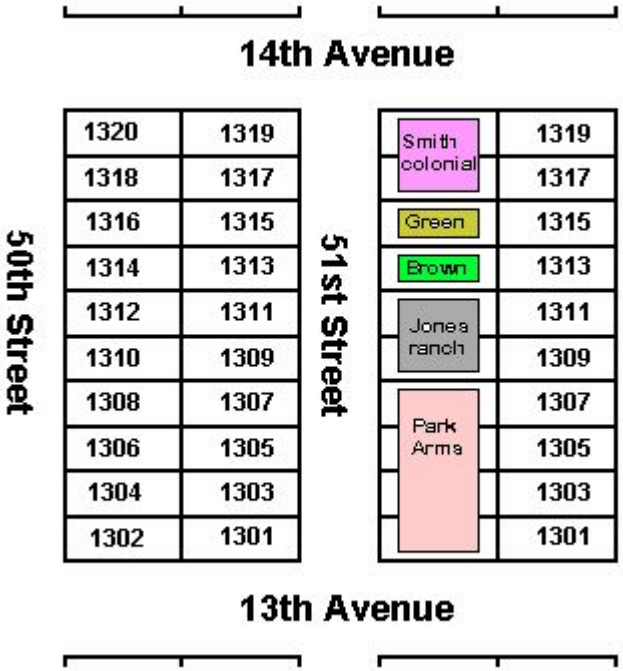
Review - Elements in Programming Language

- Input/Output (I/O)
- **Variables**
- Expression
- Statements
  - **Assignment**
  - Control



Variables

- A storage area in memory and its symbolic name
- The storage area contains a value that is referenced via the symbolic name
- Being a variable, the value stored in memory can change



- Example:
  http://www.bernstein-plus-sons.com/.dowling/Prog_Lang_Module/Computer_Memory.html

Variables

- Types
  - Specify what kind of data it will hold
  - Basic data types are **integers** (`short`, `int`, `long`), **real numbers** (`float` or `double`), or **characters** (`char`).
  - `int`: hold integer values, i.e., whole numbers such as 7, -11, 0, etc.
    - The largest `int` value is typically 2,147,483,647 but can be as small as 32,767
  - `float`: can store numbers with digits after the decimal point, such as 379.125
    - Slower than `int` in arithmetic operation
    - Is often an approximation of the number. E.g., `0.1` in a float variable might be 0.09999999999999987 stored in the system.
  - **Variables must be declared before they can be used**

Variables

- Declaration
  - Announce the properties of variables
  - Only need to declare variable's type once. Once declared it is immutable
  - Consist of a type name and a list of variables
  - Example:

```
int sum;
int fahr, celsius;
```

  - Because the variables must be declared first, the simple C program form can be rewritten as

```
directives

int main()
{
    declarations
    statements
}
```

Variables

- Identifiers
  - A name created by a programmer for an item like a variable or function
- Valid identifier in C++
  - Be a sequence of
    - Letters (`a`-`z`, `A`-`Z`)
    - Underscores (`_`)
    - Digits (`0`-`9`)
  - Start with a letter or underscore
  - Can not use reserved words (keywords) in the language.
    E.g., `for`, `int`, `return`, etc.
- Style guidelines for identifiers
  - Camel case: Lower camel case abuts multiple words, capitalizing each word except the first, as in `numApples` or `peopleOnBus`.
  - Underscore separated: Words are lowercase and separated by an underscore, as in `num_apples` or `people_on_bus`.

Assignments

- Assignments
  - Ties the storage area and the symbolic name together
  - Example:

    x ← 6

    place 6 in memory and assigns it the name x.
    - **x** has the value of **6**
    - **6** is assigned to **x**
- Assignment in C/C++
  - General pattern for assignment:

    symbolic_name = value;
  - A variable can be given a value by means of assignment
  - Example:

    x = 6;
  - **=**: assignment operator, **not** equality in mathematics.
  - **6**: a constant

Assignments

- Assignment in C/C++
  - Assignments do **not** commute. This is wrong:

    `6 ← x` or `6 = x;`

    - Symbolic names (aka identifiers) must begin with a letter or underscore
    - `=`: the value in the right-hand side (RHS) will be assignment to the memory address in the left-hand side (LHS)
      → the program cannot hard-code addresses as memory is ultimately managed by the operating system.
  - The following statements are valid:

```
int x, y;
y = 3;
x = y;
```

    - In the first statement, `y` is on the LHS of the assignment operator and is the symbolic name
    - In the second statement, `y` is on the RHS of the assignment operator and the value stored at `y` is used
      → the variable `y` is evaluated when it is on the RHS

Assignments

- Declaration and Assignment
  - A variable must be declared before it is assigned a value or used in any other way
  - Correct:
    ```
    int height;
    height = 8;
    ```
  - Incorrect:
    ```
    height = 8;
    int height;
    ```
  - Correct: declare variable and initialize it in one step
    ```
    int height = 8;
    ```
  - Which variables are initialized?
    ```
    int height, length, width = 8;
    ```

Assignments

- Declaration and Assignment
  - General pattern for assignment:
    symbolic_name = value;
  - The left-hand side (LHS) and right-hand side (RHS) must match in type.
  - Mixing types is possible but not always safe.
  - Example:

```
int i;
float f;
i = 72.99;     // i is now 72
f = 136;       // f is now 136.0
```

The First Program

- A simple C++ program form

```
directives

int main()
{
    statements
}
```

- Example

```
#include <iostream>
using namespace std;

int main() {
  int wage;

  wage = 20;

  cout << "Salary is ";
  cout << wage * 40 * 52;
  cout << endl;

  return 0;
}
```

`cout` statement

- Output values on the right-hand side of `<<` operator

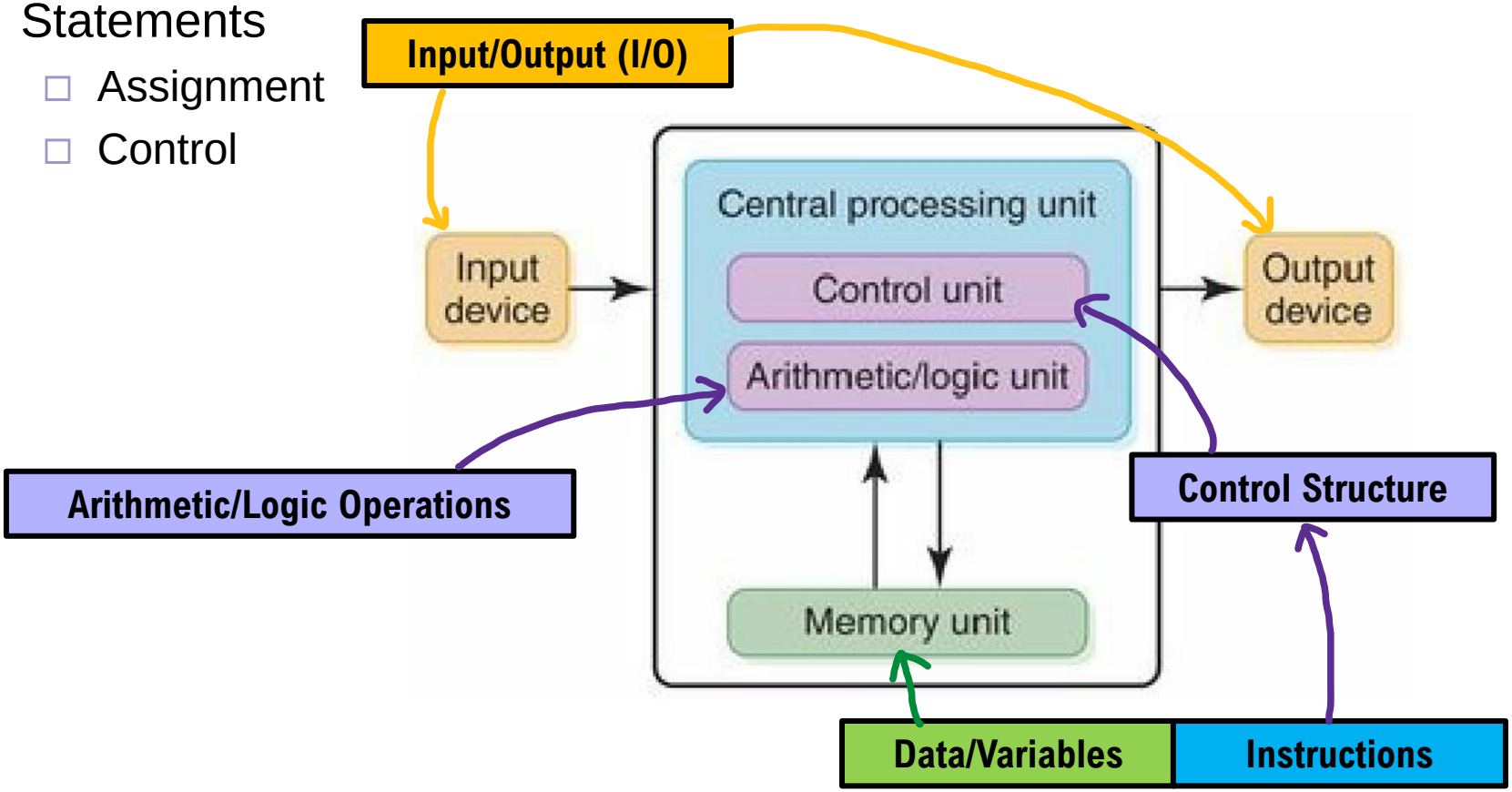
`endl`

- Stands for “end line”
- Prints a newline character to the console

Basic Input/Output

Review - Elements in Programming Language

- **Input/Output (I/O)**
- Variables
- Expression
- Statements
  - Assignment
  - Control



Basic Input/Output in C++

- Input and output
  - Are performed in the form of a sequence of bytes or more commonly known as streams
  - Input Stream: If the direction of flow of bytes is from the device (for example, keyboard) to the main memory then this process is called input.
  - Output Stream: If the direction of flow of bytes is opposite, i.e. from main memory to device (display screen) then this process is called output.
- `<iostream>` Header
  - Stands for standard input-output stream
  - Contains definitions of objects like `cin` (for standard input) and `cout` (for standard output)

https://www.geeksforgeeks.org/basic-input-output-c/

Basic Input/Output in C++

- Standard output stream (`cout`)
  - The C++ `cout` statement is the instance of the `ostream` class.
  - It is used to produce output on the standard output device which is usually the display screen.
  - The data needed to be displayed on the screen is inserted in the standard output stream (`cout`) using the stream insertion operator (`<<`).
  - Example: output a string
    ```
    cout << "Salary is ";
    ```
  - Example: output a single variable
    ```
    cout << wage;
    ```
  - Example: output an expression
    ```
    cout << wage * 40 * 52;
    ```
  - Example: output multiple items with one statemen
    ```
    cout << "Wage is: " << wage << endl;
    ```

https://www.geeksforgeeks.org/basic-input-output-c/

Basic Input/Output in C++

- standard input stream (`cin`):
  - C++ `cin` statement is the instance of the class `istream`
  - Is used to read input from the standard input device which is usually a keyboard.
  - The stream extraction operator (`>>`) is used along with the object `cin` for reading inputs.
  - Example:
    ```
    cin >> wage;
    ```

Practice zyDE 1.3.2

https://www.geeksforgeeks.org/basic-input-output-c/

Comments, Formatting, and Debugging

Comments

- The text a programmer adds to code, to be read by humans to better understand the code but ignored by the compiler.
- Single-line comment
  - Starts with `//` and includes all the following text on that line.
  - Single-line comments commonly appear after a statement on the same line.
- Multi-line comment
  - Starts with `/*` and ends with `*/`, where all text between `/*` and `*/` is part of the comment.
  - A multi-line comment is also known as a block comment.

Whitespace

- A compiler ignores most whitespace
  - Blank spaces (space and tab characters) between items within a statement
  - Blank lines between statements (called newlines)
- Good practice of using whitespaces
  - Use blank lines to separate conceptually distinct statements.
  - Indent lines the same amount.
  - Align items to reduce visual clutter.
  - Use a single space before and after any operators like =, +, *, or << to make statements more readable.

Style Guidelines

Errors and Warnings

- Syntax Errors
  - Example:

```
main.cpp:6:27: error: expected ';' after expression
   cout << "Traffic today"
                          ^
                          ;
```

  - Sometime it is unclear:

```
main.cpp:6:7: error: expected ';' after expression
  cout "Traffic today";
      ^
      ;
```

- Good practice for fixing errors
  - Focus on FIRST error message, ignoring the rest.
  - Look at reported line of first error message. If error found, fix. Else, look at previous few lines.
  - Compile, repeat.

Errors and Warnings

- Logic Errors
  - The program compiled perfectly but isn't working
- Compiler Warning
  - Does not stop the compiler from creating an executable program
  - Indicates a possible logic error. E.g., divided by 0