



**In this lecture you will learn:**

- **Programming Language.**
- **Algorithms**

## **Programming language**

The most basic language of a computer, the machine language, provides program instructions in bits. Even though most computers perform the same kinds of operations, the designers of the computer may have chosen different sets of binary codes to perform the operations. Therefore, the machine language of one machine is not necessarily the same as the machine language of another machine. The only consistency among computers is that in any modern computer, all data is stored and manipulated as binary codes.

suppose you want to use the equation:

$$\mathbf{Wages = rate * hours}$$

To calculate weekly wages. Further, suppose that the binary code **100100** stands for load, **100110** stands for multiplication, and **100010** stands for store. In machine language, you might need the following sequence of instructions to calculate weekly wages:

**100100 010001**

**100110 010010**

**100010 010011**

Assembly languages were developed to make the programmer's job easier. Table below shows some examples of instructions in assembly language and their corresponding machine language code.

Assembly Language	Machine Language
LOAD	100100
STOR	100010
MULT	100110
ADD	100101
SUB	100011

Using assembly language instructions, you can write the equation to calculate the weekly wages as follows:

**LOAD rate**

**MULT hours**

**STOR wages**

As you can see, it is much easier to write instructions in assembly language. However, a computer cannot execute assembly language instructions directly. The instructions first have to be translated into machine language. A program called an assembler translates the assembly language instructions into machine language.

**Assembler:** A program that translates a program written in assembly language into an equivalent program in machine language.

The next step toward making programming easier was to devise high-level languages that were closer to natural languages. Basic, FORTRAN, COBOL, Pascal, C, C++, C#, and Java are all high-level languages. In C++, you write the weekly wages equation as follows:

**Wages = rate \* hours;**

To run on a computer, these C++ instructions first need to be translated into machine language. A program called a **compiler** translates instructions written in high-level languages into machine code.

**Compiler:** A program that translates instructions written in a high-level language into the equivalent machine language.

## Processing a C++ Program

In the previous sections, we discussed machine language and high-level languages and showed a C++ program. Because a computer can understand only machine language, you are ready to review the steps required to process a program written in C++. Consider the following C++ program:

```
#include <iostream>  
  
using namespace std;  
  
int main()  
  
{  
  
cout << "My first C++ program." << endl;  
  
return 0;
```

```
}
```

This will display the following line on the screen:

### **My first C++ program.**

The following steps are necessary to process a C++ program.

1. You use a text editor to create a C++ program following the rules, or syntax, of the high-level language. This program is called the source code, or source program. The program must be saved in a text file that has the extension `.cpp`.
2. The C++ program given in the preceding section contains the statement **#include <iostream>**. In a C++ program, statements that begin with the symbol `#` are called **preprocessor directives**. These statements are processed by a program called preprocessor.
3. After processing **preprocessor directives**, the next step is to verify that the program obeys the rules of the programming language—that is, the program is syntactically correct—and translate the program into the equivalent machine language. The compiler checks the source program for syntax errors and, if no error is found, translates the program into the equivalent machine language. The equivalent machine language program is called an object program.

**Object program:** The machine language version of the high-level language program.

4. The programs that you write in a high-level language are developed using an integrated development environment (IDE). The IDE contains many programs that are useful in creating your program. This

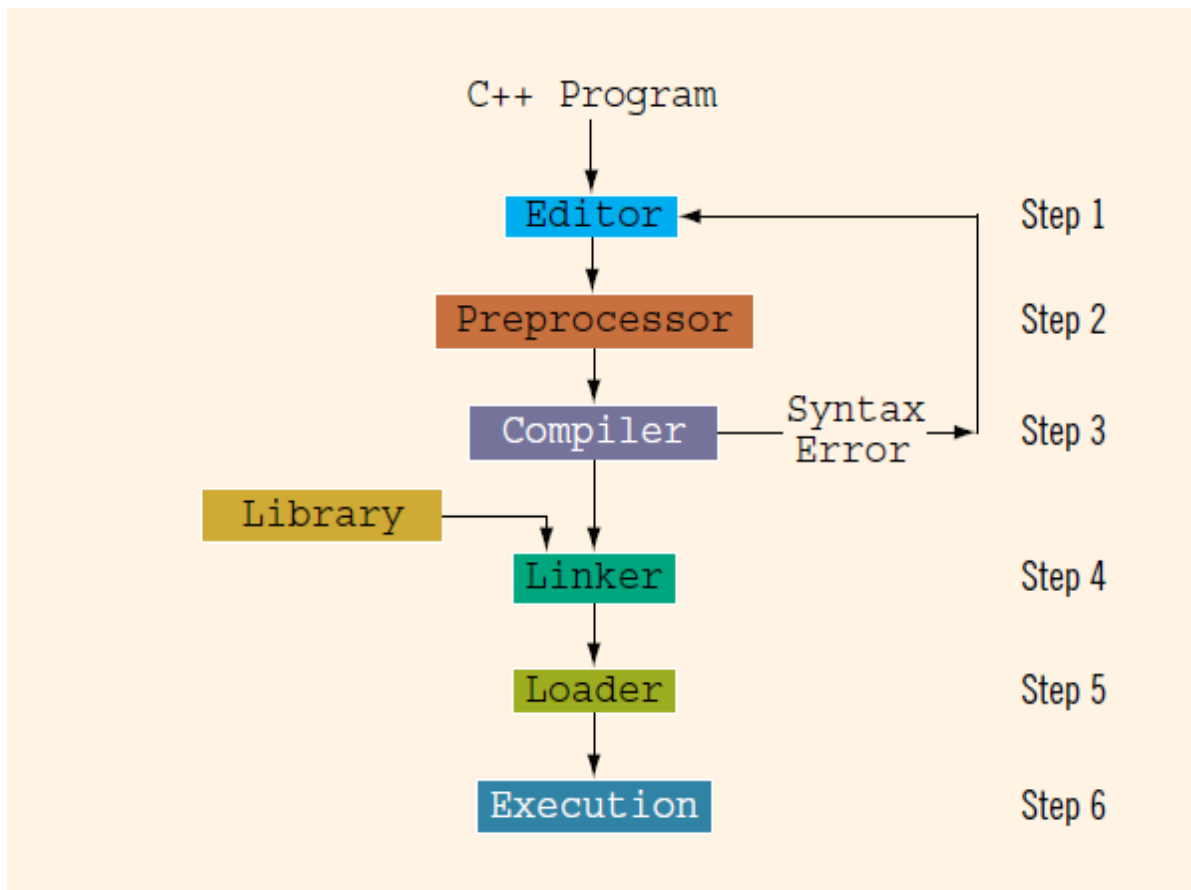
prewritten code (program) resides in a place called the library. A program called a linker combines the object program with the programs from libraries.

**Linker:** A program that combines the object program with other programs in the library and is used in the program to create the executable code.

5. You must next load the executable program into main memory for execution. A program called a loader accomplishes this task.

**Loader:** A program that loads an executable program into main memory.

6. The final step is to execute the program.



## **Programming with the Problem Analysis–Coding–Execution Cycle**

*Programming* is a process of problem solving. Different people use different techniques to solve problems. Some techniques are nicely outlined and easy to follow. They not only solve the problem, but also give insight into how the solution was reached. These problem-solving techniques can be easily modified if the domain of the problem changes.

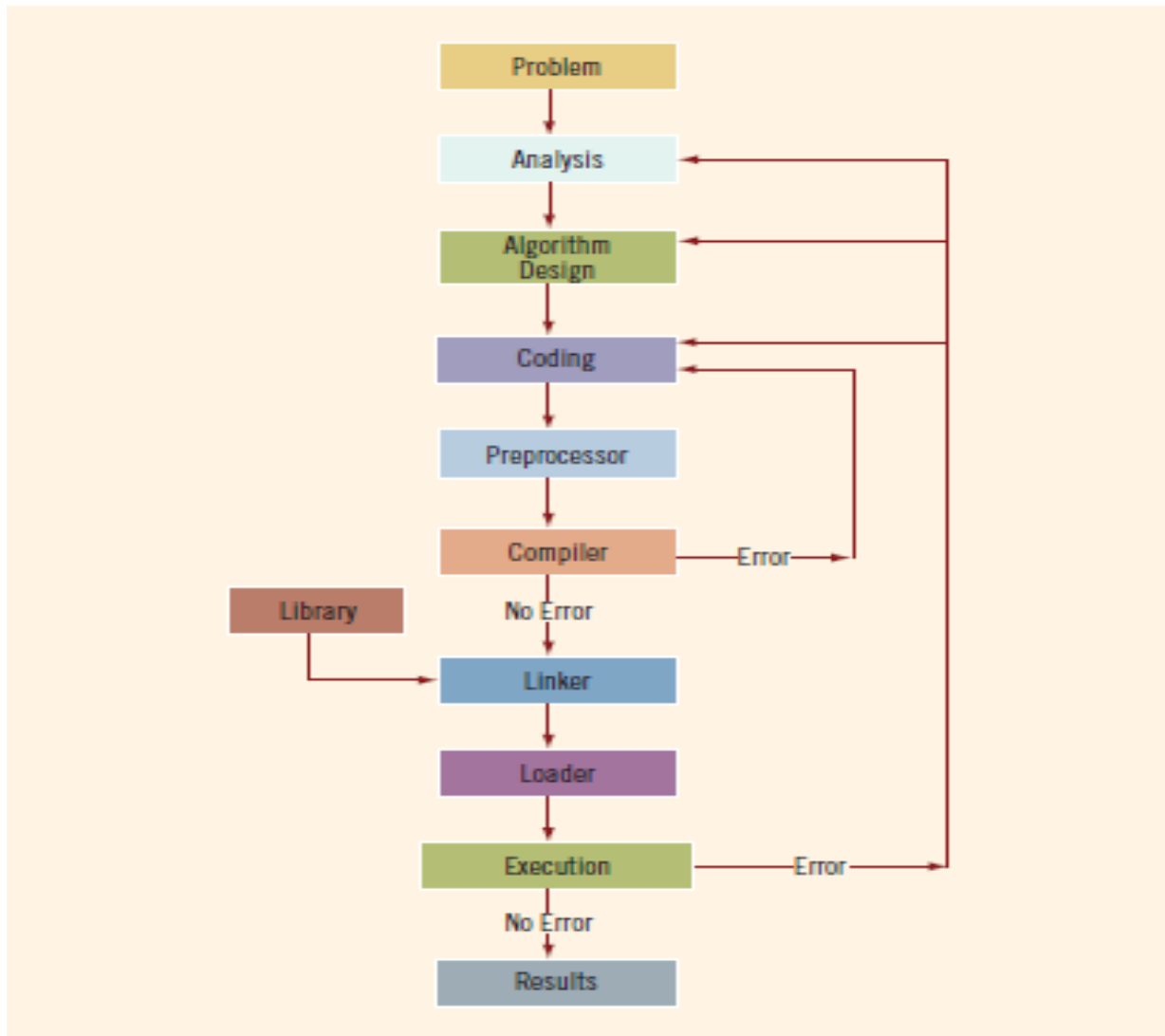
To be a good problem solver and a good programmer, you must follow good problem solving techniques. One common problem-solving technique includes *analyzing* a problem, *outlining* the problem requirements, and *designing* steps, called an algorithm, to solve the problem.

**Algorithm:** A step-by-step problem-solving process in which a solution is arrived at in a finite amount of time.

In a programming environment, the problem-solving process requires the following three steps:

1. Analyze the problem, outline the problem and its solution requirements, and design an algorithm to solve the problem.
2. Implement the algorithm in a programming language, such as C++, and verify that the algorithm works.

3. Maintain the program by using and modifying it if the problem domain changes.



### Example 1:

In this example, we design an algorithm to find the perimeter and area of a rectangle. To find the perimeter and area of a rectangle, you need to know the rectangle's length and width.

The perimeter and area of the rectangle are then given by the following formulas:

$$\text{Perimeter} = 2 * (\text{length} + \text{width})$$

$$\text{Area} = \text{length} * \text{width}$$

The algorithm to find the perimeter and area of the rectangle is:

1. Get the length of the rectangle.
2. Get the width of the rectangle.
3. Find the perimeter using the following equation:

$$\text{Perimeter} = 2 * (\text{length} + \text{width})$$

4. Find the area using the following equation:

$$\text{Area} = \text{length} * \text{width}$$

### **Example 2:**

In this example, we design an algorithm that calculates the sales tax and the price of an item sold in a particular state.

The sales tax is calculated as follows: The state's portion of the sales tax is 4%, and the city's portion of the sales tax is 1.5%. If the item is a luxury item, such as a car more than \$50,000, then there is a 10% luxury tax. To calculate the price of the item, we need to calculate the state's portion of the sales tax, the city's portion of the sales tax, and, if it is a luxury item, the luxury tax.



Suppose **salePrice** denotes the selling price of the item, **stateSalesTax** denotes the state's sales tax, **citySalesTax** denotes the city's sales tax, **luxuryTax** denotes the luxury tax, **salesTax** denotes the total sales tax, and **amountDue** denotes the final price of the item. To calculate the sales tax, we must know the selling price of the item and whether the item is a luxury item. The **stateSalesTax** and **citySalesTax** can be calculated using the following formulas:

$$\text{stateSalesTax} = \text{salePrice} * 0.04$$

$$\text{citySalesTax} = \text{salePrice} * 0.015$$

Next, you can determine **luxuryTax** as follows:

if (item is a luxury item)

$$\text{luxuryTax} = \text{salePrice} \_ 0.1$$

otherwise

$$\text{luxuryTax} = 0$$

Next, you can determine **salesTax** as follows:

$$\text{salesTax} = \text{stateSalesTax} + \text{citySalesTax} + \text{luxuryTax}$$

Finally, you can calculate **amountDue** as follows:

$$\text{amountDue} = \text{salePrice} + \text{salesTax}$$

The algorithm to determine **salesTax** and **amountDue** is, therefore:

1. Get the selling price of the item.

2. Determine whether the item is a luxury item.
3. Find the state's portion of the sales tax using the formula:

$$\text{stateSalesTax} = \text{salePrice} \_ 0.04$$

4. Find the city's portion of the sales tax using the formula:

$$\text{citySalesTax} = \text{salePrice} \_ 0.015$$

5. Find the luxury tax using the following formula:

if (item is a luxury item)

$$\text{luxuryTax} = \text{salePrice} \_ 0.1$$

otherwise

$$\text{luxuryTax} = 0$$

6. Find salesTax using the formula:

$$\text{salesTax} = \text{stateSalesTax} + \text{citySalesTax} + \text{luxuryTax}$$

7. Find amountDue using the formula:

$$\text{amountDue} = \text{salePrice} + \text{salesTax}$$

### References:

- [1] Deitel, P. & Deitel, R., "C++ How to Program", Eighth Edition, Pearson Education Inc., 2012.
- [2] Stefan B., " C++ Primer Plus ", Sixth Edition, Pearson Education Inc., 2012.