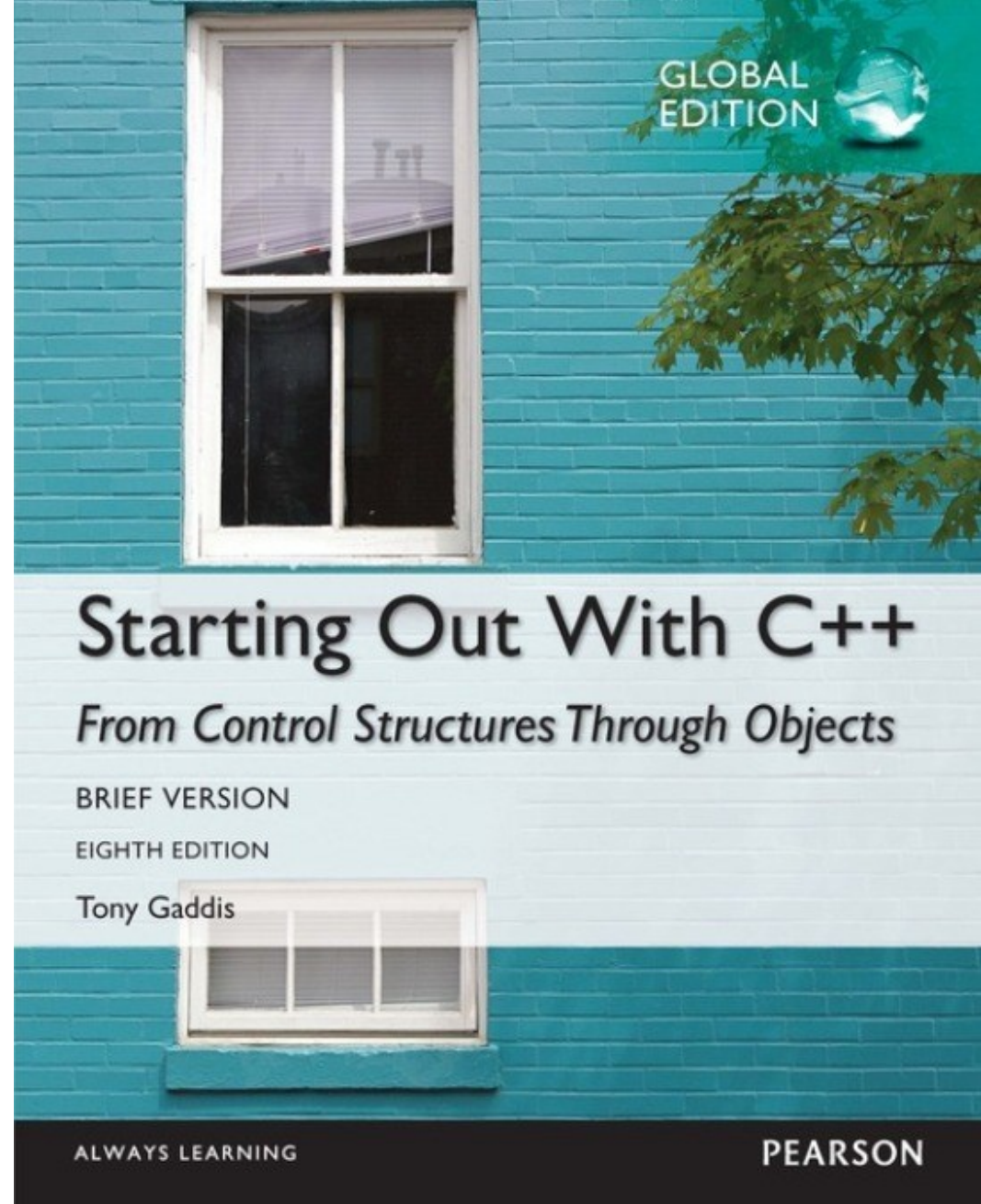


Chapter 1:

Introduction to Computers and Programming



Why Program?

Computer – programmable machine designed to follow instructions

Program – instructions in computer memory to make it do something

Programmer – person who writes instructions (programs) to make computer perform a task

SO, without programmers, no programs;
without programs, a computer cannot do anything

Main Hardware Component Categories:

1. Central Processing Unit (CPU)
2. Main Memory
3. Secondary Memory / Storage
4. Input Devices
5. Output Devices

Main Hardware Component Categories

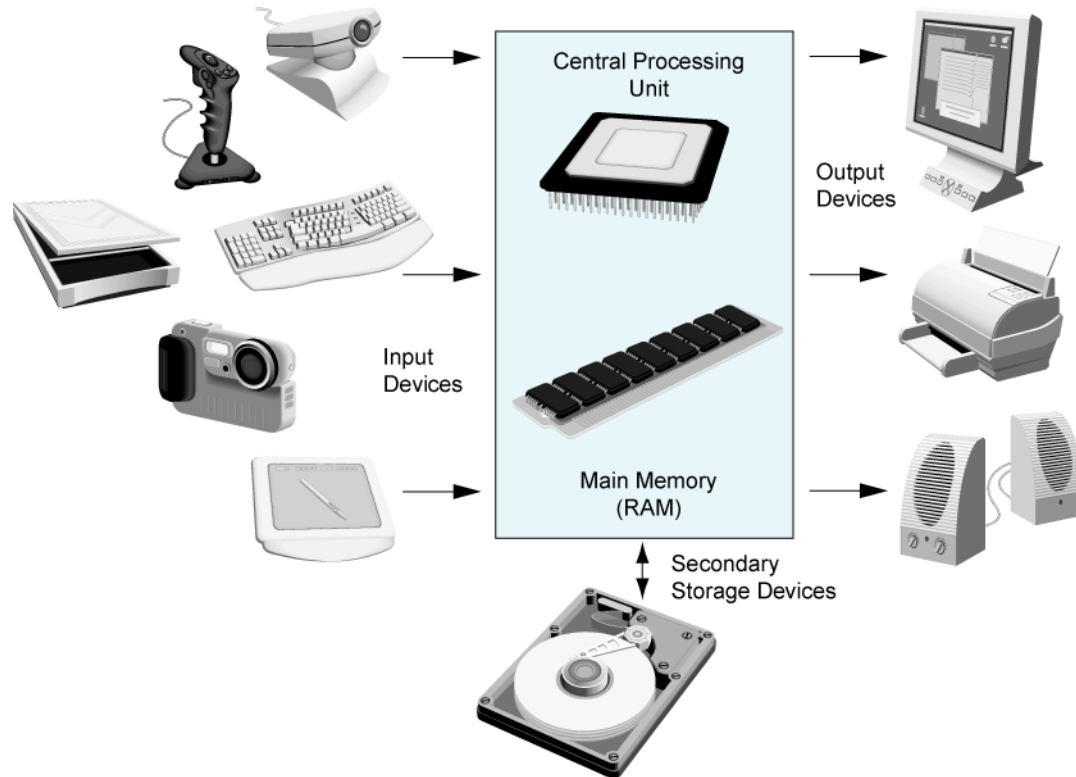


Figure 1-2

Central Processing Unit (CPU)

Comprised of:

Control Unit

- Retrieves and decodes program instructions

- Coordinates activities of all other parts of computer

Arithmetic & Logic Unit

- Hardware optimized for high-speed numeric calculation

- Hardware designed for true/false, yes/no decisions

CPU Organization

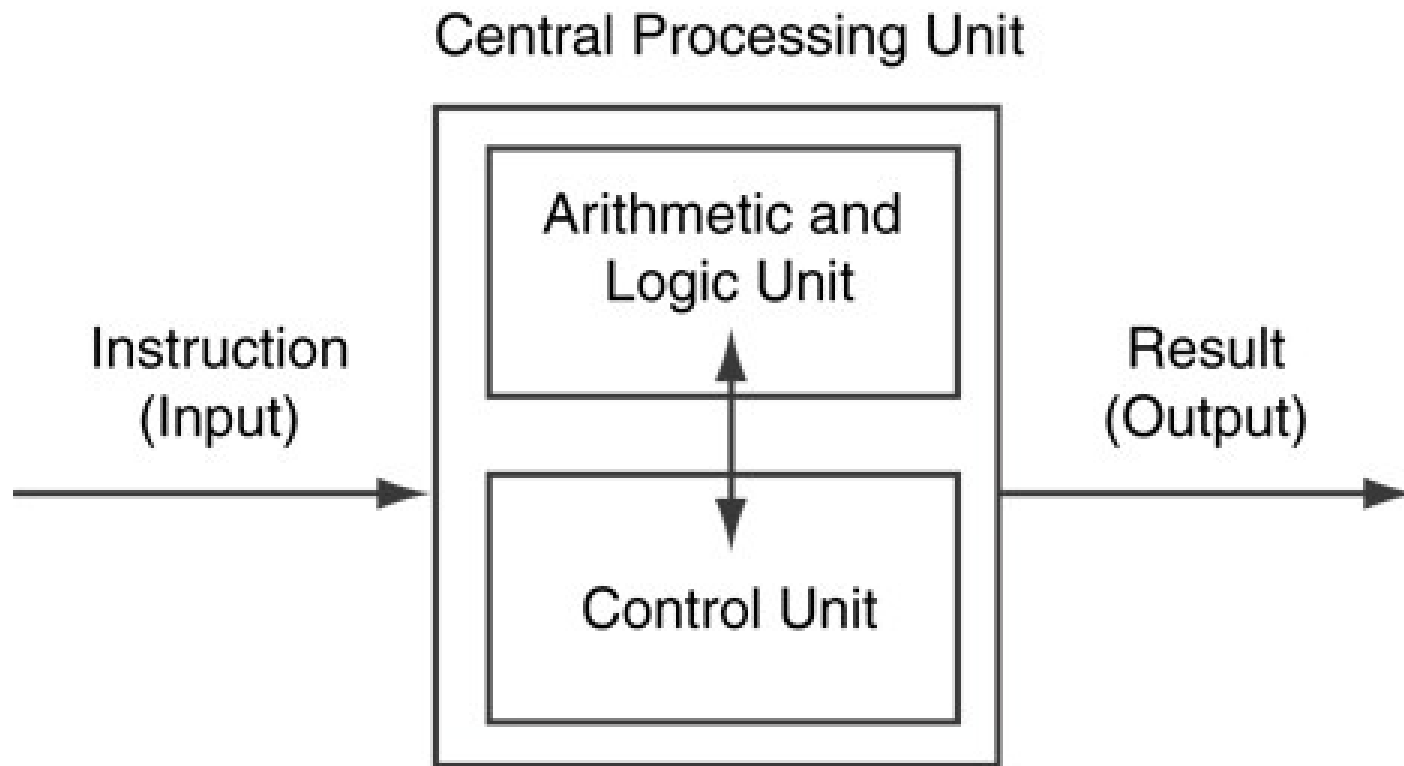


Figure 1-3

Main Memory

- It is volatile. Main memory is erased when program terminates or computer is turned off
- Also called Random Access Memory (RAM)
- Organized as follows:
 - bit: smallest piece of memory. Has values 0 (off, false) or 1 (on, true)
 - byte: 8 consecutive bits. Bytes have addresses.

Main Memory

- Addresses – Each byte in memory is identified by a unique number known as an *address*.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16 149	17	18	19
20	21	22	23 72	24	25	26	27	28	29

- In Figure 1-4, the number 149 is stored in the byte with the address 16, and the number 72 is stored at address 23.

Secondary Storage

- Non-volatile: data retained when program is not running or computer is turned off
- Comes in a variety of media:
 - magnetic: floppy disk, hard drive
 - optical: CD-ROM, DVD
 - Flash drives, connected to the USB port

Input Devices

- Devices that send information to the computer from outside
- Many devices can provide input:
 - Keyboard, mouse, scanner, digital camera, microphone
 - Disk drives, CD drives, and DVD drives

Output Devices

- Used to send information from the computer to the outside
- Many devices can be used for output
 - Computer screen, printer, speakers, disk drive, CD/DVD recorder, USB flash drive

Software-Programs That Run on a Computer

- Categories of software:
 - System software: programs that manage the computer hardware and the programs that run on them. *Examples:* operating systems, utility programs, software development tools
 - Application software: programs that provide services to the user. *Examples :* word processing, games, programs to solve specific problems

Programs and Programming Languages

- A program is a set of instructions that the computer follows to perform a task
- We start with an *algorithm*, which is a set of well-defined steps.
- Algorithm is not ready to be executed on the computer.
- The computer only executes *machine language* instructions

Machine Language

- Machine language instructions are binary numbers, such as

1011010000000101

- Rather than writing programs in machine language, programmers use *programming languages*.

Programs and Programming Languages

Types of languages:

- Low-level: used for communication with computer hardware directly. Often written in binary machine code (0's/1's) directly.
- High-level: closer to human language

High level (Easily read by humans)



Low level (machine language)
10100010 11101011



Some Well-Known Programming Languages (Table 1-1 on Page 10)

C++

BASIC

Ruby

Java

FORTRAN

Visual Basic

COBOL

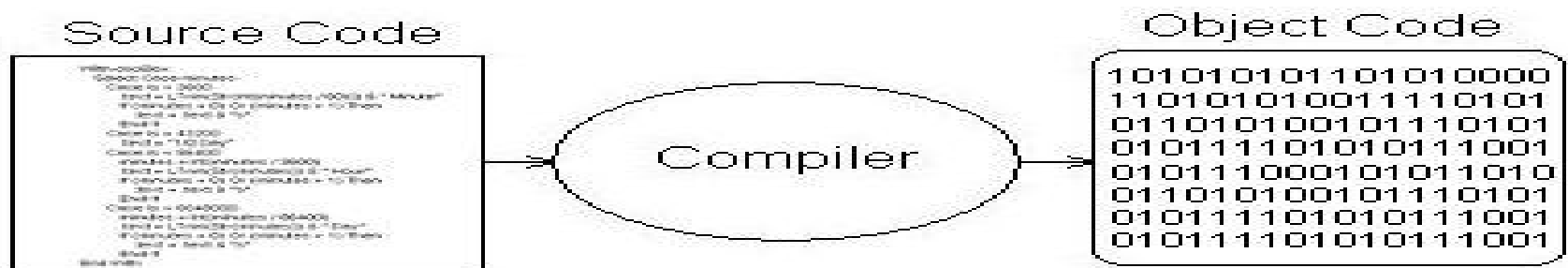
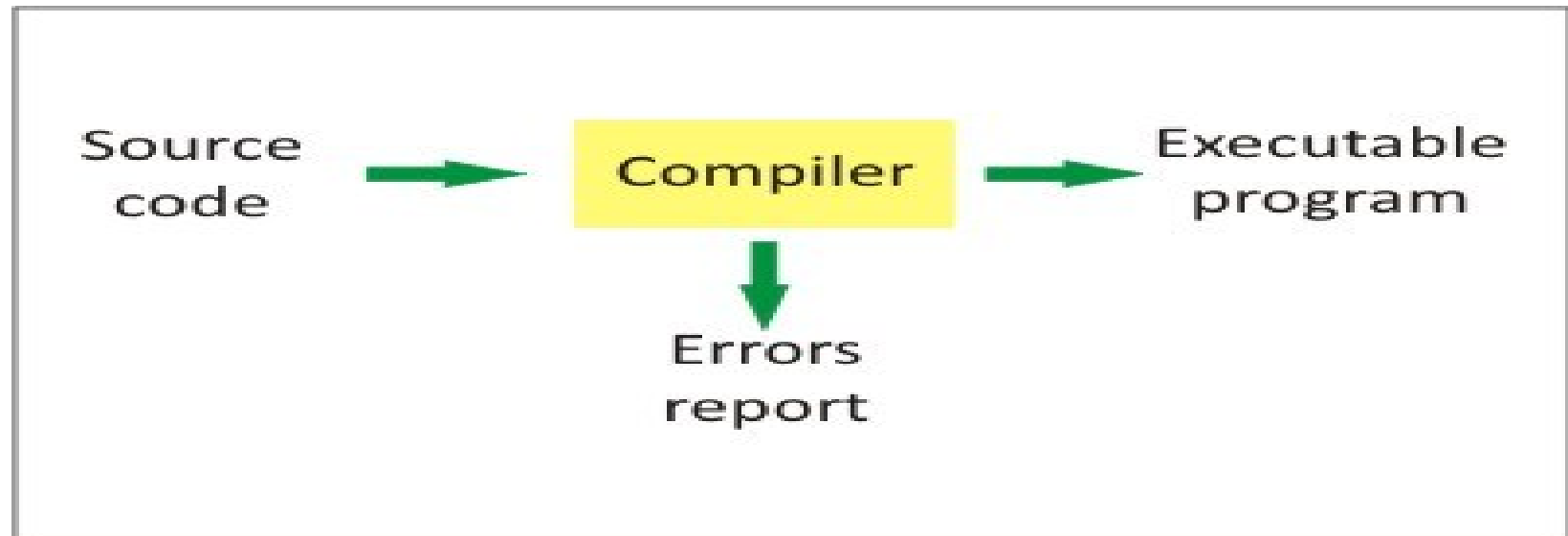
C#

JavaScript



Python

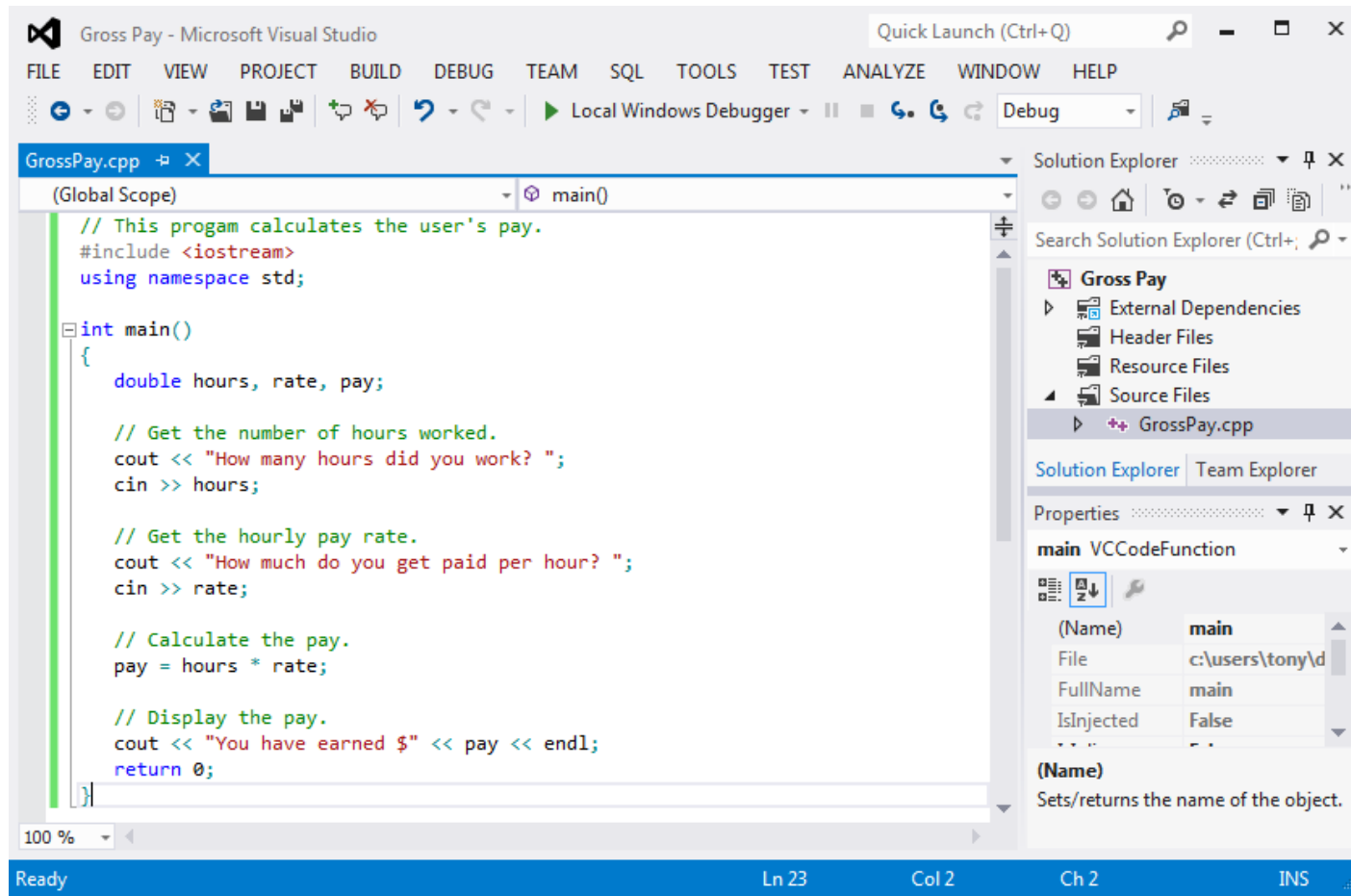
From High Level to Low Level



Integrated Development Environments (IDEs)

- An integrated development environment, or IDE, combine all the tools needed to write, compile, and debug a program into a single software application.
- Examples are Microsoft Visual C++, Turbo C++ Explorer, CodeWarrior, etc.

Integrated Development Environments (IDEs)



What is a Program Made of?

- Common elements in programming languages:
 - Key Words
 - Programmer-Defined Identifiers
 - Operators
 - Punctuation
 - Syntax

Key Words

- Also known as reserved words
- Have a special meaning in C++
- Can not be used for any other purpose
- Key words in the Program 1-1: `using,`
`namespace, int, double, and return`

Key Words

```
1  // This program calculates the user's pay.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      double hours, rate, pay;
8
9      // Get the number of hours worked.
10     cout << "How many hours did you work? ";
11     cin >> hours;
12
13     // Get the hourly pay rate.
14     cout << "How much do you get paid per hour? ";
15     cin >> rate;
16
17     // Calculate the pay.
18     pay = hours * rate;
19
20     // Display the pay.
21     cout << "You have earned $" << pay << endl;
22     return 0;
23 }
```

Programmer-Defined Identifiers

- Names made up by the programmer
- Not part of the C++ language
- Used to represent various things: variables (memory locations), functions, etc.
- In Program 1-1: `hours`, `rate`, and `pay`.

Operators

- Used to perform operations on data
- Many types of operators:
 - Arithmetic - ex: $+$, $-$, $*$, $/$
 - Assignment – ex: $=$
- Some operators in Program1-1:
 $<< \gg = *$

Operators

```
1  // This program calculates the user's pay.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      double hours, rate, pay;
8
9      // Get the number of hours worked.
10     cout << "How many hours did you work? ";
11     cin >> hours;
12
13     // Get the hourly pay rate.
14     cout << "How much do you get paid per hour? ";
15     cin >> rate;
16
17     // Calculate the pay.
18     pay = hours * rate;
19
20     // Display the pay.
21     cout << "You have earned $" << pay << endl;
22     return 0;
23 }
```

Punctuation

- Characters that mark the end of a statement, or that separate items in a list
- In Program 1-1: , and ;

Punctuation

```
1  // This program calculates the user's pay.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      double hours, rate, pay;
8
9      // Get the number of hours worked.
10     cout << "How many hours did you work? ";
11     cin >> hours;
12
13     // Get the hourly pay rate.
14     cout << "How much do you get paid per hour? ";
15     cin >> rate;
16
17     // Calculate the pay.
18     pay = hours * rate;
19
20     // Display the pay.
21     cout << "You have earned $" << pay << endl;
22     return 0;
23 }
```

Syntax

- The rules of grammar that must be followed when writing a program
- Controls the use of key words, operators, programmer-defined symbols, and punctuation

Variables

- A variable is a named storage location in the computer's memory for holding a piece of data.
- In Program 1-1 we used three variables:
 - The **hours** variable was used to hold the hours worked
 - The **rate** variable was used to hold the pay rate
 - The **pay** variable was used to hold the gross pay

The Programming Process

1. Clearly define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools such as a hierarchy chart, flowcharts, or pseudocode to create a model of the program.
4. Check the model for logical errors.
5. Type the code, save it, and compile it.
6. Correct any errors found during compilation. Repeat Steps 5 and 6 as many times as necessary.
7. Run the program with test data for input.
8. Correct any errors found while running the program. Repeat Steps 5 through 8 as many times as necessary.
9. Validate the results of the program.

Algorithm

An algorithm is a procedure or formula for solving a problem. The word derives from the name of the mathematician, Mohammed ibn-Musa al-Khwarizmi



An algorithm is a set of well-defined steps for performing a task or solving a problem.

Example of algorithm (average of two numbers)

1. Display a message on the screen “Enter first number?”
2. Wait for the user to enter a number. Once the user enters a number, store it in memory in a variable call it X
3. Display a message on the screen “Enter second number?”
4. Wait for the user to enter a number. Once the user enters a number, store it in memory in a variable call it Y
5. Calculate total of X and Y in a variable call it S
 $S = X + Y$

6. Calculate the average in a variable call it A

$$A=S/2$$

7. Display a message on the screen “The average=” and display the value of A

The above instructions are called an **algorithm**.

Notice these steps are sequentially ordered. Step 1 should be performed before step 2, and so forth. It is important that these instructions be performed in their proper sequence.

Same algorithm (more formal)

1. Start
2. Print “Enter the first number”
3. Read x
4. Print “Enter the second number”
5. Read y
6. $s = x + y$
7. $a = s / 2$
8. Print a
9. End

Algorithm to calculate area of a rectangle

1. Start
2. Print “Enter the length”
3. Read x
4. Print “Enter the width”
5. Read y
6. $a = x * y$
7. Print a
8. End

Challenges

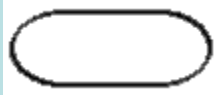
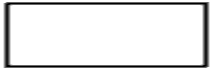

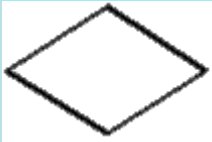


- Write an algorithm to calculate the area of a square
- Write an algorithm to calculate the area of a circle
- Write an algorithm to read a salary of an employee and deduct 15% as a tax, then display the net salary

Flowcharts

Both algorithm and flowchart describe how to carry out a process

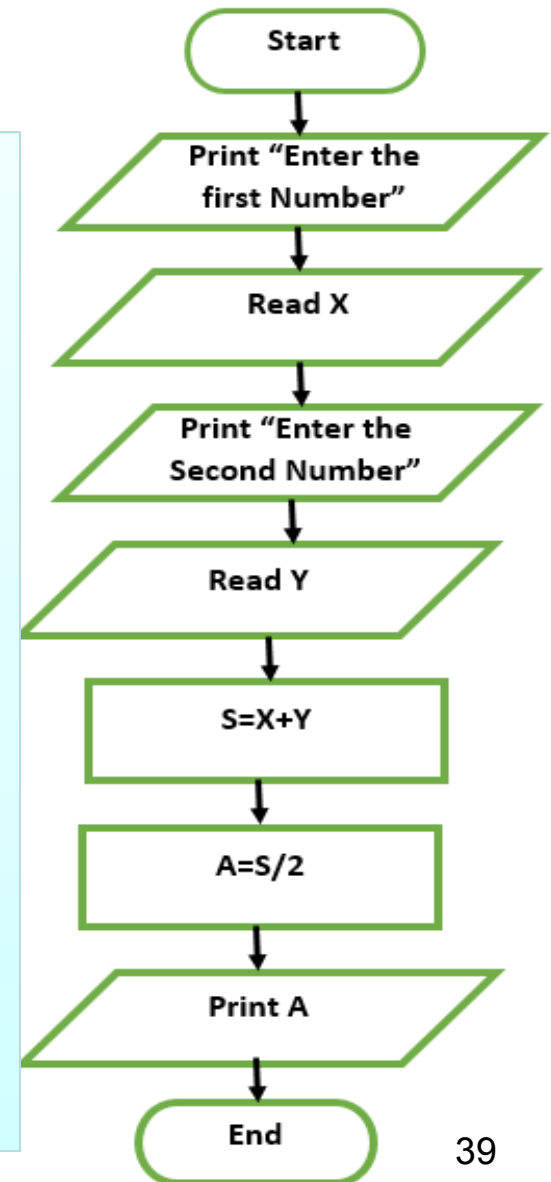
- Algorithm uses normal language to describe a step-by-step process on how to solve the problem
- Flowchart uses symbols to describe steps of the solution.
- Computer programmers use flowchart to describe the “flow” of a program.

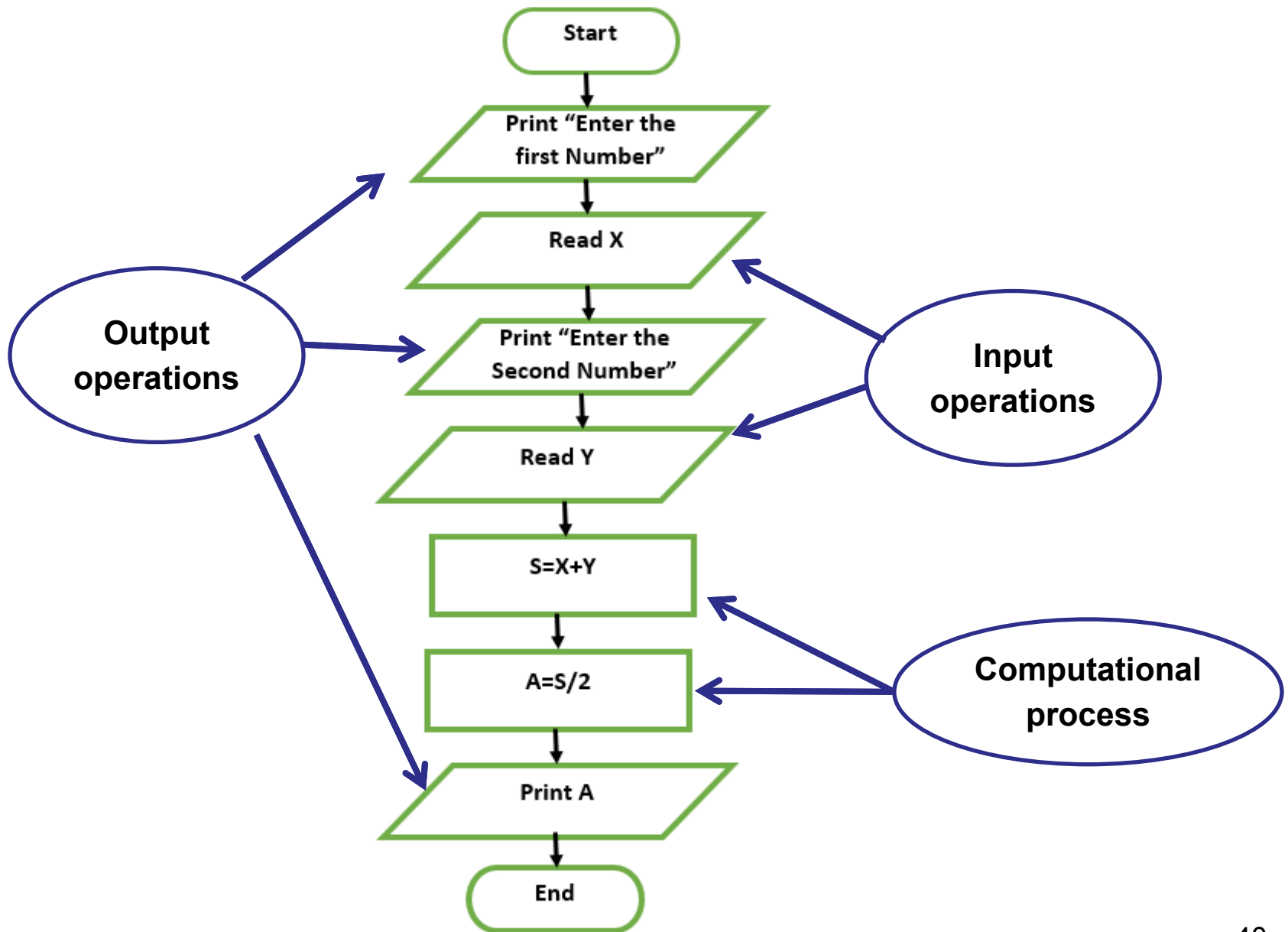
Flowchart Notations:

Symbol	Its meaning
	Start or end of the program
	Computational steps or processing
	Input or output operation
	Decision making and branching
	Connector or joining
	Arrows represent the sequence and direction of a program.

Algorithm and Flowcharts

1. Start
2. Print "Enter the first number".
3. Read X
4. Print "Enter the second number".
5. Read Y
6. $T = X + Y$
7. $A = T / 2$
8. Print A
9. End





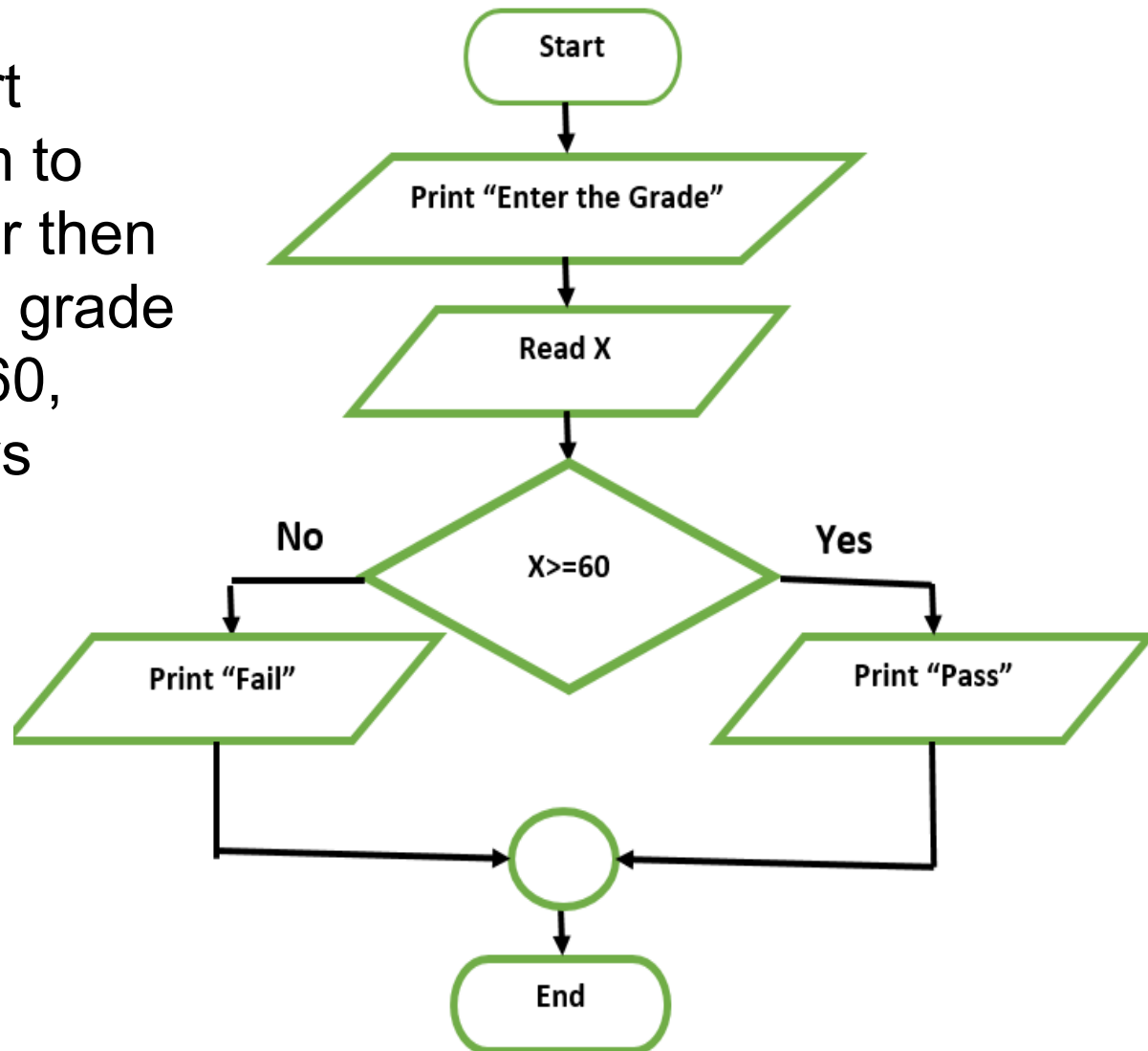
Challenges

- Draw a flowchart to calculate the area of a square
- Draw a flowchart to calculate the area of a circle
- Draw a flowchart to read a salary of an employee and deduct 15% as a tax, then display the net salary

Selection in algorithm and flowchart

The following flowchart describes an algorithm to read a grade from user then

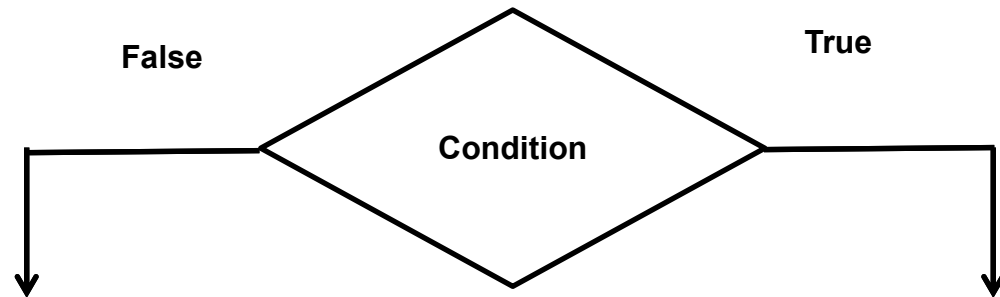
- display “Pass” if the grade is greater or equal 60,
- otherwise, it displays “Fail”



Selection is based on a condition

Condition has:

- Left side
- Relational operator
- Right side



X **<=** **230**

Left **Relational Operator** **Right**

Relational Operator	Its meaning
=	Equal
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal
!=	Not equal

Challenges

- Draw a flowchart to read a salary of an employee and deduct the tax as follows:
if the salary is greater than or equal to 10,000 the tax is 15% , otherwise the tax is 20%. Then, display the net salary
- Draw a flowchart to get the value of X and calculate Y according to the following:

$$Y = \begin{cases} X + 1 & X > 0 \\ X^2 + 2X + 10 & X \leq 0 \end{cases}$$