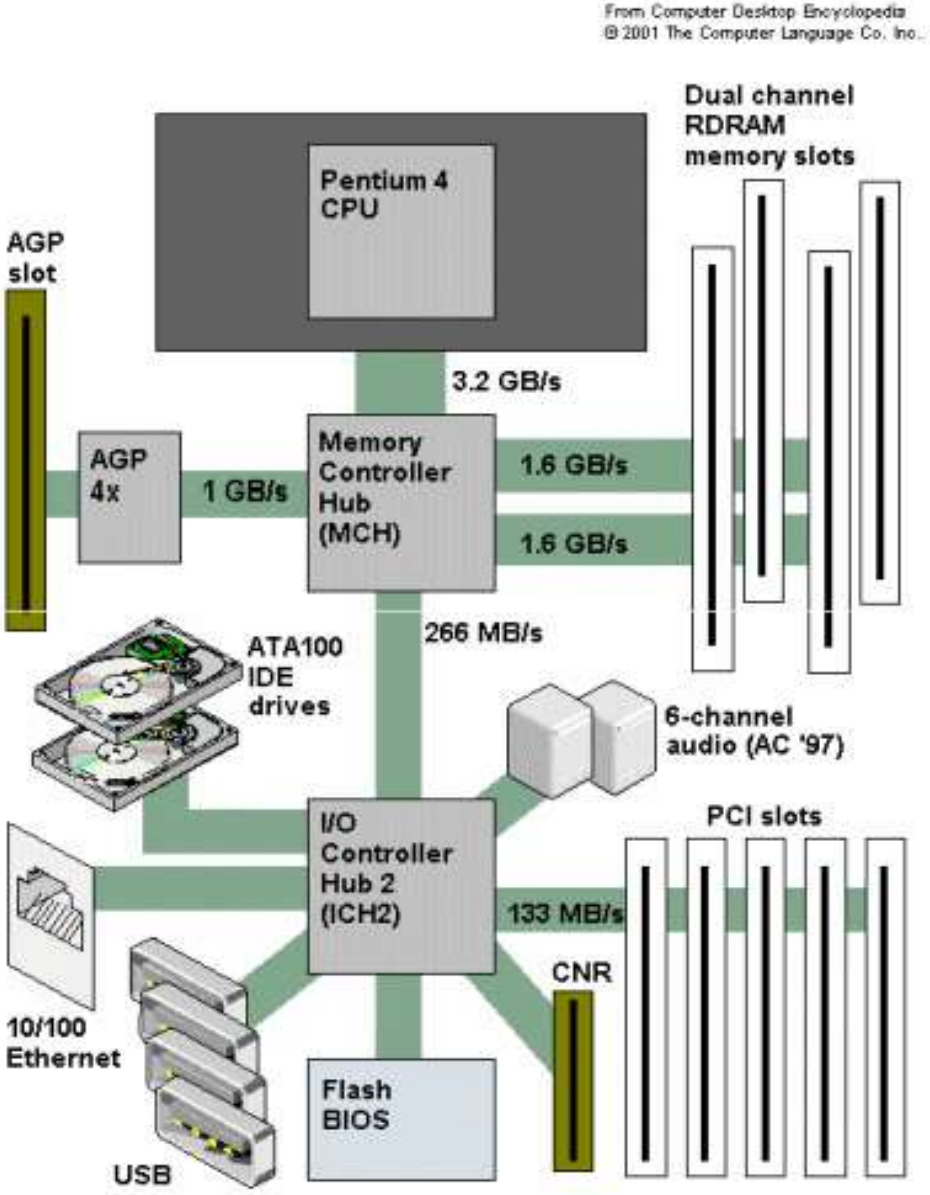
Architecture

Typical system architecture for a desktop PC



CPU (Central Processing Unit)

  - All computations take place here in order for the computer to perform a designated task.
  - It has a large number of registers which temporarily store data and programs (instructions).
  - It has functional units (circuitry) to carry out arithmetic and logic operations
  - It retrieves instructions from the memory, interprets (decodes) them, and performs the requested operation
  - Fetch → Decode → Execute cycle
- CPU is also referred to as the processor
- Computers may have multiple processors
- Modern processors are multi-core (multiple processors in one chip)

Main Memory

- Uses semiconductor technology
  - Allows direct access
- Memory sizes in the range of 256 MegaBytes to 8 GigaBytes are typical today.
- Some measures to be remembered
  - 1 K = 2^{10} (= 1024)
  - 1 M = 2^{20} (= one million approx.)
  - 1 G = 2^{30} (= one billion approx.)

I/O and Peripherals

- Input Device
  - Keyboard, Mouse, Scanner, Digital Camera
- Output Device
  - Monitor, Printer
- Storage Peripherals
  - Magnetic Disks: hard disk, floppy disk (obsolete)
    - Allows direct (semi-random) access
  - Optical Disks: CDROM, CD-RW, DVD
    - Allows direct (semi-random) access
  - Flash Memory: pen drives
    - Allows direct access
  - Magnetic Tape: DAT (obsolete)
    - Only sequential access

A Sample Configuration of a PC

- **Processor:** Intel® Core™ i3-530 Processor (2.93GHz 1333MHz 4MB)
- **Total memory:** 2 GB DDR3 1333MHz
- **Display type:** 23.0 " With integrated camera 0.3M 1920x1080
- **Hard drive device:** 320GB
- **Optical device:** DVD Recordable (Dual Layer)
- **Input Device:** Keyboard, Mouse
- **Ports:** USB, Infrared
- **Chipset …**
- **Graphics …**

Classification of Software

- **Two categories:**

1. **Application Software**
   - Used to solve a particular problem.
   - Editor, financial accounting, weather forecasting, etc.

2. **System Software**
   - Helps in running other programs.
   - Compiler, operating system, etc.

Computer Languages

- **Machine Language**
  - Expressed in binary.
  - Directly understood by the computer.
  - Not portable; varies from one machine type to another.
    - Program written for one type of machine will not run on another type of machine.
  - Difficult to use in writing programs.

Contd.

- **Assembly Language**
  - Mnemonic form of machine language.
  - Easier to use as compared to machine language.
    - For example, use “ADD” instead of “10110100”.
  - Not portable (like machine language).
  - Requires a translator program called *assembler*.

Contd.

- Assembly language is also difficult to use in writing programs.
  - Requires many instructions to solve a problem.

- Example: Find the average of three numbers.

```
MOV     A,X       ; A = X
ADD     A,Y       ; A = A + Y
ADD     A,Z       ; A = A + Z
DIV     A,3       ; A = A / 3
MOV     RES,A     ; RES = A
```

In C,

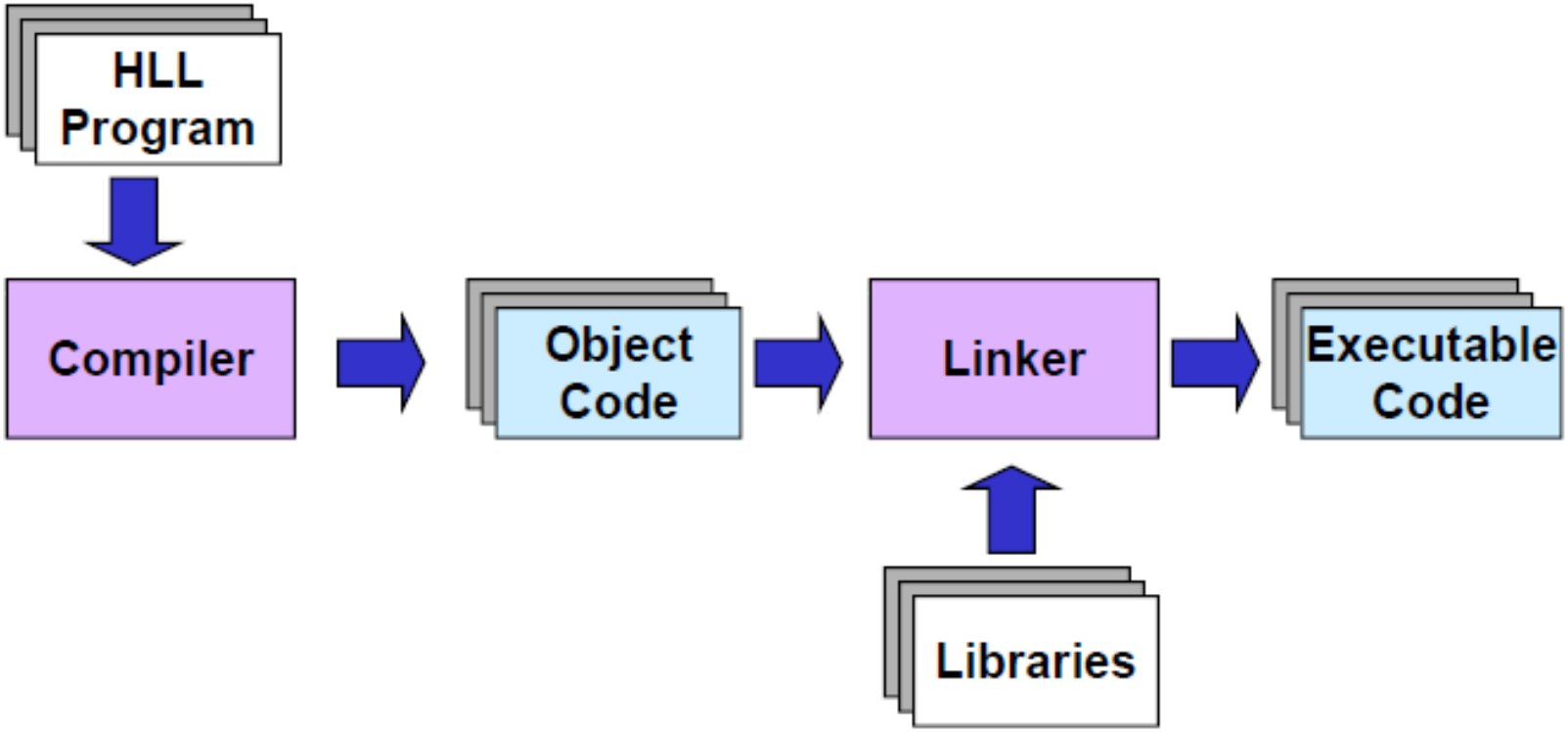
RES = (X + Y + Z) / 3

High-Level Language

- Machine language and assembly language are called low-level languages.
  - They are closer to the machine.
  - Difficult to use.

- High-level languages are easier to use.
  - They are closer to the programmer.
  - Examples:
    - Fortran, Cobol, C, C++, Java.
  - Requires an elaborate process of translation.
    - Using a software called *compiler*.
  - They are portable across platforms.

From HLL to executable



Operating Systems

- Makes the computer easy to use.
  - Basically the computer is very difficult to use.
  - Understands only machine language.
- Operating systems make computers easy to use.
- Categories of operating systems:
  - Single user
  - Multi user
    - Time sharing
    - Multitasking
    - Real time

Contd.

- Computers connected in a network.
- Many users may work on a computer.
  - Over the network.
  - At the same time.
  - CPU and other resources are shared among the different programs.
    - Called time sharing.
    - One program executes at a time.

How does a computer work?

- **Stored program concept.**
  - Main difference from a calculator.
- **What is a program?**
  - Set of instructions for carrying out a specific task.
- **Where are programs stored?**
  - In secondary memory, when first created.
  - Brought into main memory, during execution.

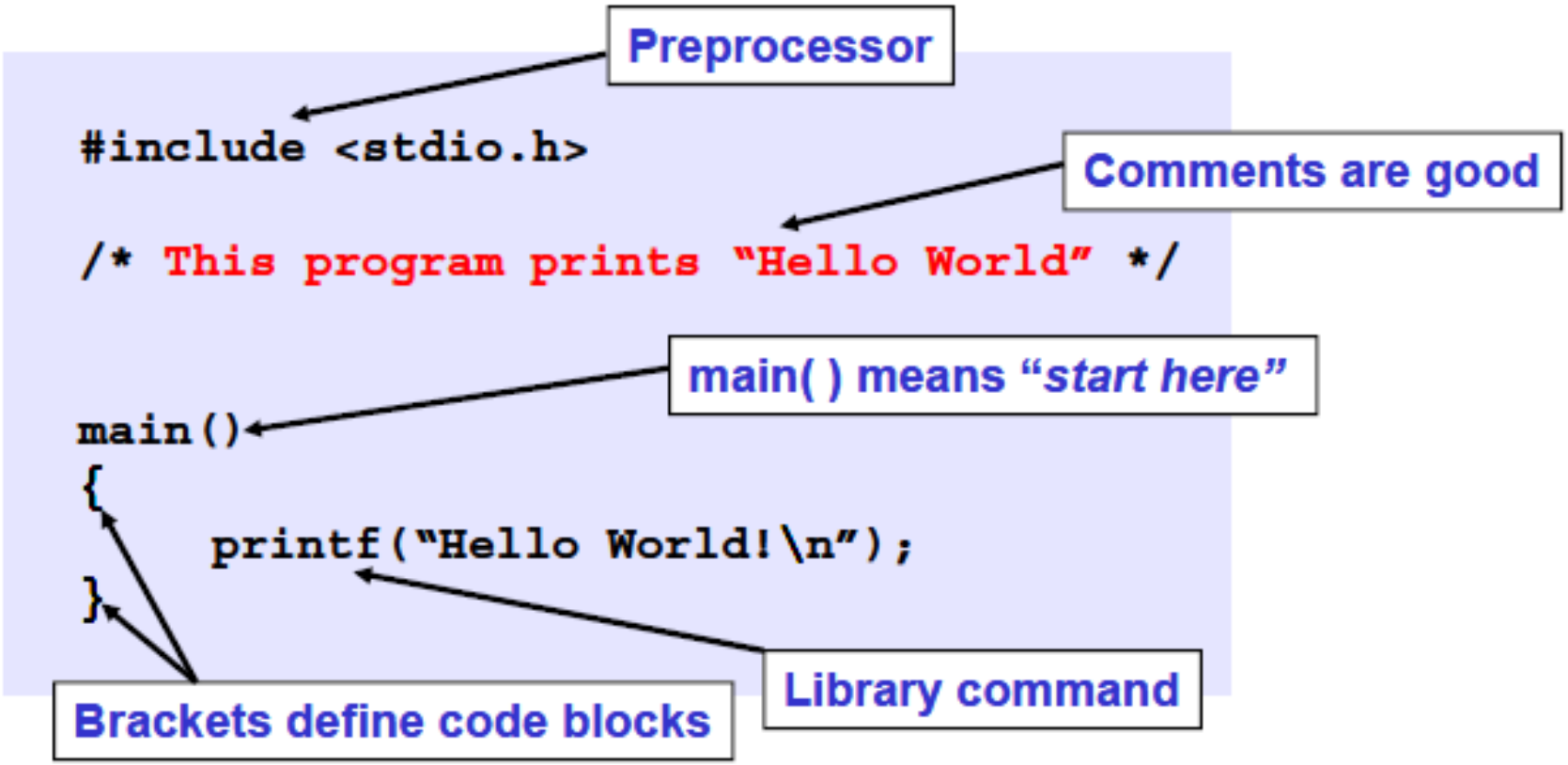
Why teach C?

- C is *small* (only 32 keywords).
- C is *common* (lots of C code about).
- C is *stable* (the language doesn't change much).
- C is *quick running*.
- C is the *basis for many other languages* (Java, C++, awk, Perl).
- It may not feel like it but C is one of the easiest languages to learn.

Some programmer jargon

- **Some words that will be used a lot:**
  - **Source code:** The stuff you type into the computer. The program you are writing.
  - **Compile (build):** Taking source code and making a program that the computer can understand.
  - **Executable:** The compiled program that the computer can run.
  - **Language:** The core part of C central to writing C code.
  - **Library:** Added functions for C programming which are bolted on to do certain tasks.
  - **Header file:** Files ending in .h which are included at the start of source code.

Our First C Program: *Hello World*



Keywords of C

- Flow control (6) – `if, else, return, switch, case, default`
- Loops (5) – `for, do, while, break, continue`
- Common *types* (5) – `int, float, double, char, void`
- *structures* (3) – `struct, typedef, union`
- Counting and sizing things (2) – `enum, sizeof`
- Rare but still useful *types* (7) – `extern, signed, unsigned, long, short, static, const`
- Evil keywords which we avoid (1) – `goto`
- Wierdies (3) – `auto, register, volatile`

The C Character Set

- **The C language alphabet:**
  - Uppercase letters ‘A’ to ‘Z’
  - Lowercase letters ‘a’ to ‘z’
  - Digits ‘0’ to ‘9’
  - Certain special characters:

!	#	%	^	&	*	(	)
-	_	+	=	~	[	]	\
|	;	:	‘	“	{	}	,
.	<	>	/	?	blank		

Some simple operations for variables

- In addition to `+`, `-`, `*` and `/` we can also use

  `+=`, `-=`, `*=`, `/=`, `--` and `%` (modulo)

`n++`	*increment n*	
`n--`	*decrement n*	
`a+=5`	*is equivalent to*	`a = a+5;`
`a-=5`	*is equivalent to*	`a = a-5;`
`a*=5`	*is equivalent to*	`a = a*5;`
`a/=5`	*is equivalent to*	`a = a/5;`

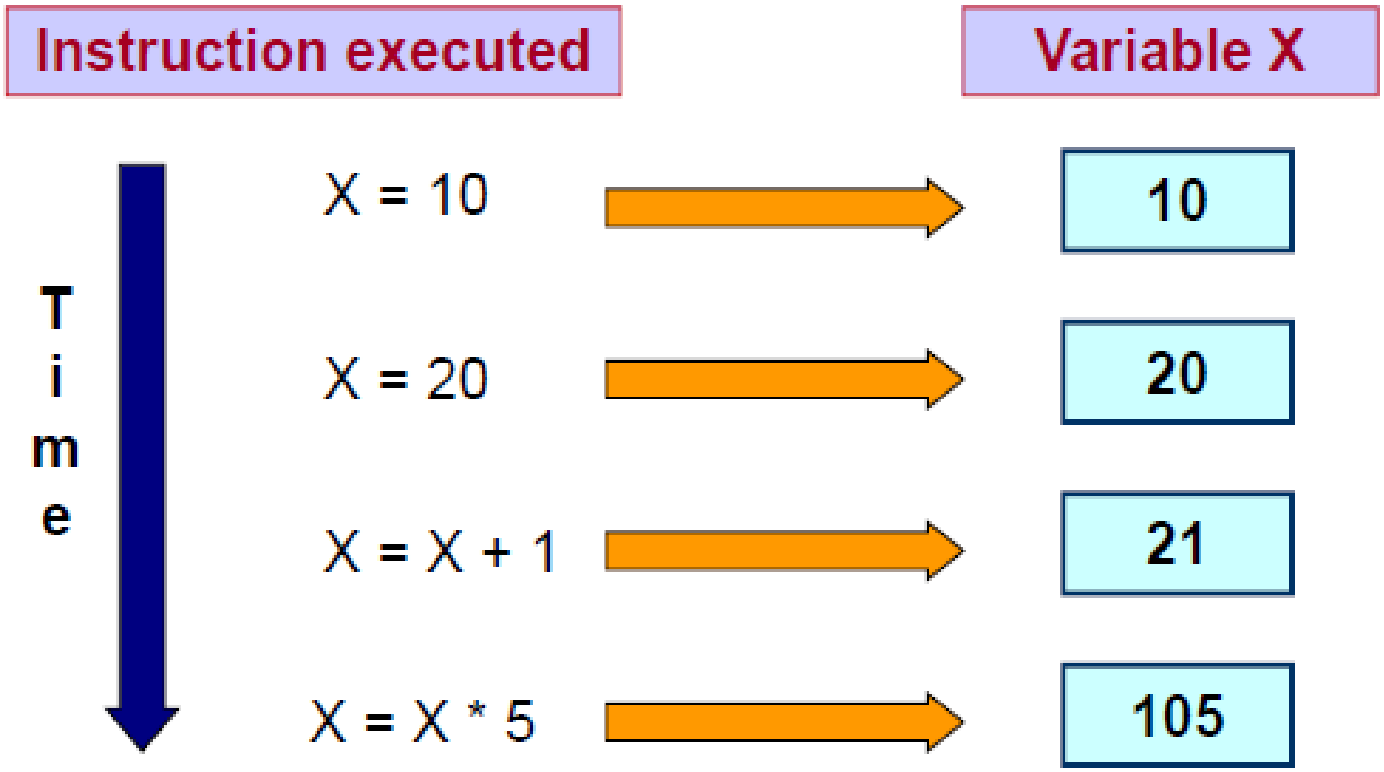
`(x % y)` gives the remainder when `x` is divided by `y`

Some Terminologies

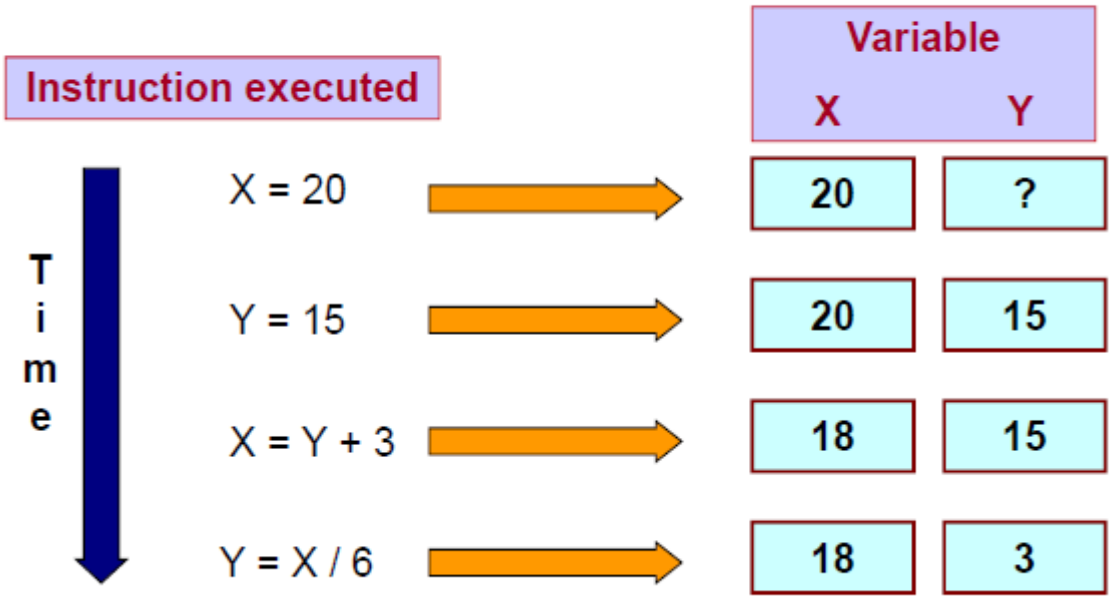
- **Algorithm / Flowchart**
  - A step-by-step procedure for solving a particular problem.
  - Independent of the programming language.

- **Program**
  - A translation of the algorithm/flowchart into a form that can be processed by a computer.
  - Typically written in a high-level language like C, C++, Java, etc.

Variables in Memory



Variables in Memory (contd.)



Data Types

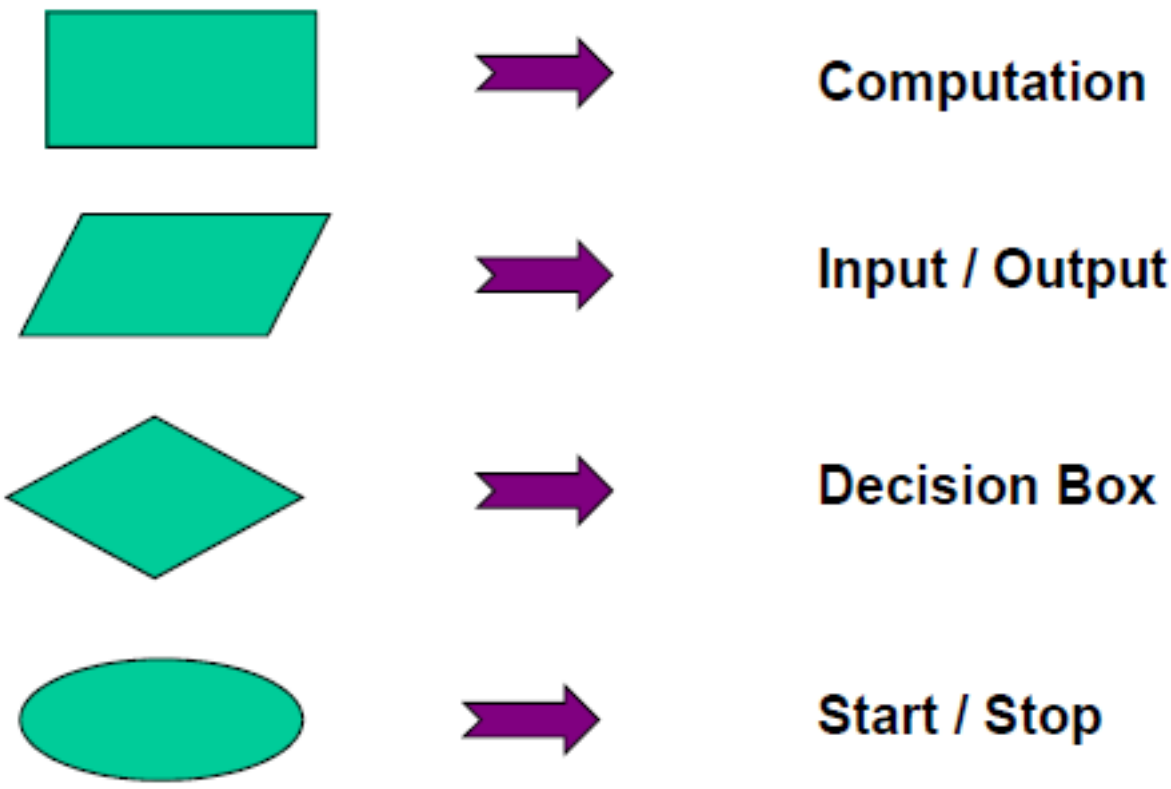
- Three common data types used:
  - Integer :: can store only whole numbers
    - Examples: 25, -56, 1, 0
  - Floating-point :: can store numbers with fractional values.
    - Examples: 3.14159, 5.0, -12345.345
  - Character :: can store a character
    - Examples: 'A', 'a', '*', '3', ' ', '+'

Data Types (contd.)

- How are they stored in memory?
  - Integer ::
    - 16 bits
    - 32 bits
  - Float ::
    - 32 bits
    - 64 bits
  - Char ::
    - 8 bits (ASCII code)
    - 16 bits (UNICODE, used in Java)

Actual number of bits vary from one computer to another

Flowchart: basic symbols

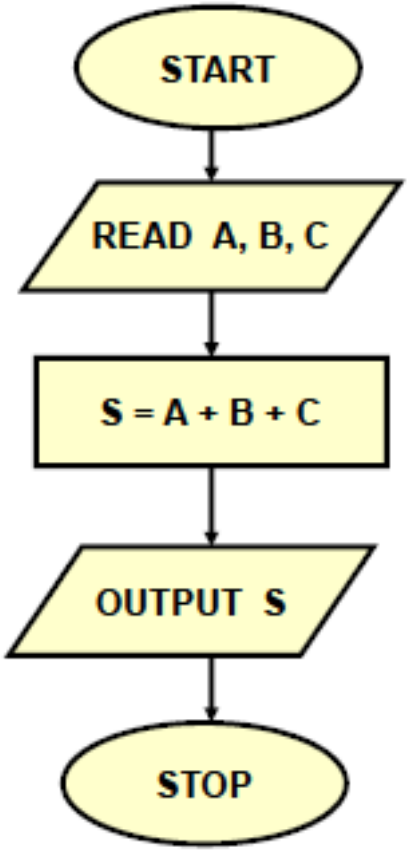


Contd.

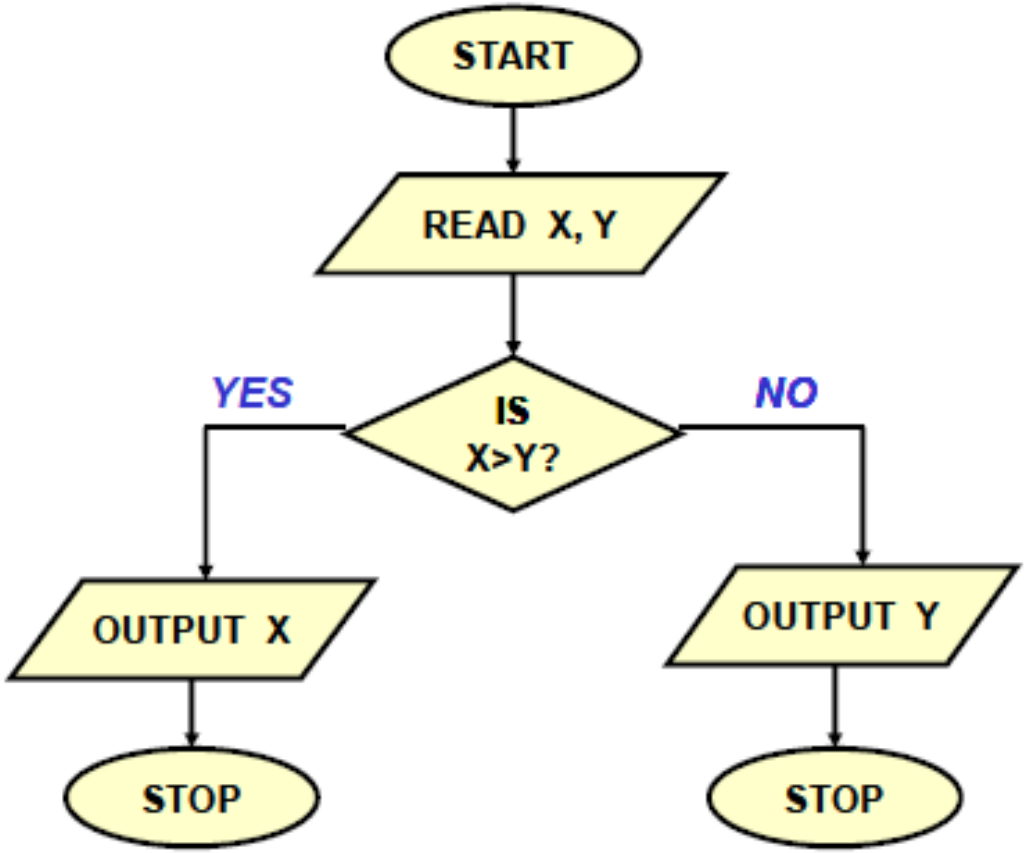
Flow of control

Connector

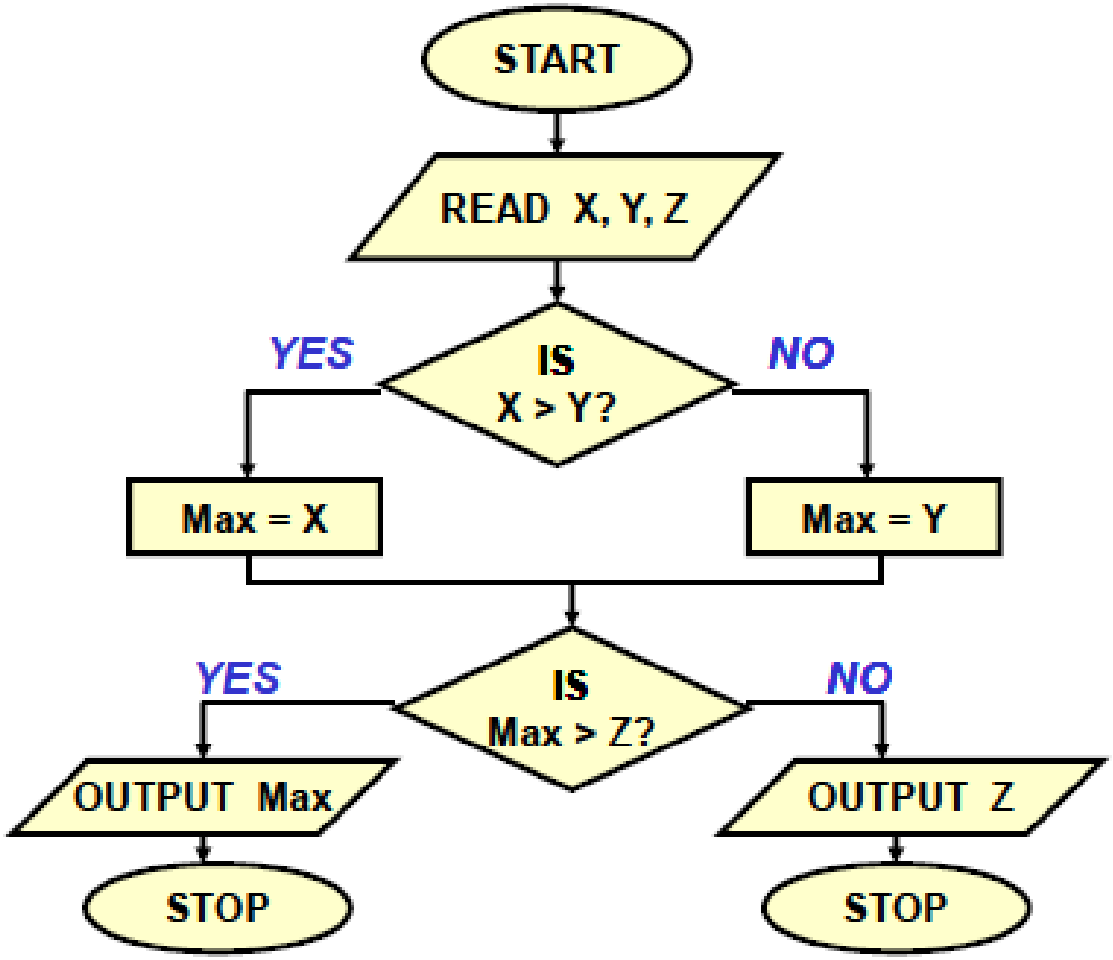
Example 1: *Adding three numbers*



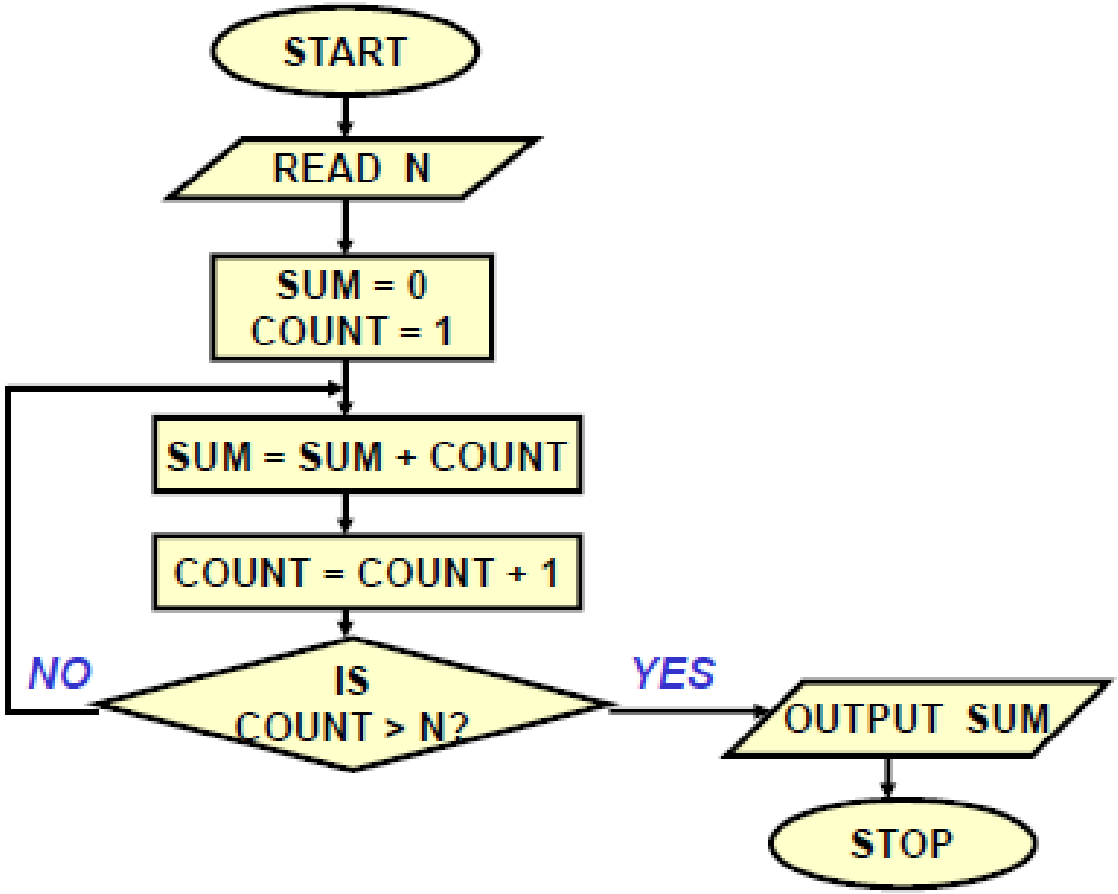
Example 2: *Larger of two numbers*



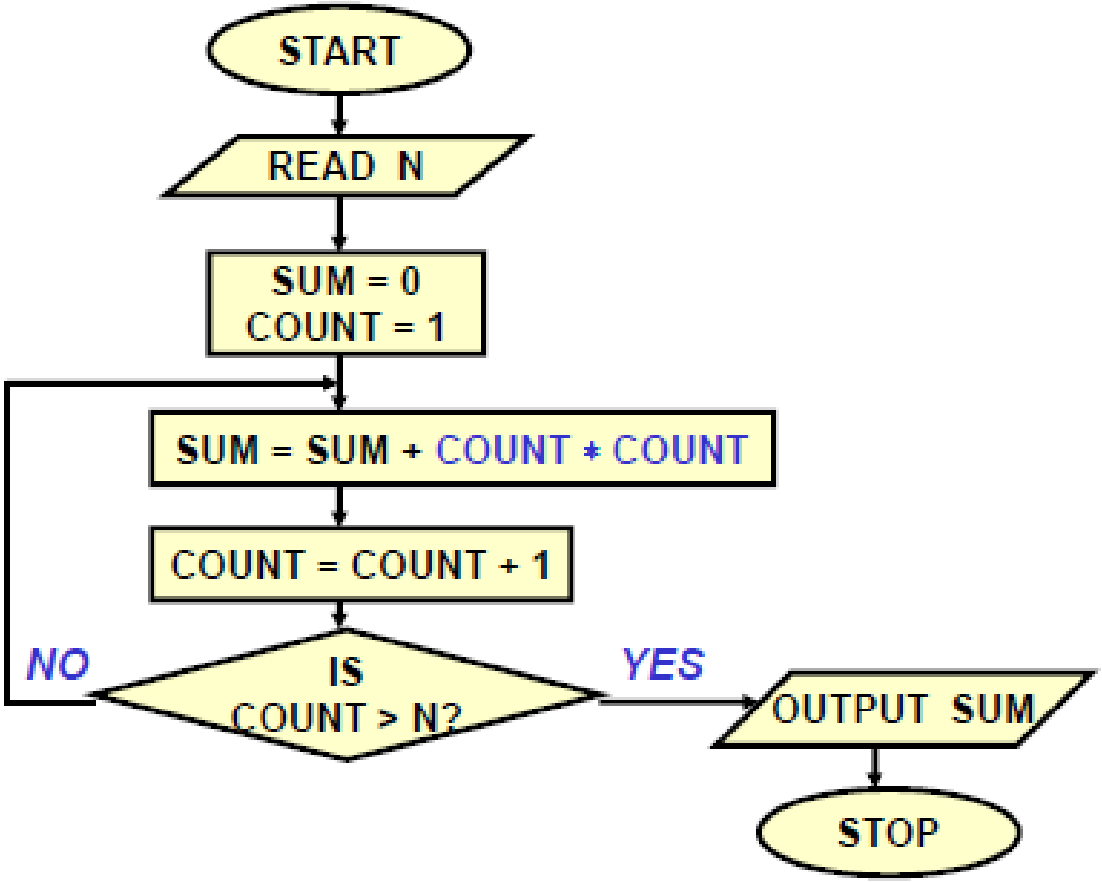
Example 3: *Largest of three numbers*



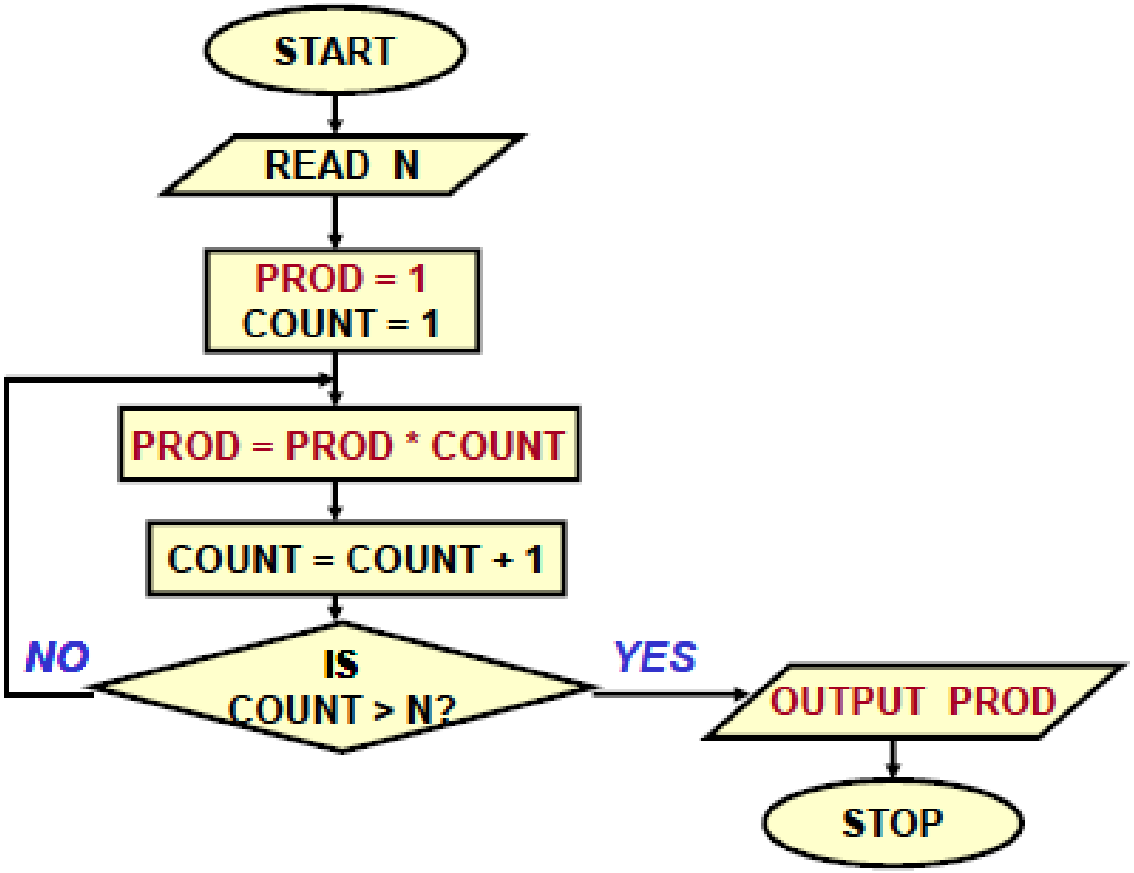
Example 4: *Sum of first N natural numbers*



Example 5: $SUM = 1^2 + 2^2 + 3^2 + N^2$



Example 7: *Computing Factorial*



Roots of a quadratic equation

$$ax^2 + bx + c = 0$$

TRY YOURSELF