# Computer Science I

## Chapter I - Introduction to Computers and Programming

Hanoi University of Science and Technology

October 8, 2025

Slide contents are mainly based on the following documents:

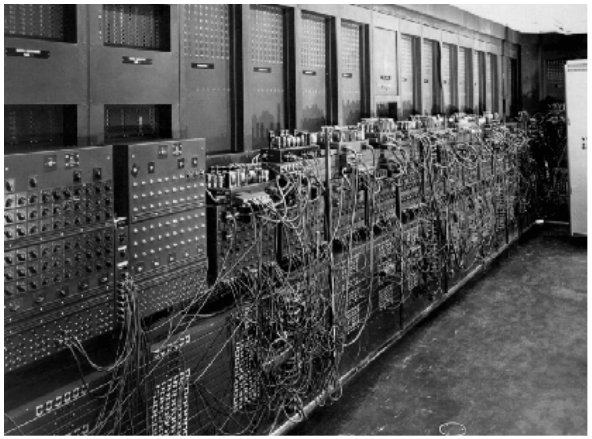
- Tony Gaddis, Starting out with C++, Six Edition, Pearson, 2009
- Gaddis Power Points Slides:
  `https://www.umsl.edu/~antognolij/gaddis.html`

# Overview

# Overview

# Why program?

History of computer development.



**1940s - 1960s**

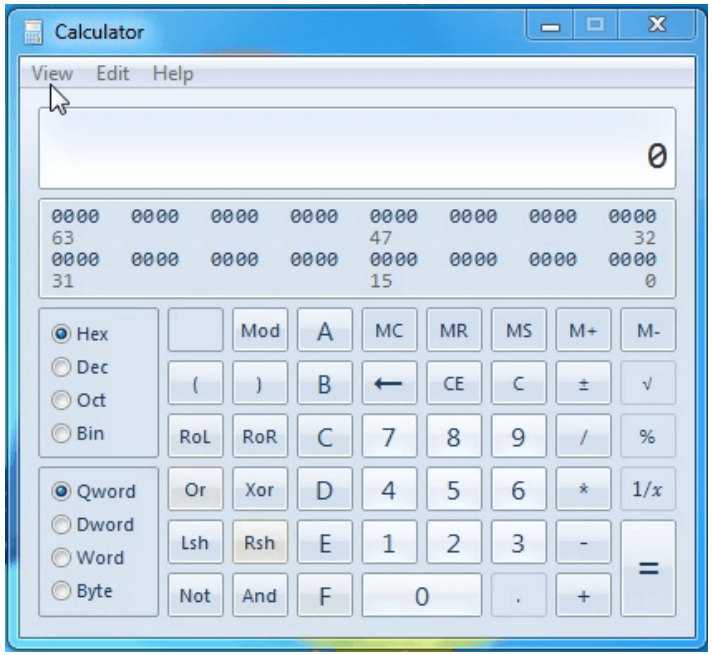
**1970s - 1990s**

**today**

**Computer**: *programmable machine designed to follow instructions*

# Why program?

**Program** *(or software): a set of instructions that a computer follows to perform a task*




An example of a computer program.

# Why program?

**Programmer** *(or software developer): a person who writes instructions to make a computer perform a task*

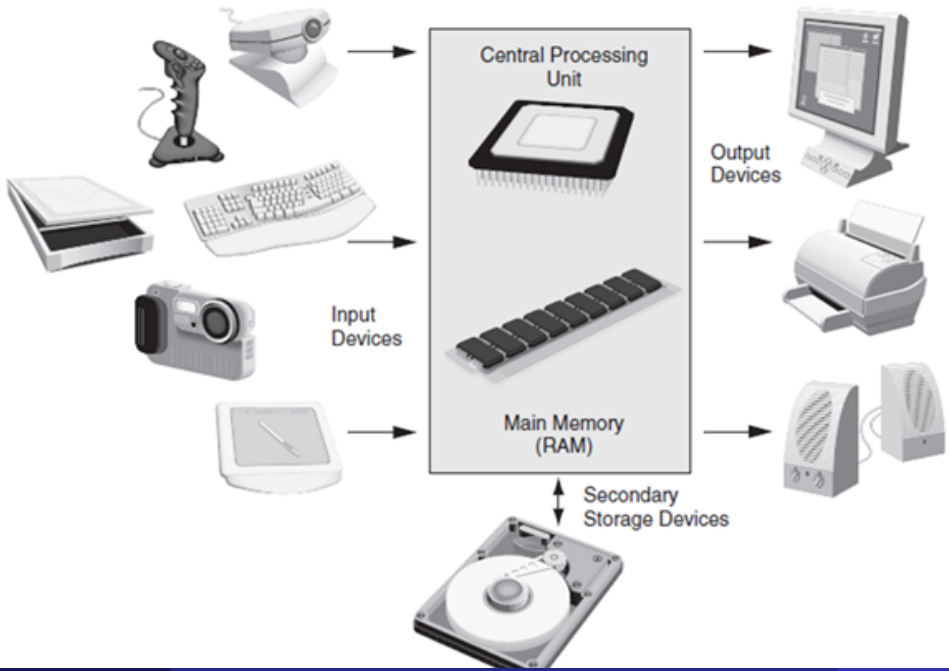
Computers can do many different tasks because they are programmable
But, without programmers, no programs
Without programs, the computer cannot do anything.

# Overview

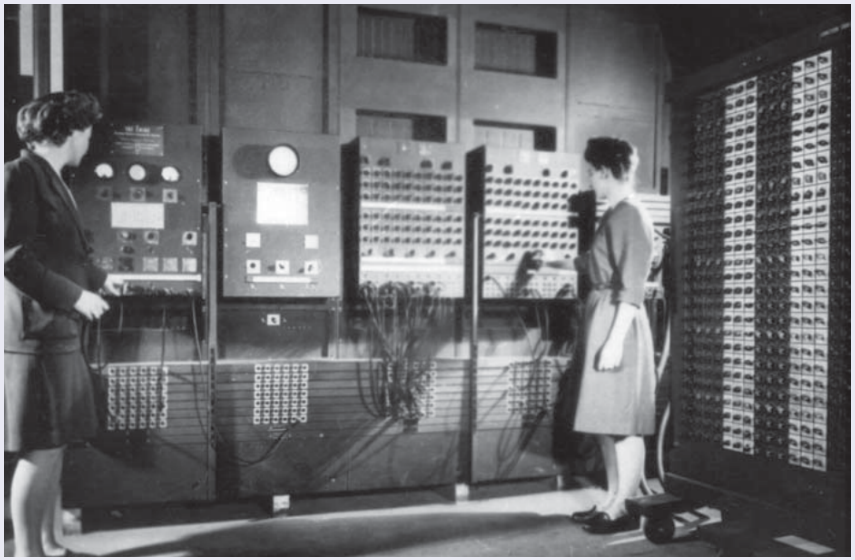
# Main Hardware Component Categories

*Hardware: physical components that a computer is made of*

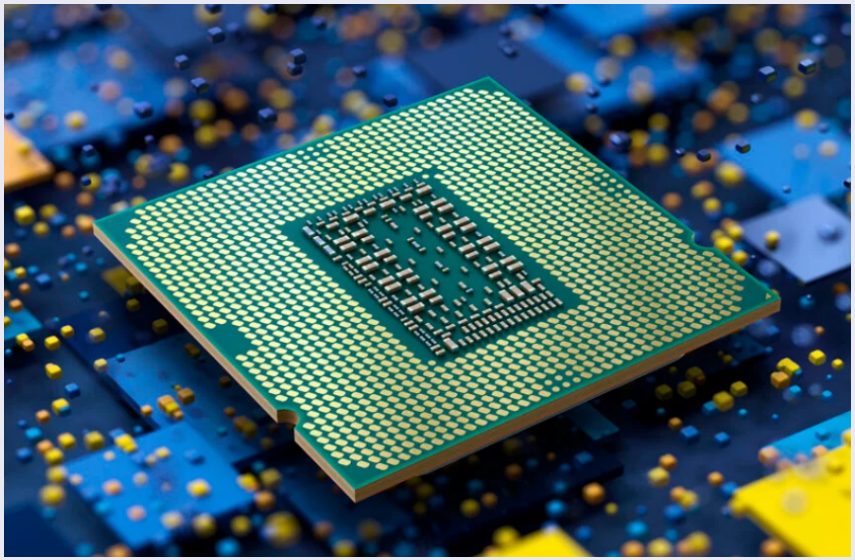
- *Central Processing Unit (CPU)*
- *Main Memory*
- *Secondary Memory / Storage*
- *Input Devices*
- *Output Devices*



# Central Processing Unit (CPU)



(a) First CPU

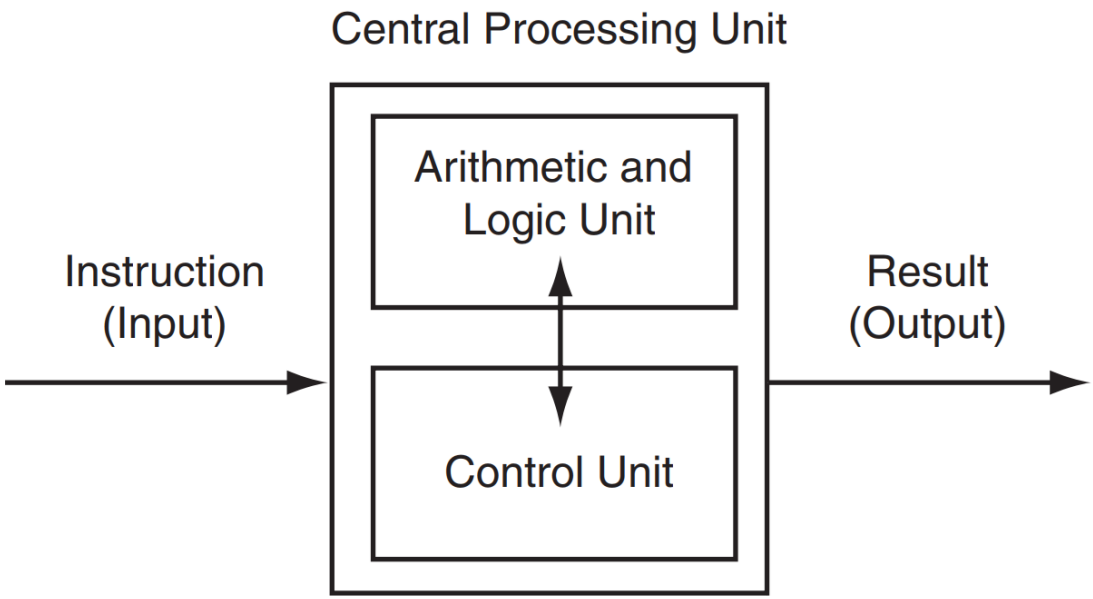


(b) Present

# 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*

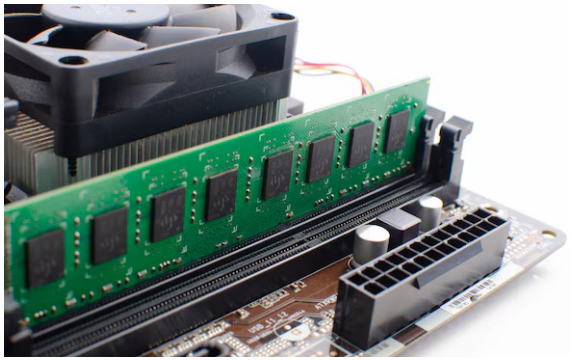



Organization of CPU

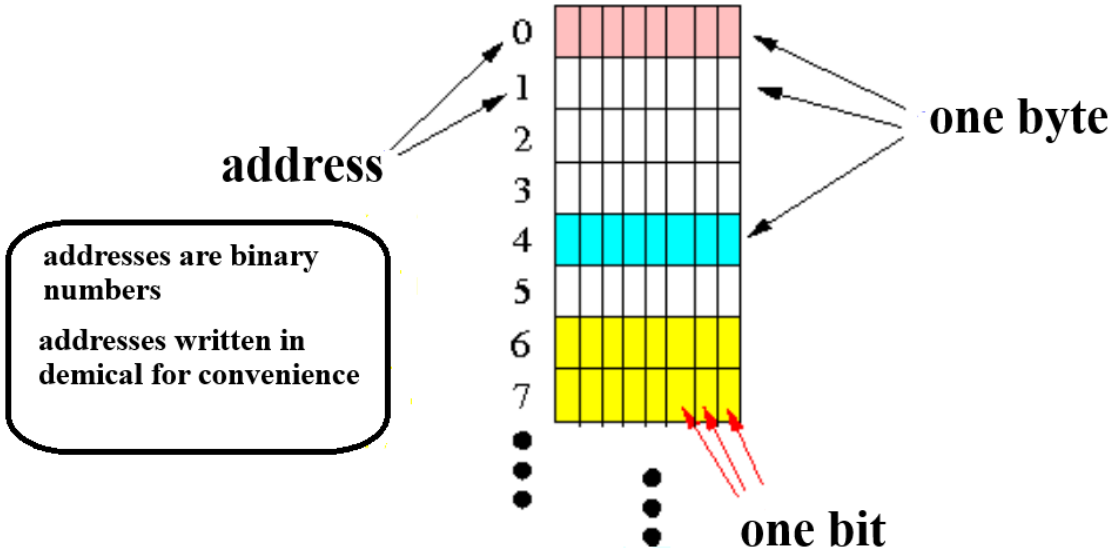
# Main Memory

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

# Central Processing Unit (CPU)



(a) RAM




(b) Address, bit, byte

| Unit | Notation | |
|---|---|---|
| byte | b | 8 bits |
| kilobyte | KB | $2^{10} = 1024$ bytes |
| megabyte | MB | 1024 KB |
| gigabyte | GB | 1024 MB |
| terabyte | TB | 1024 GB |

# 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: 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

- *Output – information the computer sends to the outside world*
- *Output device format and present it*
- *Some output devices:*
  *– Monitor, printer, speaker, ..*
  *– Disk drives, CD drives, and DVD drives can also be considered output devices*

# Software-Programs

*Categories of software:*

- *System software: programs that manage the computer hardware and the programs that run on it.*
  *- Operating system: Windows (Windows 10, 11), MacOS (Apple computers),.. - Utility programs: Windows defender, Task manager, Activity Monitor - Software development tools: Eclipse, Visual Studio, Notepad, ...*
- *Application software: programs that provide services to the user. Examples: Office 365, games, programs to solve specific problems*

# Overview



# Programs and Programming Languages

- *Program: a set of instructions that the computer follows to perform a task*

1. Display a message on the screen asking "How many hours did you work?"
2. Wait for the user to enter the number of hours worked. Once the user enters a number, store it in memory.
3. Display a message on the screen asking "How much do you get paid per hour?"
4. Wait for the user to enter an hourly pay rate. Once the user enters a number, store it in memory.
5. Multiply the number of hours by the amount paid per hour, and store the result in memory.
6. Display a message on the screen that tells the amount of money earned. The message must include the result of the calculation performed in Step 5.

- *Algorithm: a set of well-defined steps for performing a task or solving a problem*

# Machine Language

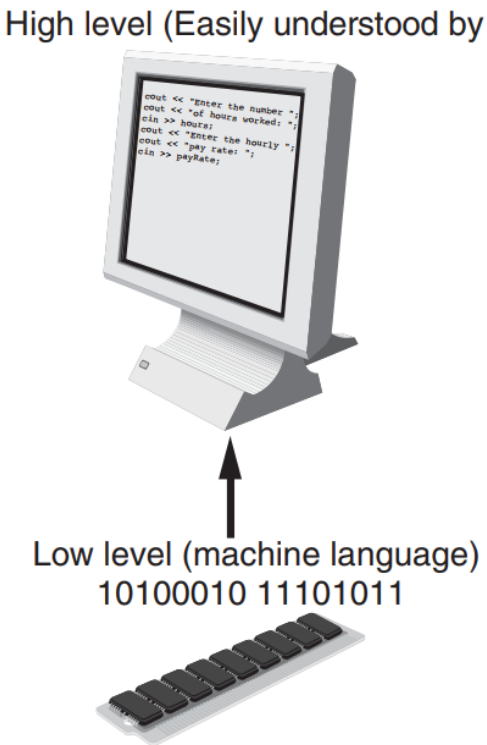
*But computer can only understand machine language:*

- *Although the previous algorithm defines the steps for calculating the gross pay, it is not ready to be executed on the computer.*
- *The computer only executes machine language instructions*
- *Machine language instructions are binary numbers, such as 1011010000000101*
- *Rather than writing programs in machine language, programmers use programming languages.*
- *A programming language: a special language used to write computer programs*

# Programs and Programming Languages

*Types of languages:*

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



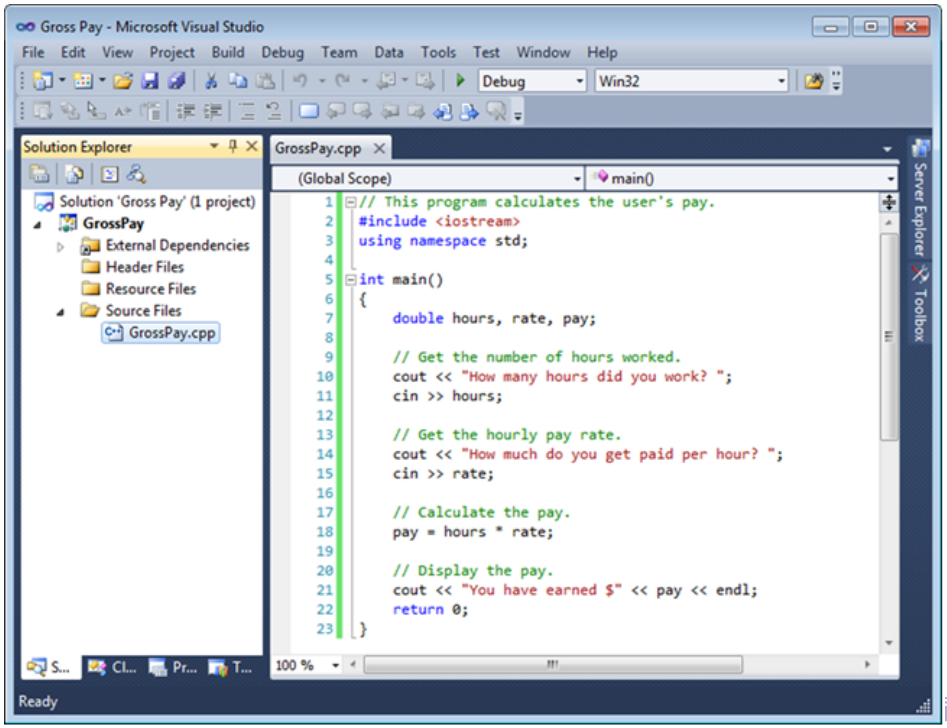
# Some Well-Known Programming Languages

# Some Well-Known Programming Languages

| Language | Description |
|---|---|
| *C* | *A structured, general-purpose language developed at Bell Laboratories. C offers both high-level and low-level features.* |
| *C++* | *Based on the C language, C++ offers object-oriented features not found in C. Also invented at Bell Laboratories.* |
| *C#* | *Pronounced "C sharp." A language invented by Microsoft for developing applications based on the Microsoft .NET platform.* |
| *Java* | *An object-oriented language invented at Sun Microsystems. Java may be used to develop programs that run over the Internet, in a Web browser.* |
| *JavaScript* | *JavaScript can be used to write small programs that run in Web pages. Despite its name, JavaScript is not related to Java.* |
| *Python* | *Python is a general-purpose language created in the early 1990s. It has become popular in both business and academic applications.* |

# Integrated Development Environments (IDEs)

- *A programming language should run in an environment*
  - *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.*



# Overview



# What is a Program Made of?

*Common elements in programming languages:*

- *Key Words*
- *Programmer-Defined Identifiers*
- *Operators*
- *Punctuation*
- *Syntax*

# What is a Program Made of?

## Program: Gross Pay

```
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 }
```

# Key Words

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

```
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 Gross Pay:* `hours`, `rate`, *and* `pay`

# Operators

- *Used to perform operations on data*
- *Many types of operators:*
  - *Arithmetic - ex: +,-,*,/*
  - *Assignment – ex: =*
- *Some operators in the Gross Pay: << >> = **

```
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;
```

# Punctuation

- *Characters that mark the end of a statement, or that separate items in a list*
- *In Gross Pay: , and ;*

```
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*

# Variable

- *A variable is a named storage location in the computer's memory for holding a piece of data.*
- *In above program, 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*
- *To create a variable in a program you must write a variable definition (also called a variable declaration)*

# Variable

- *Here is the statement from the above program that defines the variables:*
  `double hours, rate, pay;`
- *There are many different types of data, which you will learn about in this course.*
- *A variable holds a specific type of data.*
- *The variable definition specifies the type of data a variable can hold, and the variable name.*
- *The word* `double` *specifies that the variables can hold double-precision floating point numbers. (You will learn more about that in Chapter 2)*

# Overview

# Input, Processing, and Output

*Three activities of a program:*

1. *Gather input data:*
   - *From keyboard*
   - *From files on disk drives*
2. *Process the input data*
3. *Display the results as output:*
   - *Send it to the screen*
   - *Write to a file*

# Overview

# 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.

# The Programming Process

**1. Clearly define what the program is to do**

- *Purpose: To calculate the user's gross pay*
- *Input: Number of hours worked, hourly pay rate*
- *Process: Multiply number of hours worked by hourly pay rate. The result is the user's gross pay*
- *Output Display a message indicating the user's gross pay*

# The Programming Process

**2. Visualize the program running on the computer**

```
How many hours did you work? 10
How much do you get paid per hour? 15
You have earned $150
```

# The Programming Process

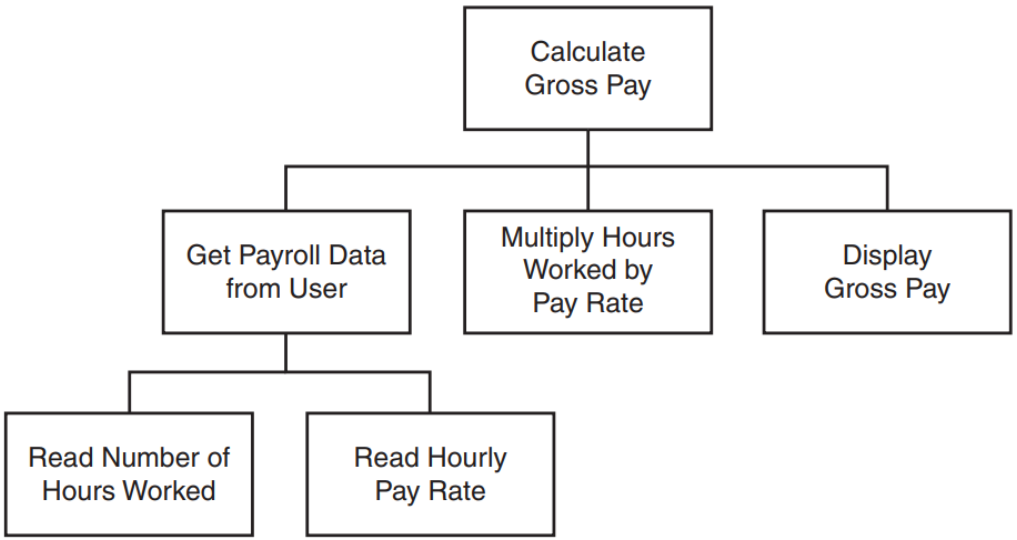
**3. Use design tools such as a hierarchy chart, flowcharts, or pseudocode to create a model/algorithm of the program.**

- *An algorithm must be specific enough so that it can be conveniently translated into a computer program (using C++, for example)*
- *An algorithm can be specified:*
  - *Textually: For example, using pseudo code (see later)*
  - *Graphically: using flowcharts, hierarchy chart (see later)*

# Hierarchy chart

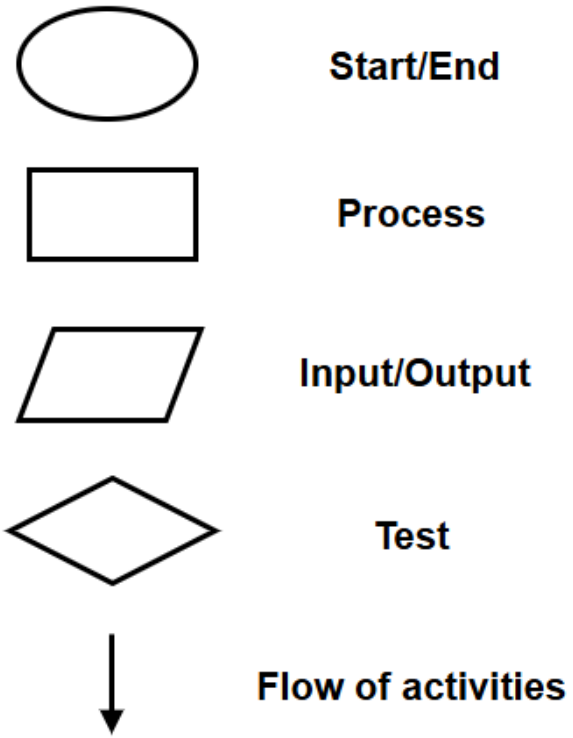
*Diagram that graphically depicts the structure of a program.*

- *Boxes representing each step in the program.*
- *Connected in a way that illustrates their relationship to one another*

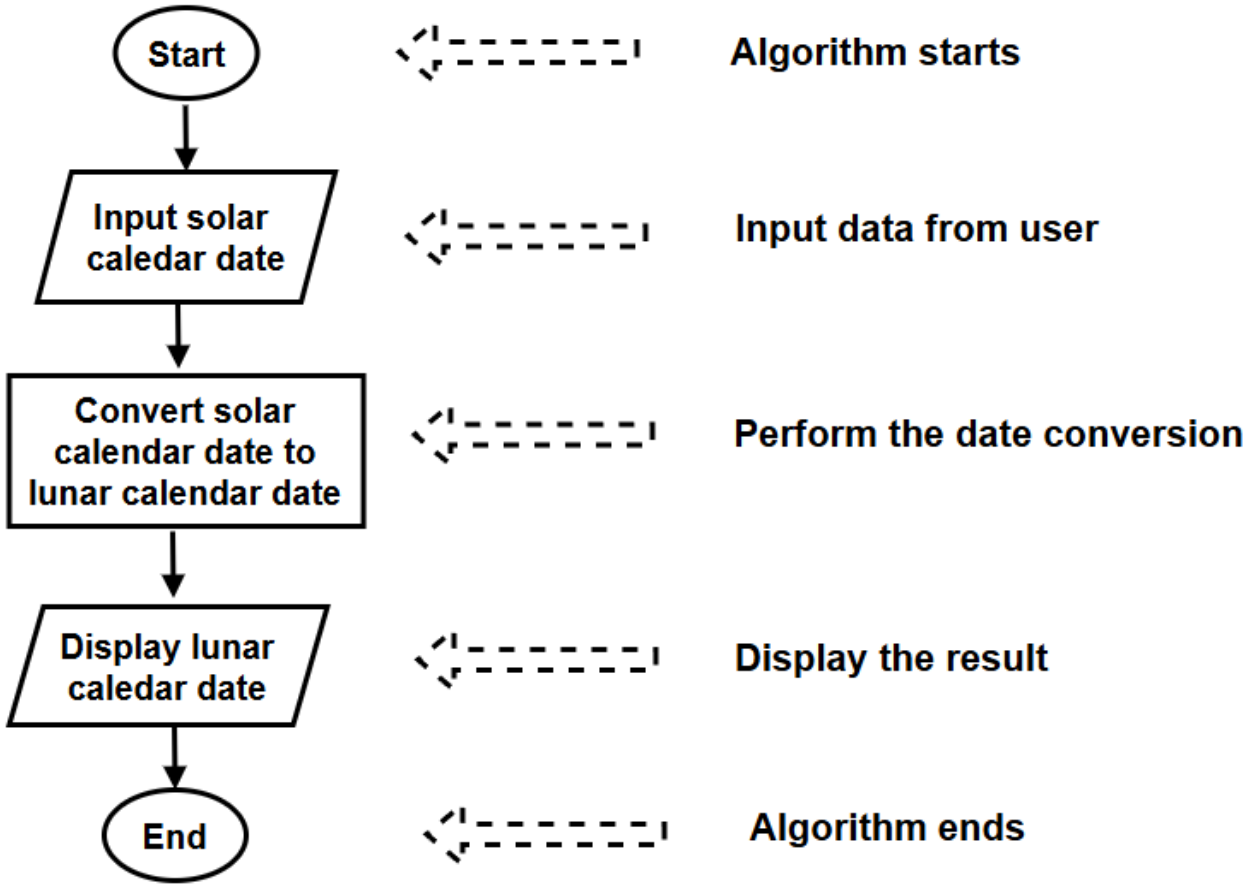


# Flowcharts

- *A flowchart is a graphical representation of the sequence of operations in a program.*
- *An algorithm can be represented graphically using a flowchart*



# Flowcharts


Start
Input solar caledar date
Convert solar calendar date to lunar calendar date
Display lunar caledar date
End
Algorithm starts
Input data from user
Perform the date conversion
Display the result
Algorithm ends

# Pseudocode

- An outline of a program, written in a form that can easily be converted into real programming statements. It resembles the actual program that will be implemented later. However, it cannot be compiled nor executed.
- Pseudocode normally codes the following actions:
  - Initialization of variables
  - Assignment of values to the variables
  - Arithmetic operations
  - Relational operations

# Example of Pseudocode

1. Start
2. Read `quantity`
3. Read `price_per_kg`
4. `price` = `quantity` * `price_per_kg`
5. Print `price`
6. End

# The Programming Process

**4. Check the model for logical errors**

- *Logical errors are errors which are related to program logic*
- *Normally, logical errors are not detectable by the compiler*
- *Logical errors are usually detected during program runtime. For example, a program producing unexpected results is an indication that it has logical errors.*
- *It is important to remember that if the compiler does not produce any error messages, it does not mean that your program is free of logical errors.*

# The Programming Process

- Possible to remove all syntax errors in a program and still not have it run
- Even if it runs, it may still not do what you meant it to do
- For example:

  `2 + 3 * 5` and `(2 + 3) * 5`

  are both syntactically correct expressions, but have different meanings

# The Programming Process

**5. Type the code, save it, and compile it**

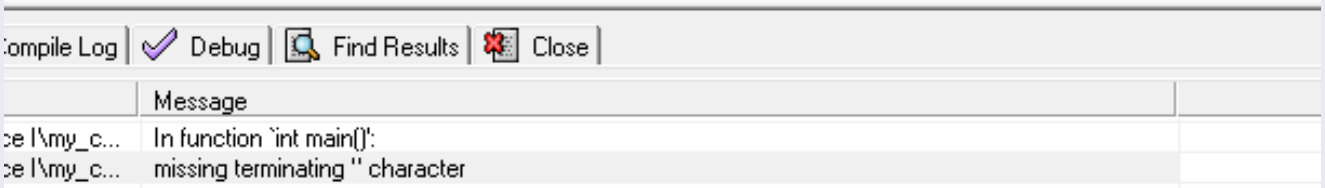
- *Syntax errors are errors in the source code which are related to the syntax of the language*
- *Syntax errors are detected by the compiler. An executable file will be generated by the compiler only if the source code it compiles has no syntax errors*
- *Syntax errors are reported by the compiler in the form of error messages*

# The Programming Process

```cpp
#include <iostream>
#include <conio.h>  // Only needed if you're using getch()

int main() {
    std::cout << "Hello World!

    getch();  // Waits for a key press before closing the console
    return 0;
}
```



# The Programming Process

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

# Exercise

Write an algorithm to find $x$ (real number) such that $ax^2 + bx + c = 0$ where $a, b, c$ are input parameters.

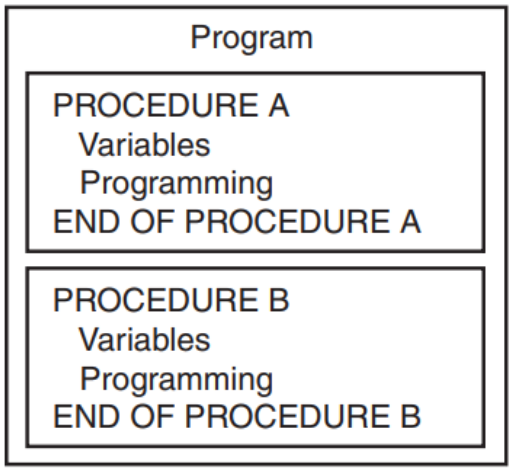
# Overview



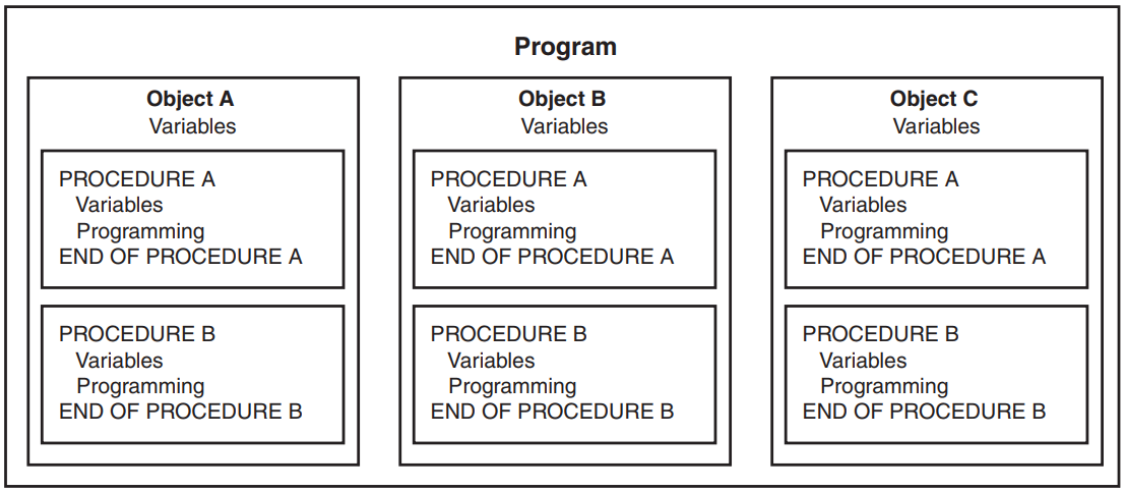
# Procedural and Object-Oriented Programming

*Two ways of program design:*

- *Procedural programming: focus is on the process. Procedures/functions are written to process data.*
- *Object-Oriented programming: focus is on objects, which contain data and the means to manipulate the data. Messages sent to objects to perform operations.*
- *C++ can do both.*




(a) Procedural




(b) Object-oriented

# First week

- Checkpoint: 1.7, 1.13, 1.16, 1.20
- Algorithm workbench: 1.31
- Predict the Result: 1.33, 1.34, 1.35