Object-oriented Programming

YoungWoon Cha
CSE Department
Spring 2023

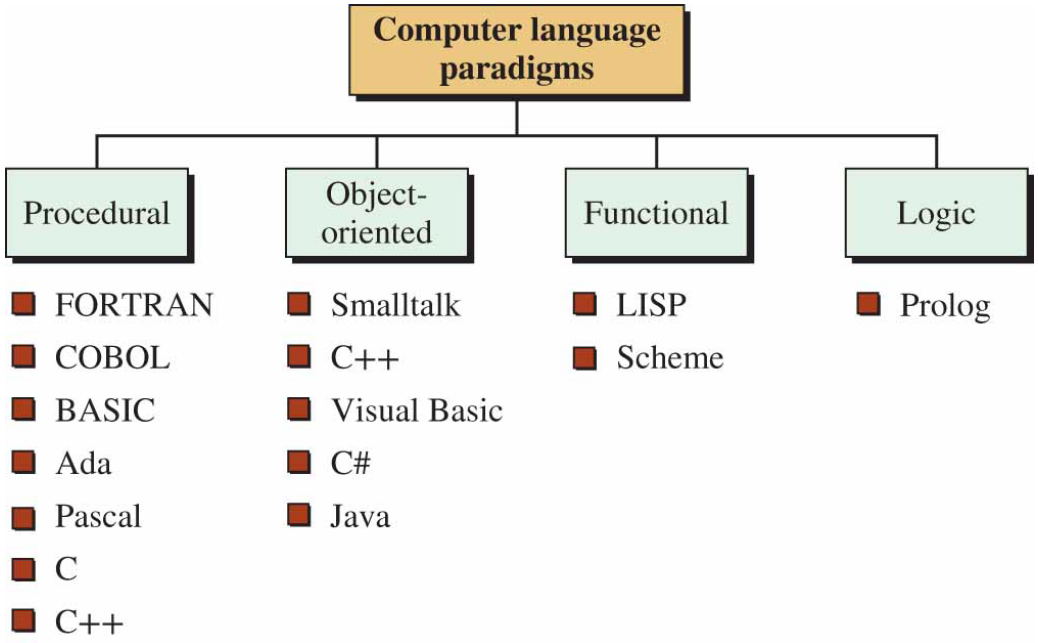
KU KONKUK UNIVERSITY

OOP Overview

LANGUAGE PARADIGMS

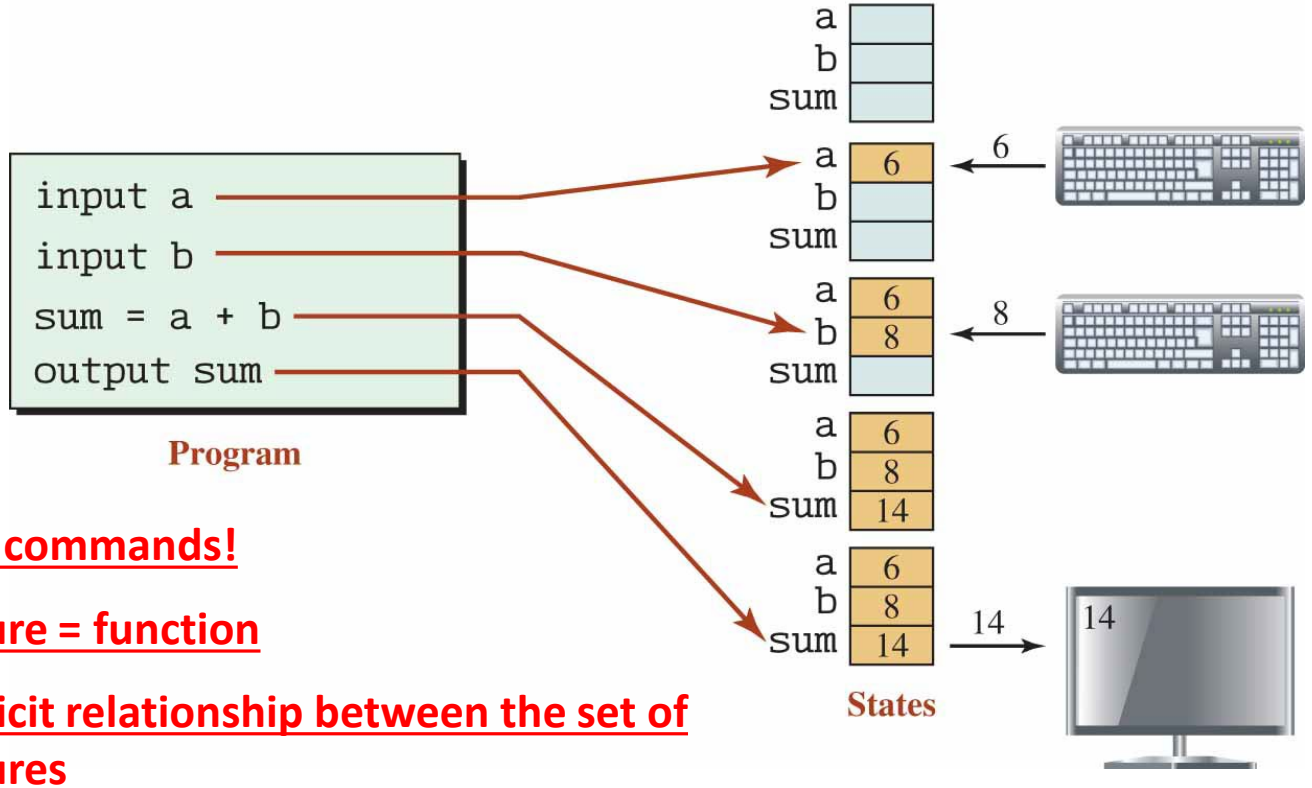
A paradigm is a model or a framework for describing how a program handles data.

Figure 1.10 Language paradigms



Procedural Paradigm

Figure 1.11 An example of a procedural paradigm



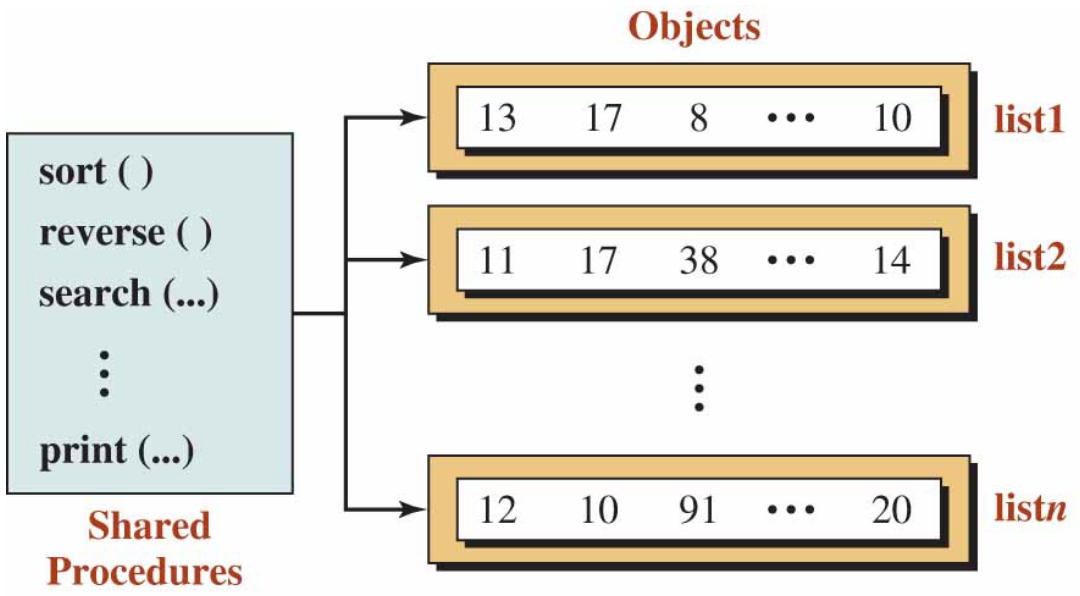
A set of commands!

Procedure = function

No explicit relationship between the set of procedures

Object-Oriented Paradigm

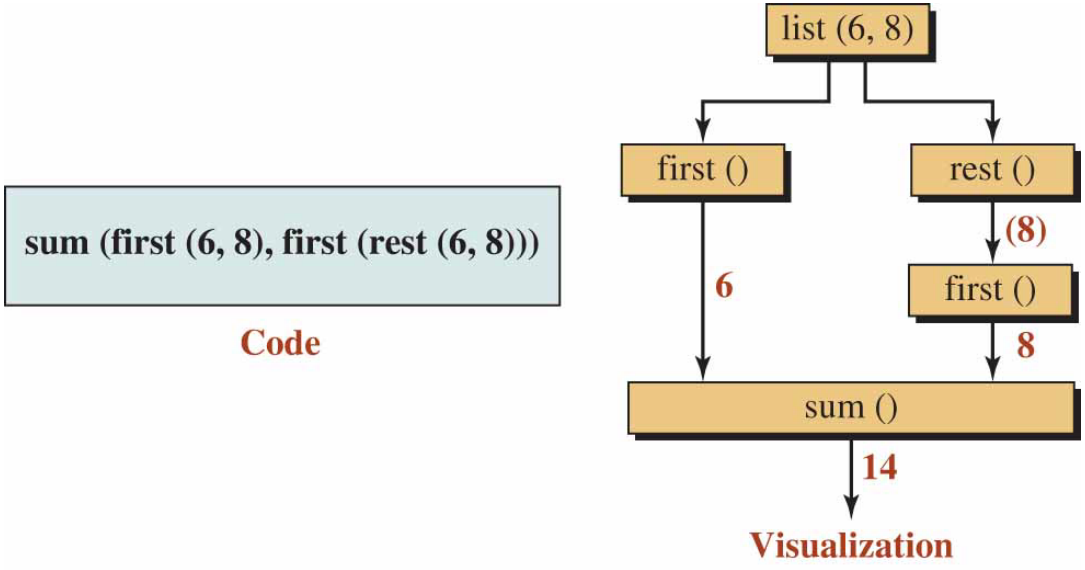
Figure 1.12 ***An example of an object-oriented paradigm***



An object is a package containing all possible operations to particular type of data structure.

Functional Paradigm

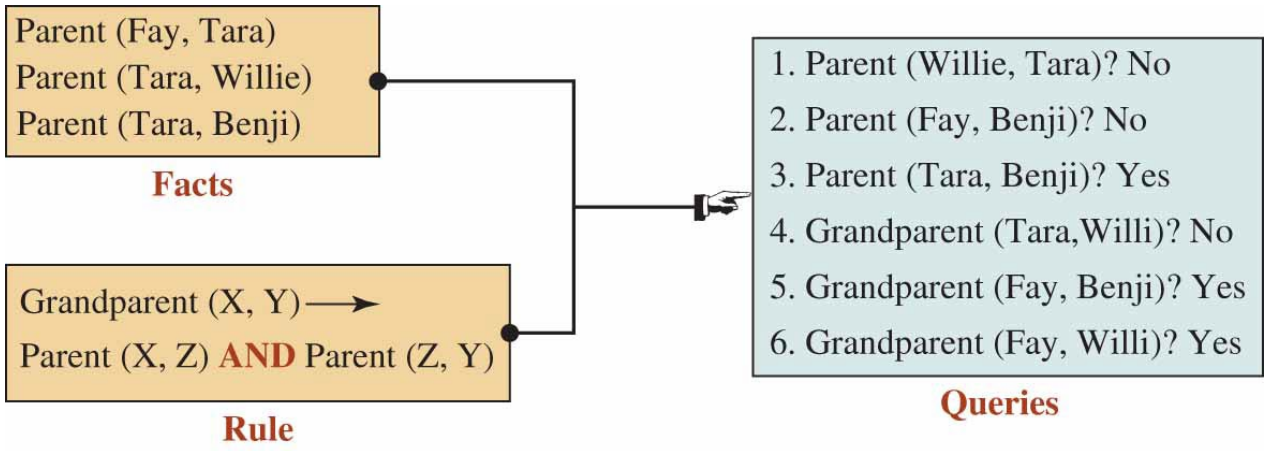
Figure 1.13 *An example of a functional paradigm*



A program is a mathematical function.
A function is a black box that maps a list of inputs to a list of outputs.
We write a program or a new function by combining primitive functions.

Logic Paradigm

Figure 1.14 An example of a logic paradigm



It is based on formal logic using a set of facts and a set of rules to answer queries. A fact such as Parent(Fay, Tara) is read as "fay is the parent of Tara."

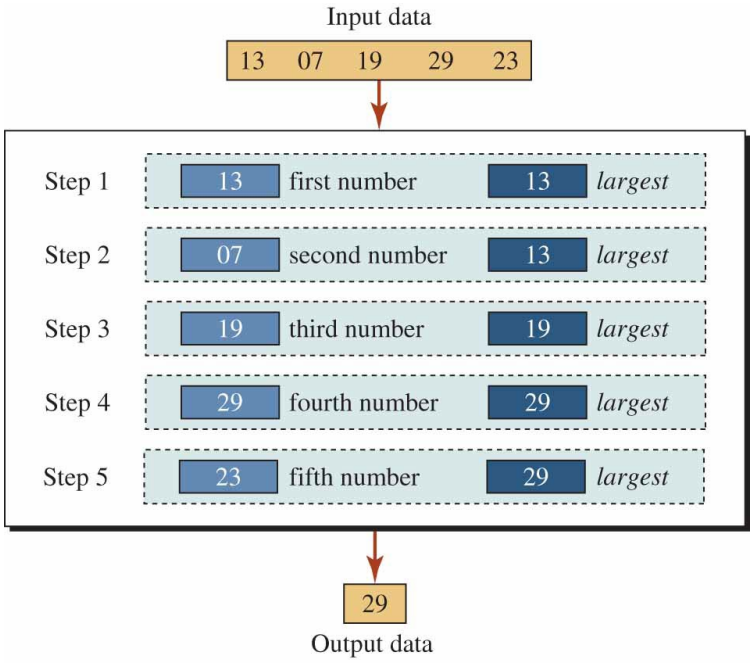
Paradigms in C++ Language

C++ is an extension to the C language and is based on the procedural paradigm.

However, the existence of classes and objects allows the language to be used as an object-oriented language.

Develop the Solution

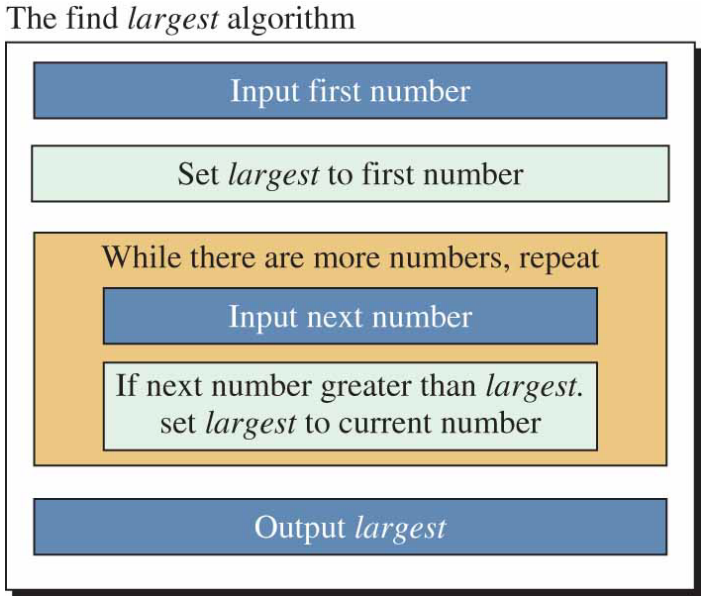
Figure 1.15 Find largest among five integers



A problem consists of an input and an output, and the transformation from the input to the output is unknown. The transform is the solution to the problem.

Algorithm Generalization

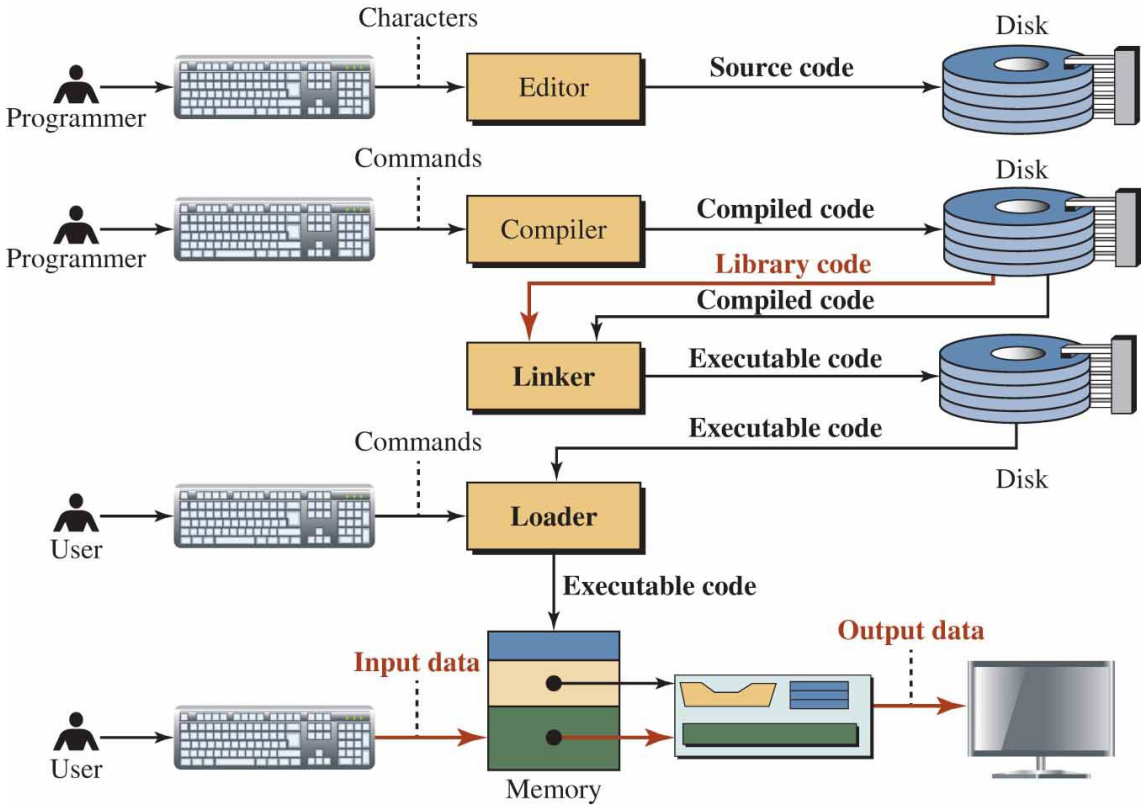
Figure 1.16 Algorithm to find largest among n numbers



Algorithm is a set of logical steps necessary to solve a problem.

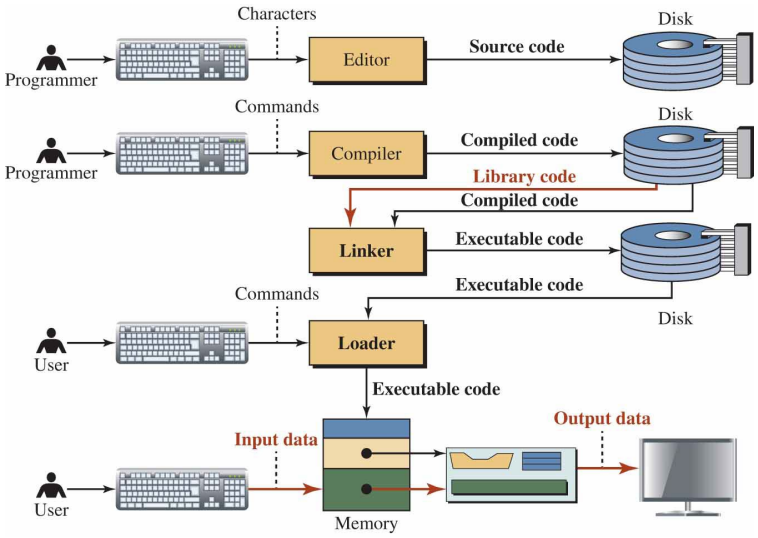
PROGRAM DEVELOPMENT

Figure 1.17 ***Writing, editing, and executing a program***



The general procedure for turning a program written in any language into machine language.

Write And Edit Program

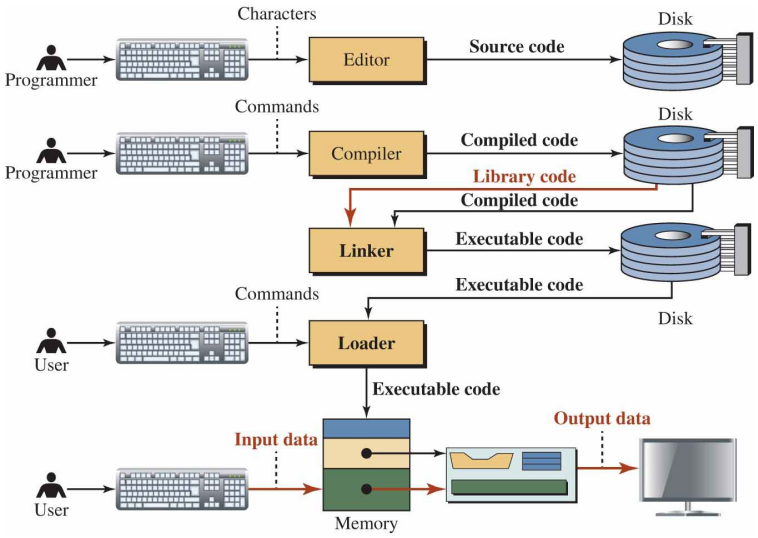


The software used to write programs is known as a text editor.

After we complete a program, we save our file to disk.

This file then becomes the input to the compiler; it is known as a source file.

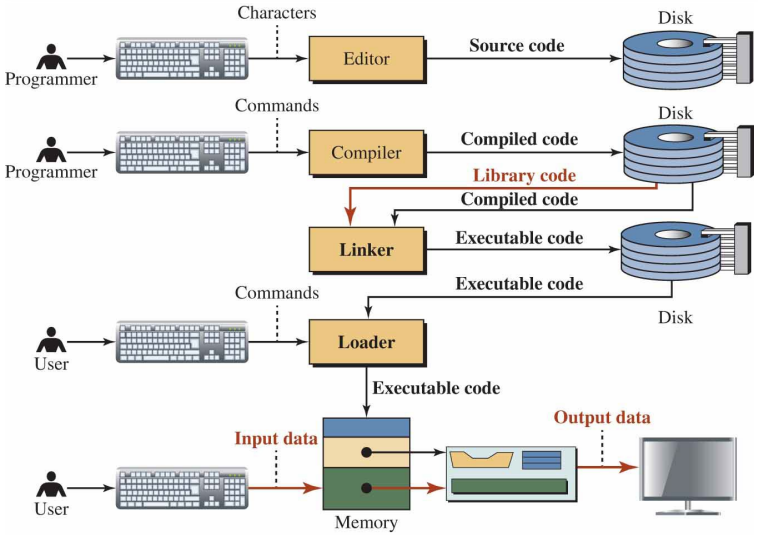
Compile Programs

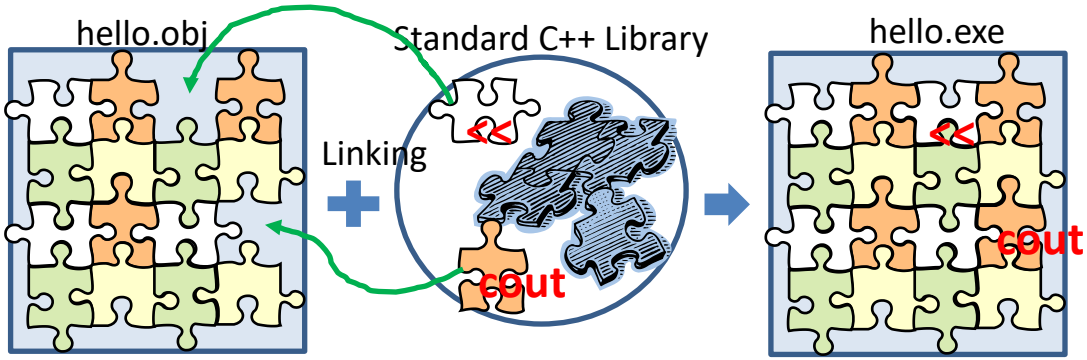


The information in a source file stored on disk must be translated into machine language so the computer can understand it.

This is the job of the compiler.

Link Programs





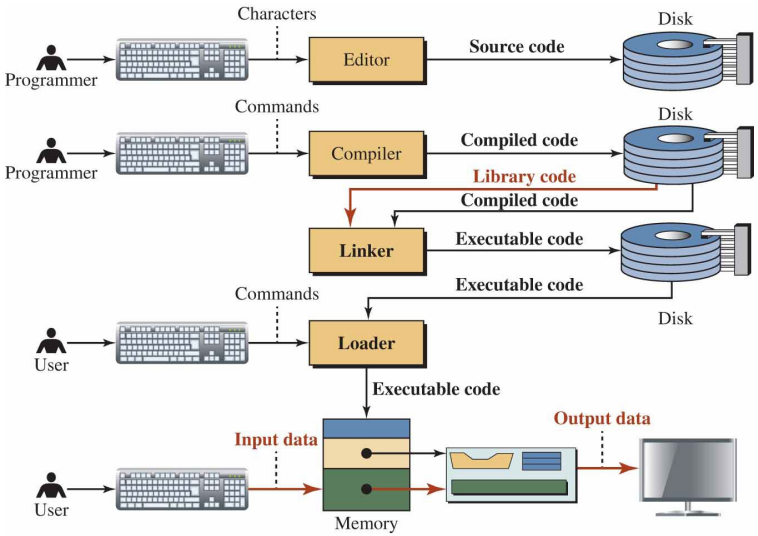
A program is made up of many functions.

Some of these functions are written by us and are a part of our source program.

However, there are other functions, such as input/output processes and mathematical library functions, that exist elsewhere and must be attached to our program.

The linker assembles the system functions and ours into the executable file.

Execute Program



Once the program has been linked, it is ready for execution.

To execute a program we use an operating system command, such as run, to load the program into main memory and execute it.

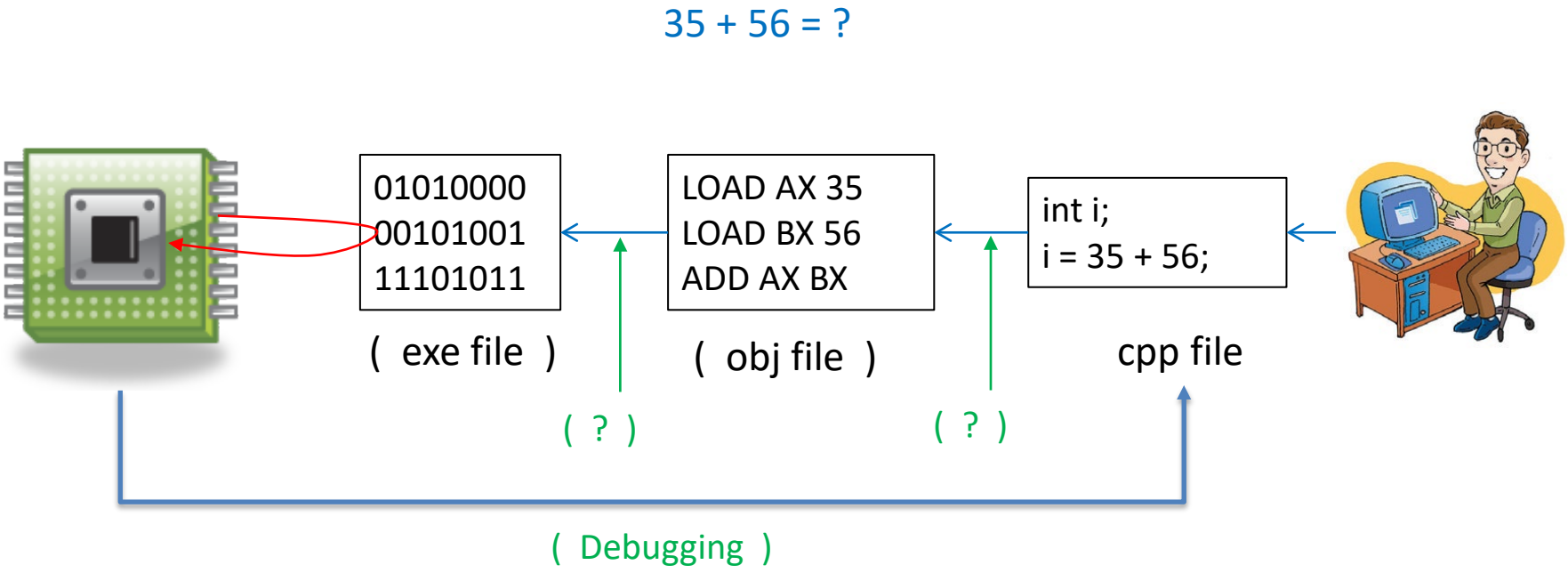
In a typical program execution, the program reads input data for processing.

After the program processes the data, it prepares the output.

Data output can be written to the user's monitor or to a file.

When the program is finished, it tells the operating system, which removes the program from memory.

Generating Executable Program



Program Errors

There are three general classifications of errors:

Specification Errors

Specification errors occur when the problem definition is either incorrectly stated or misinterpreted.

Code Errors

Code errors usually generate a compiler error message. These errors are the easiest to correct.

Logic Errors

The most difficult errors to find and correct are logic errors.
e.g. division by zero. They can be corrected only by through testing.

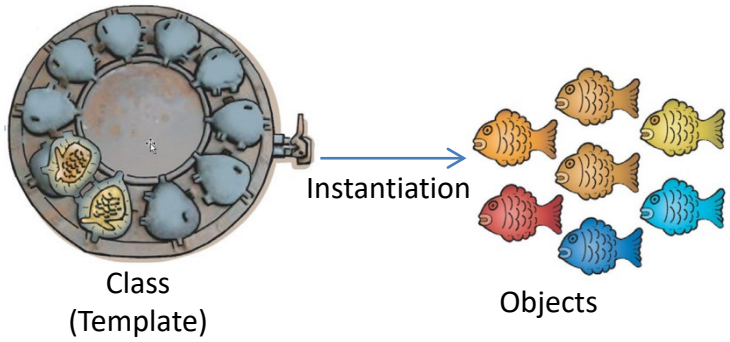
OOP Paradigms: Encapsulation

- Class
  - User defined data type. Operations to access the data
- Object
  - An instance of a class, which has its own unique state and behavior.
- Encapsulation:
  - practice of hiding the internal details of an object and exposing only the necessary functionality through its interface.
  - It helps to reduce the complexity of the program and prevent errors by preventing unauthorized access to an object's internal state.

Members

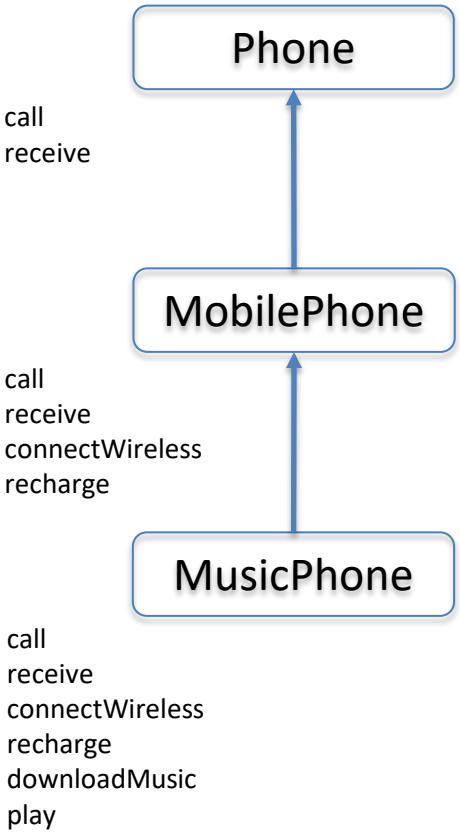
```
class Circle {
  private:
     int radius;
  public:
     Circle(int r) { radius = r; }
     double getArea() { return 3.14*radius*radius; }
};

Circle c1;
Circle c2;
```



OOP Paradigms: Inheritance

- Inheritance
  - It allows classes to inherit attributes and methods from other classes.
  - It enables code reuse.




```
class Phone {
    void call();
    void receive();
};
```

```
class MobilePhone  : public Phone{
    void connectWireless();
    void recharge();
};
```

```
class MusicPhone  : public MobilePhone{
    void downloadMusic();
    void play();
};
```

OOP Paradigms: Polymorphism

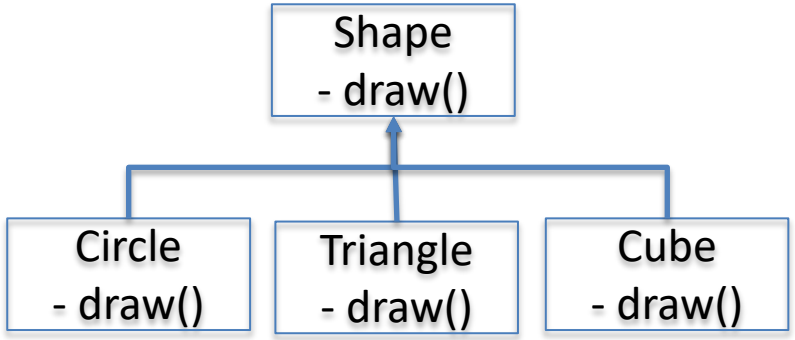
- Polymorphism
  - It allows objects of different classes to be treated as if they were of the same class.
  - It enables code to be more flexible and adaptable, as the same method can be applied to different objects.

```
void add(int a, int b) { ... }
void add(int a, int b, int c) { ... }
void add(int a, double d) { ... }
```

add function overloading

```
      2 + 3            --> 5
  "Male" + "Female"            --> "MaleFemale"
Color red + Color blue   --> Color purple
```

+ operator overloading



draw function overriding

OOP Paradigms: Generic Programming

- Type independent code
  - Write code for one generic type.
  - Use it for multiple types.

```
Without Generic Programming:
   void swap(int & x, int & y)      { int tmp = x; x = y; y = tmp;    }
   void swap(long & x, long & y){ long tmp = x; x = y; y = tmp; }
   …

With Generic Programming
template <typename T>
    void swap(T & x, T & y) {  T tmp = x; x = y; y = tmp; }
```

OOP Paradigms

- Why OOP?
  - It provides a powerful set of tools for organizing and structuring code, can lead to more efficient and maintainable programs.

- Improving productivity in programming.
  - Reusable codes.
  - Readable and understandable in code.
  - Reducing errors.

  - In short, OOP helps to save your time in programming!

End of Class

Thank you

E-mail: youngcha@konkuk.ac.kr