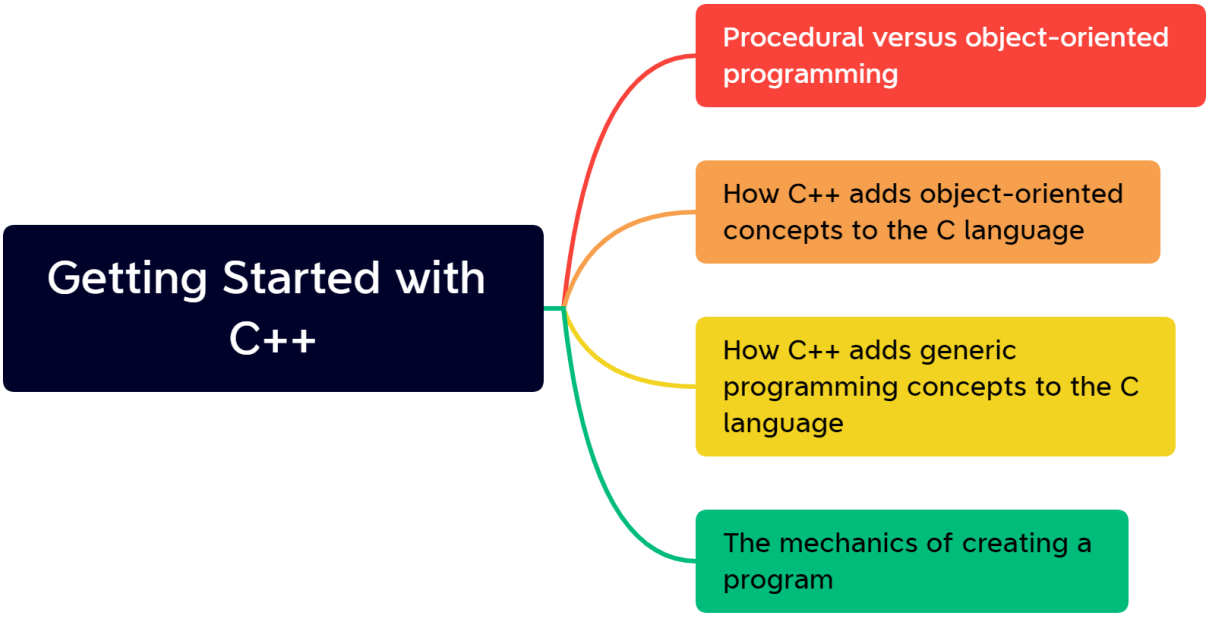
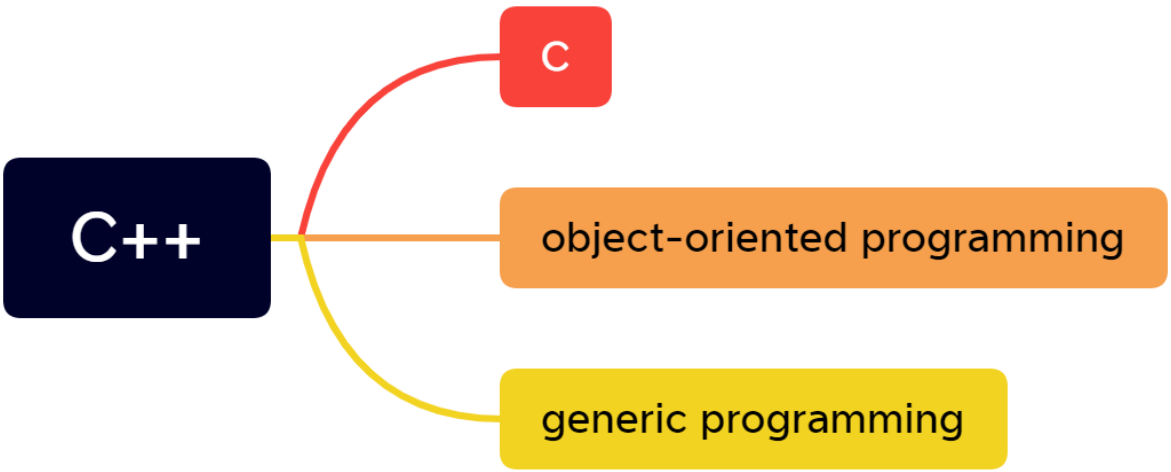
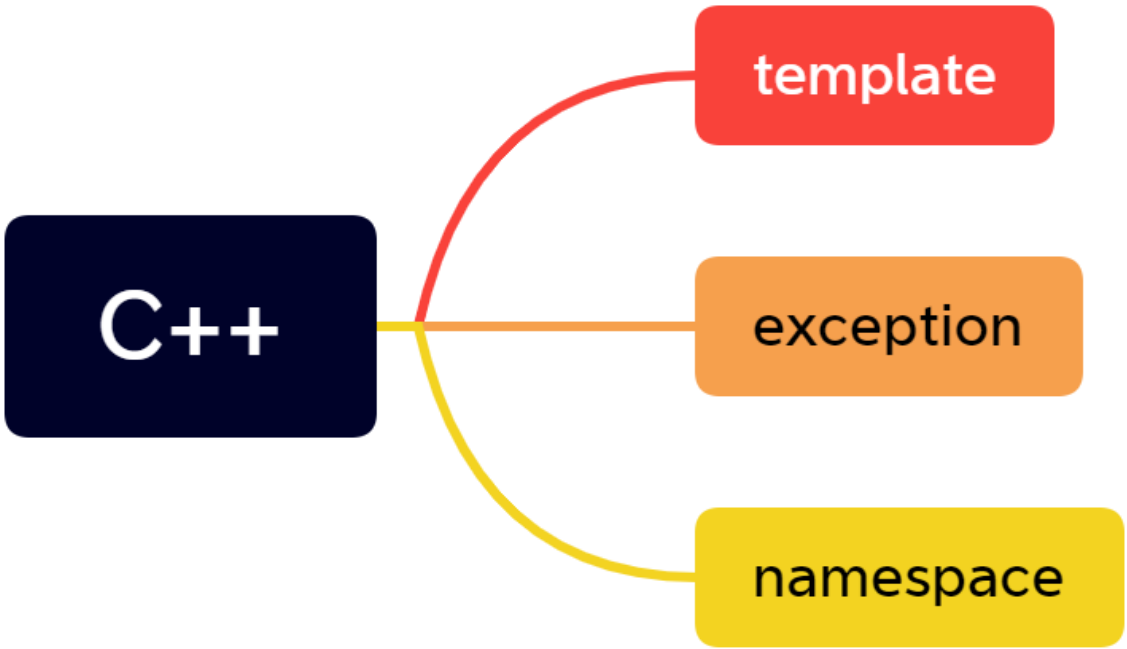
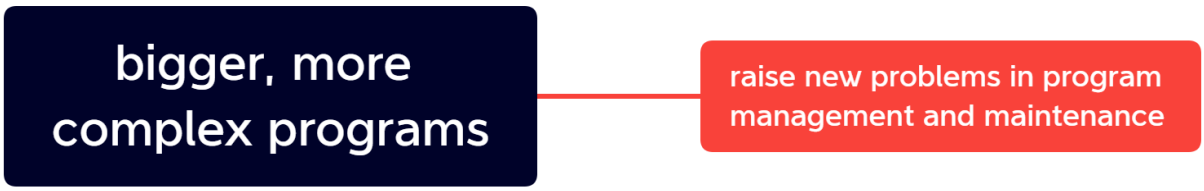






template, exception, namespace



computer languages deal with two concepts—data and algorithms.
The data constitutes the information a program uses and processes.The algorithms are the methods the program uses

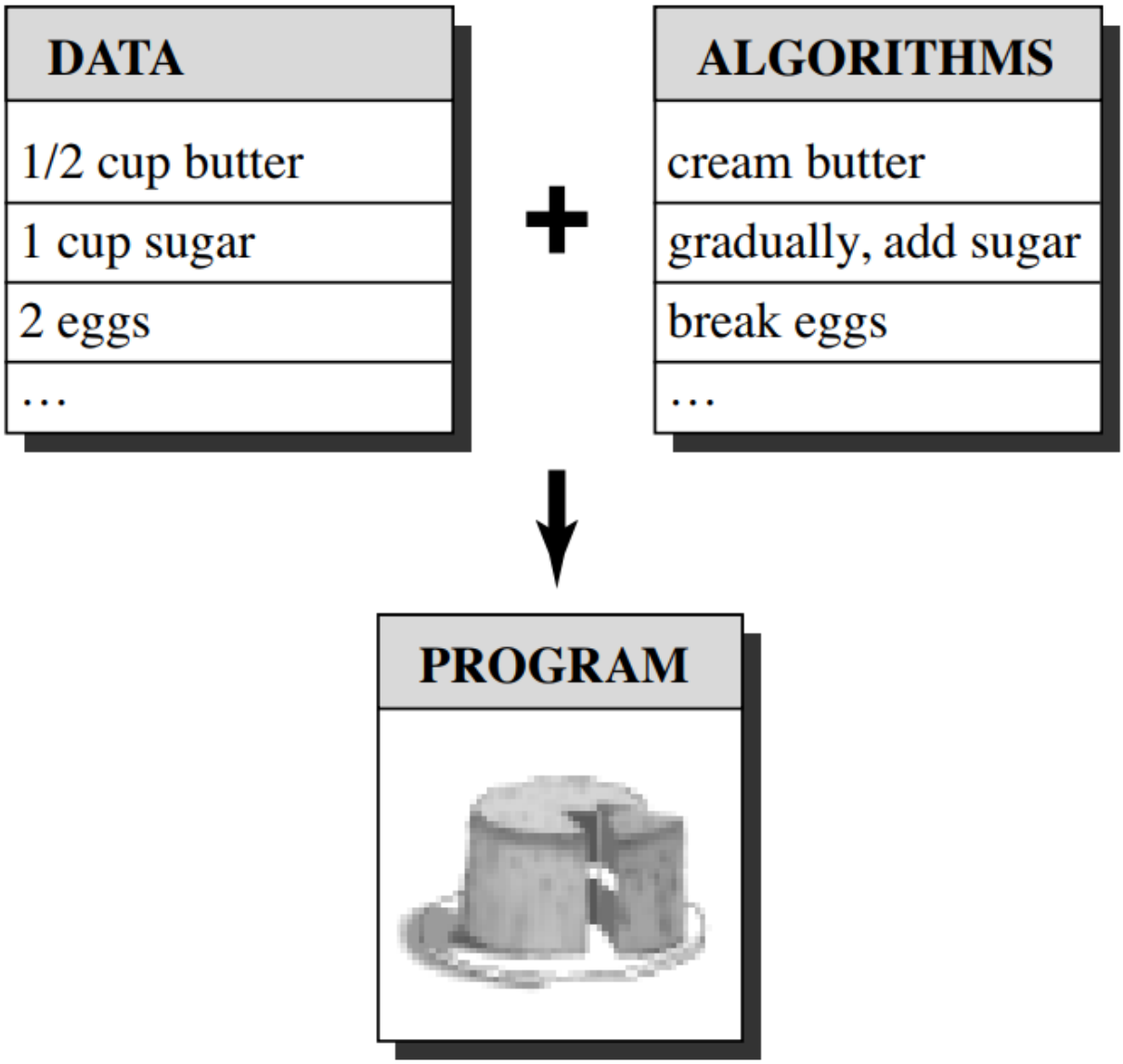


Figure 1.1 Data + algorithms = program.

C is a procedural language.

it emphasizes the algorithm side of programming.

Conceptually, procedural programming consists of figuring out the actions a computer should take and then using the programming language to implement those actions.

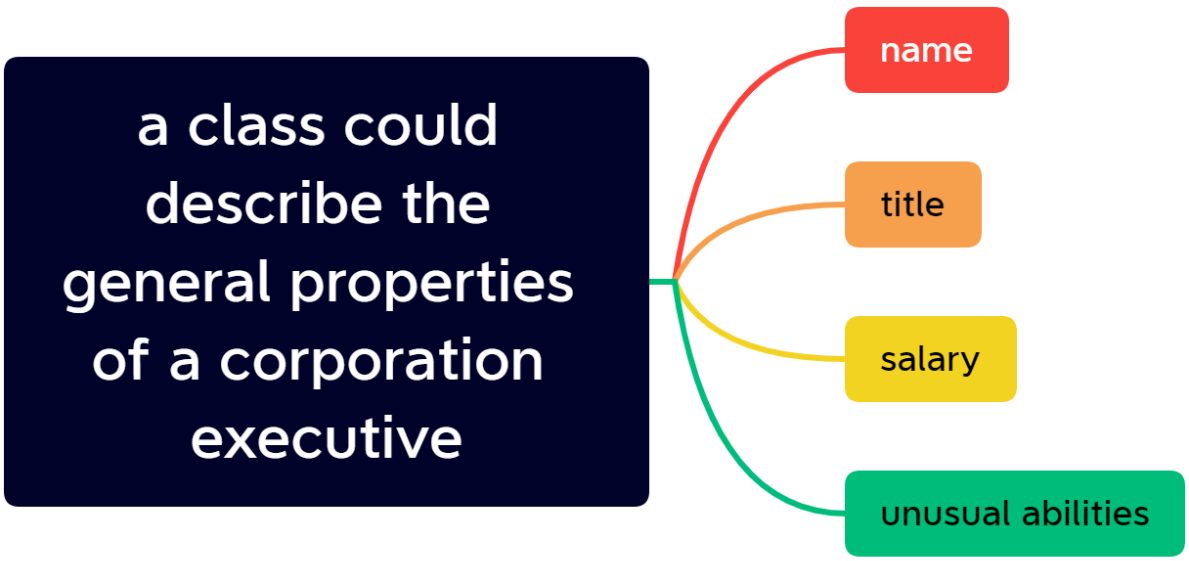
structured programming

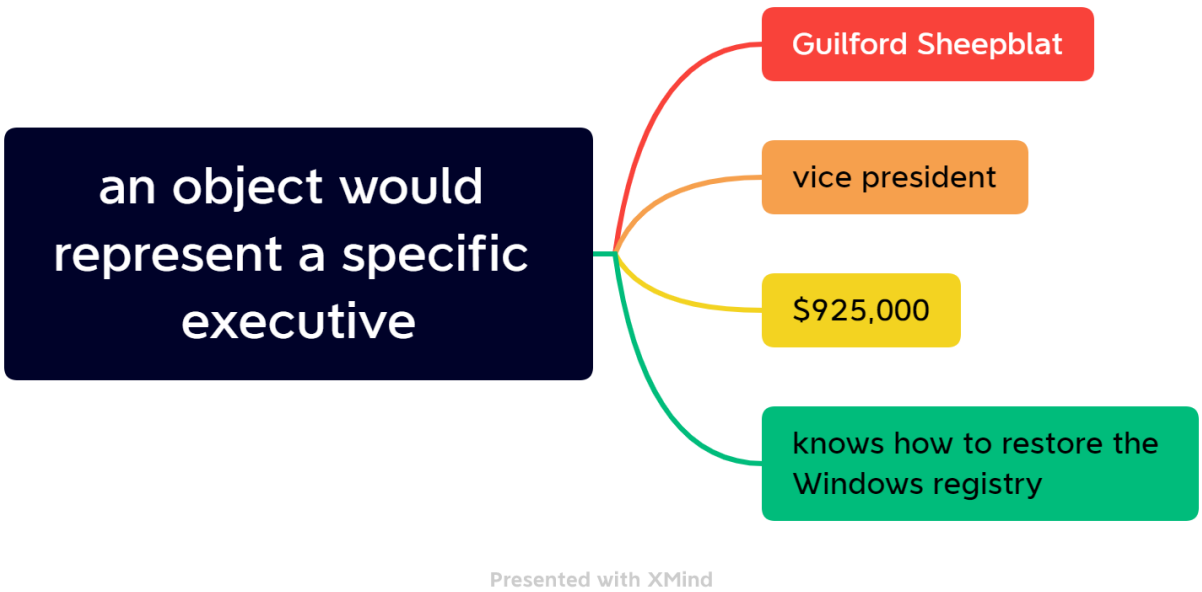
Top-down design

With C, the idea is to break a large program into smaller, more manageable tasks. C’ s design facilitates this approach, encouraging you to develop program units called functions to represent individual task modules.

The C++ Shift: Object-Oriented Programming

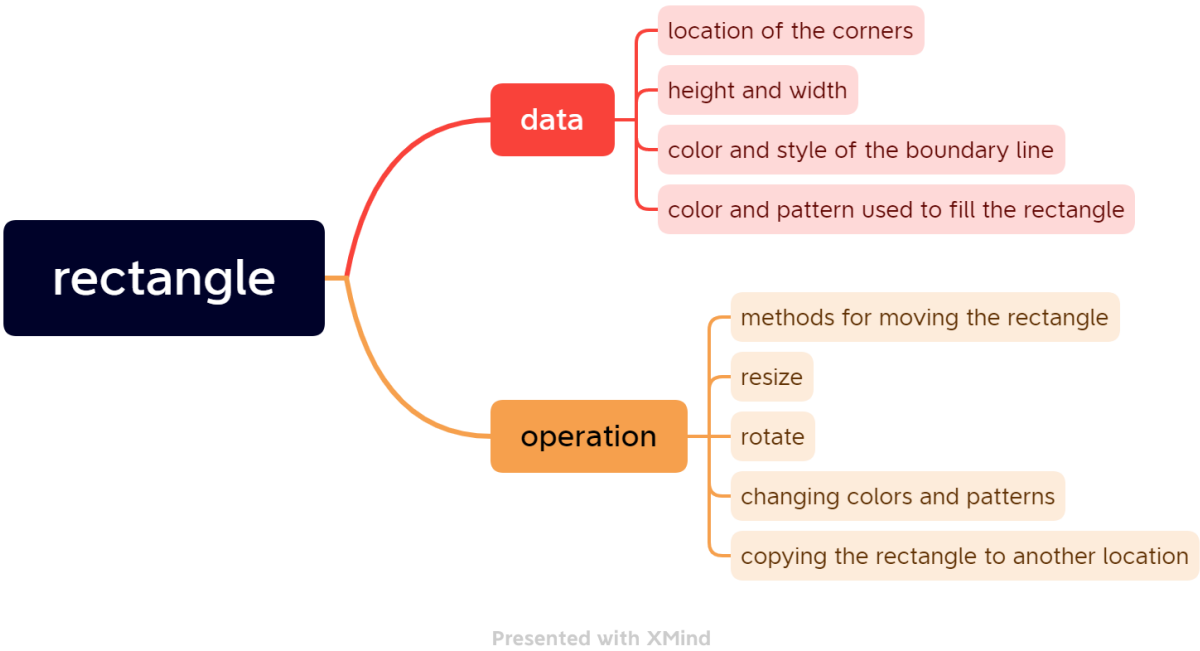
large-scale programming still remains a challenge.
OOP brings a new approach to that challenge.
OOP emphasizes the data
OOP attempts to fit the language to the problem.
The idea is to design data forms that correspond to the essential features of a problem.
a class is a specification describing such a new data form, and an object is a particular data structure constructed according to that plan.





In general, a class defines what data is used to represent an object and the operations that can be performed on that data.

For example, suppose you were developing a computer drawing program capable of drawing a rectangle.You could define a class to describe a rectangle.



If you then used your program to draw a rectangle, it would create an object according to the class specification.That object would hold all the data values describing the rectangle, and you could use the class methods to modify that rectangle.

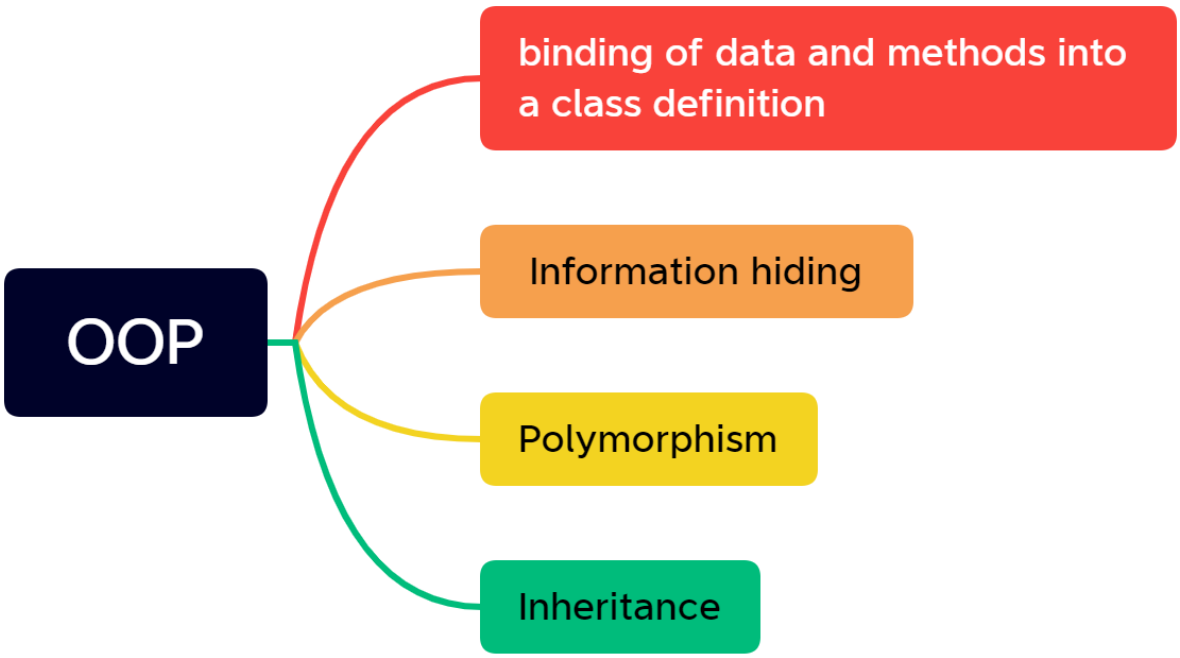
The process of going from a lower level of organization, such as classes, to a higher level, such as program design, is called **bottom-up** programming.

There’ s more to OOP than the binding of data and methods into a class definition.

OOP facilitates creating reusable code

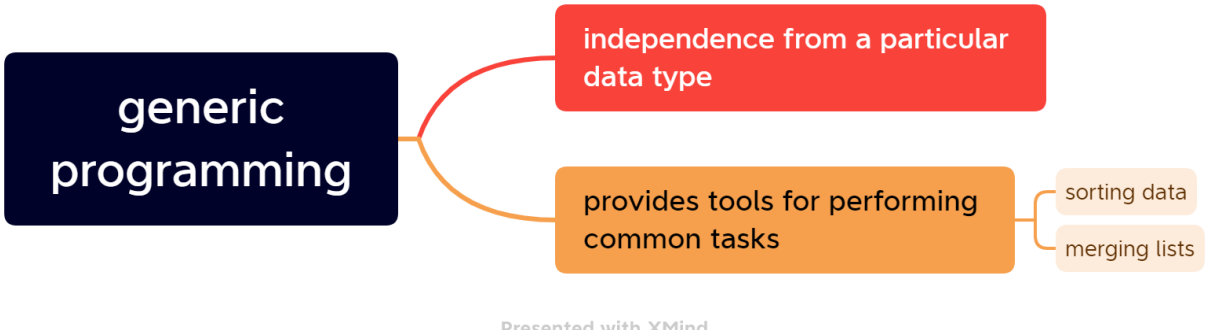
Polymorphism lets you create multiple definitions for operators and functions, with the programming context determining which definition is used.

Inheritance lets you derive new classes from old ones.



OOP languages make it simple to incorporate existing classes into your own programming.

C++ and Generic Programming



The term **generic** refers to code that is type independent.

C++ data representations come in many types—integers, numbers with fractional parts, characters, strings of characters, and user-defined compound structures of several types. If, for example, you wanted to sort data of these various types, you would normally have to create a separate sorting function for each type. Generic programming involves extending the language so that you can write a function for a generic (that is, an unspecified) type once and use it for a variety of actual types. C++ templates provide a mechanism for doing that.

The Genesis of C++

C++ programs can use existing C software libraries.

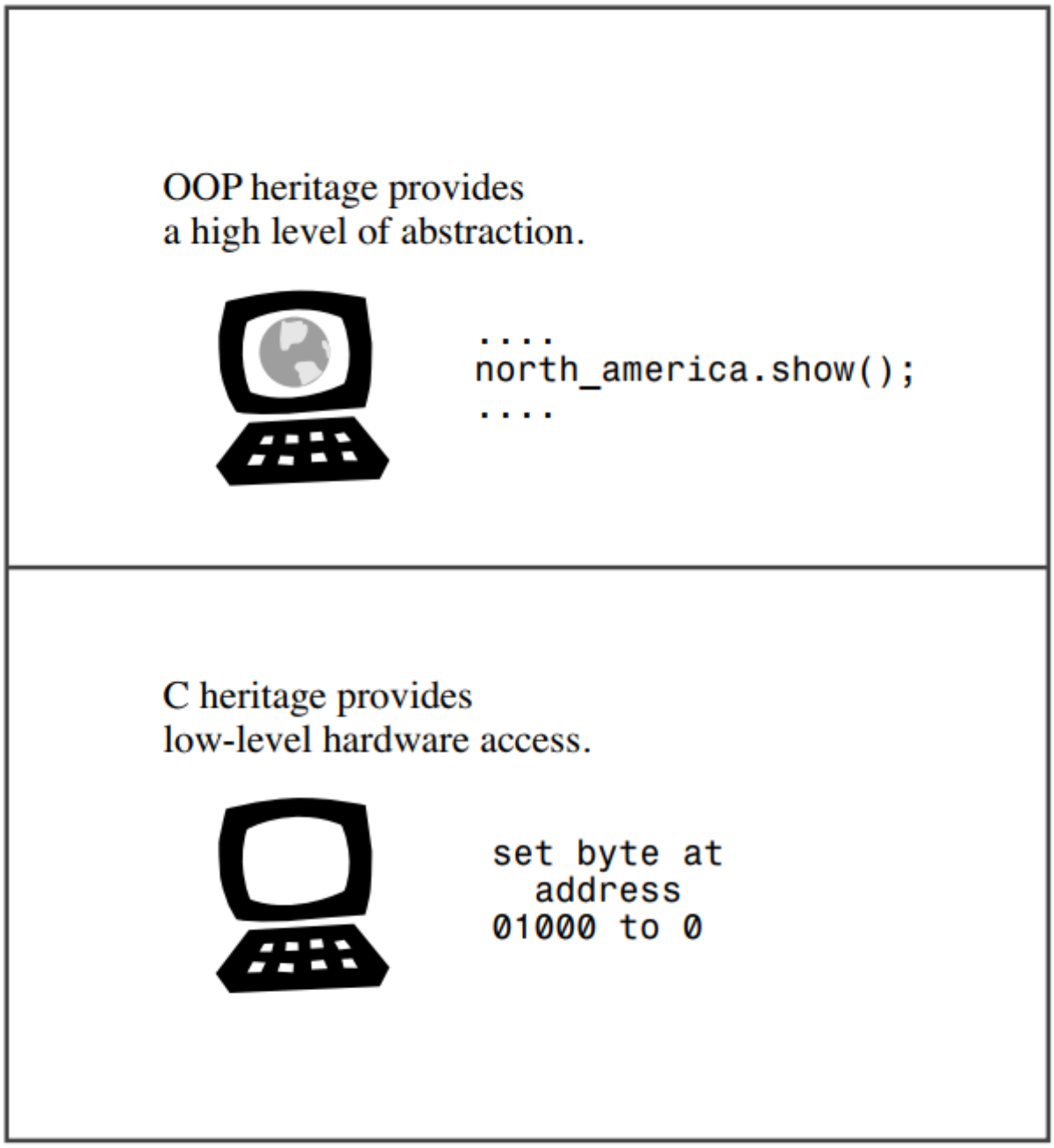


Figure 1.2 C++ duality.

The Mechanics of Creating a Program

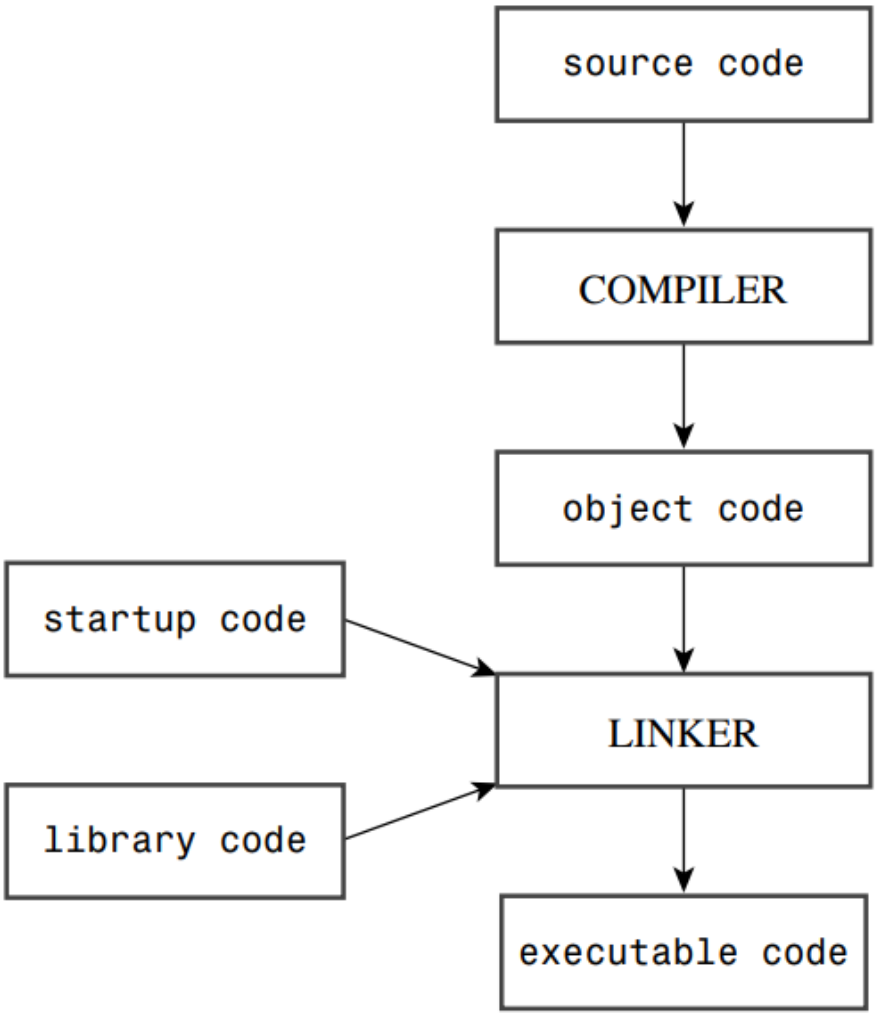


Figure 1.3 Programming steps.

Creating the Source Code File

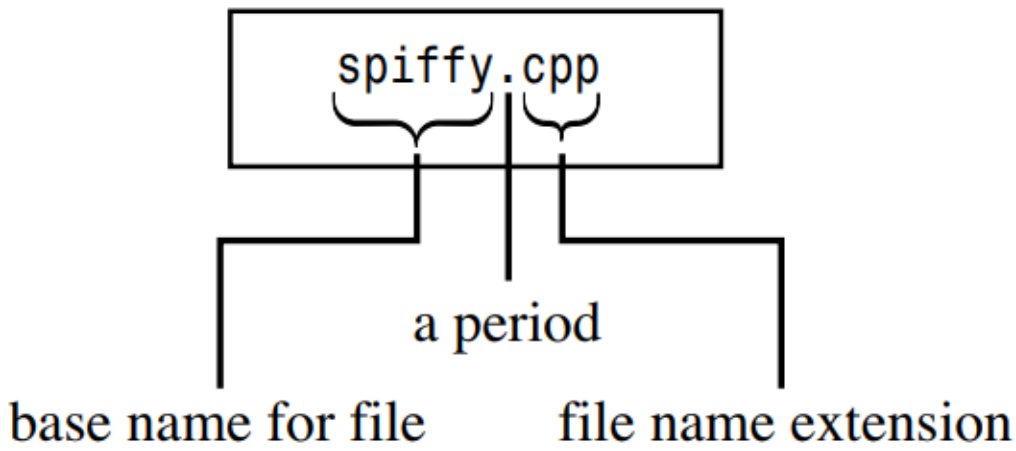


Figure 1.4 The parts of a source code filename.

Table 1.1 **Source Code Extensions**

C++ Implementation	Source Code Extension(s)
Unix	`C`, `cc`, `cxx`, `c`
GNU C++	`C`, `cc`, `cxx`, `cpp`, `c++`
Digital Mars	`cpp`, `cxx`
Borland C++	`cpp`
Watcom	`cpp`
Microsoft Visual C++	`cpp`, `cxx`, `cc`
Freestyle CodeWarrior	`cpp`, `cp`, `cc`, `cxx`, `c++`