Chapter 13

Templates

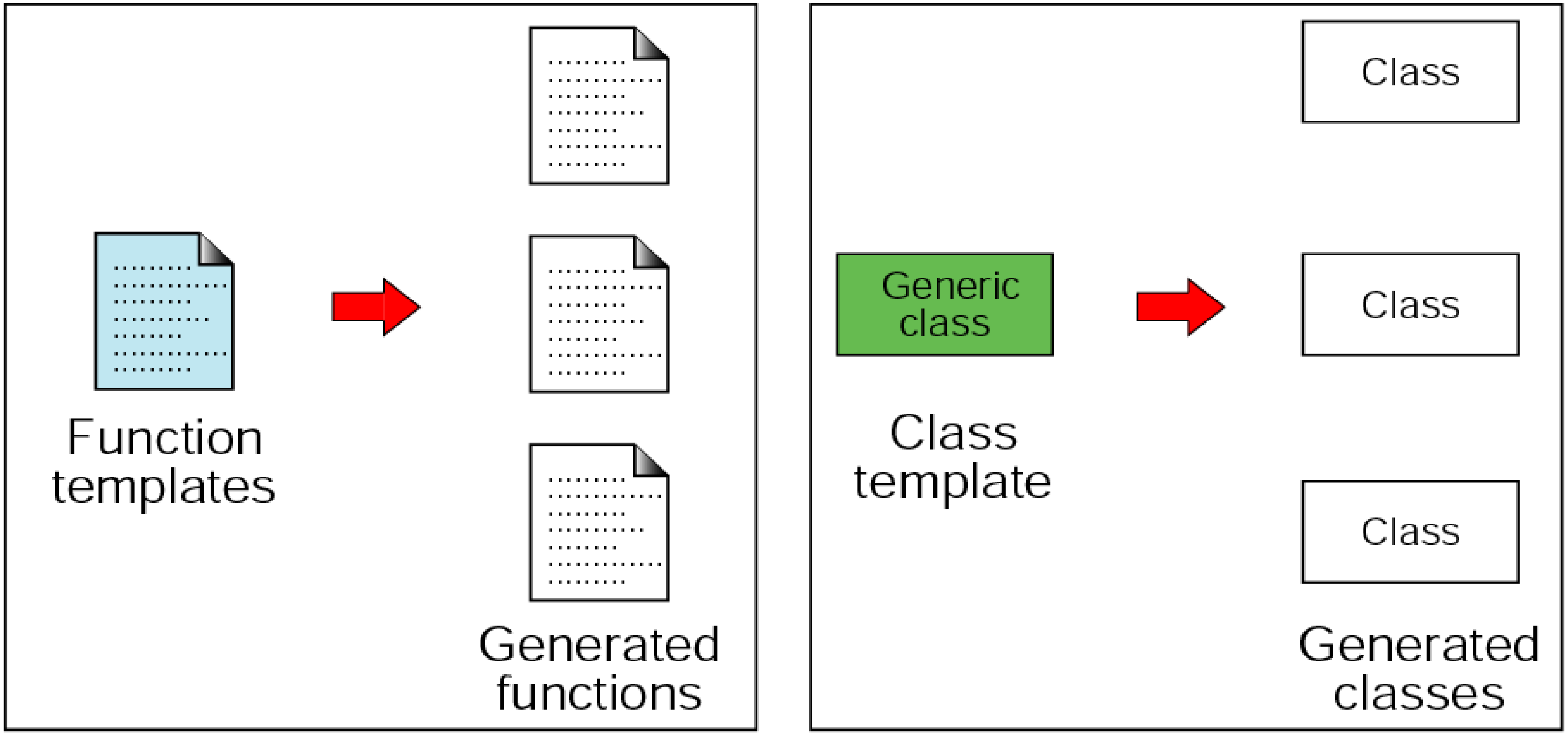
Brooks/Cole
Thomson Learning™

OBJECTIVES

After studying this chapter you will be able to:

- **Understand and create templates for functions and classes.**
- **Overload a function template.**
- **Understand the differences between generic and concrete types.**
- **Understand the concepts of atomic and composite types.**
- **Understand the concept of an Abstract Data Type (ADT).**

Figure 13-1 Basic template concepts



(a) Function template

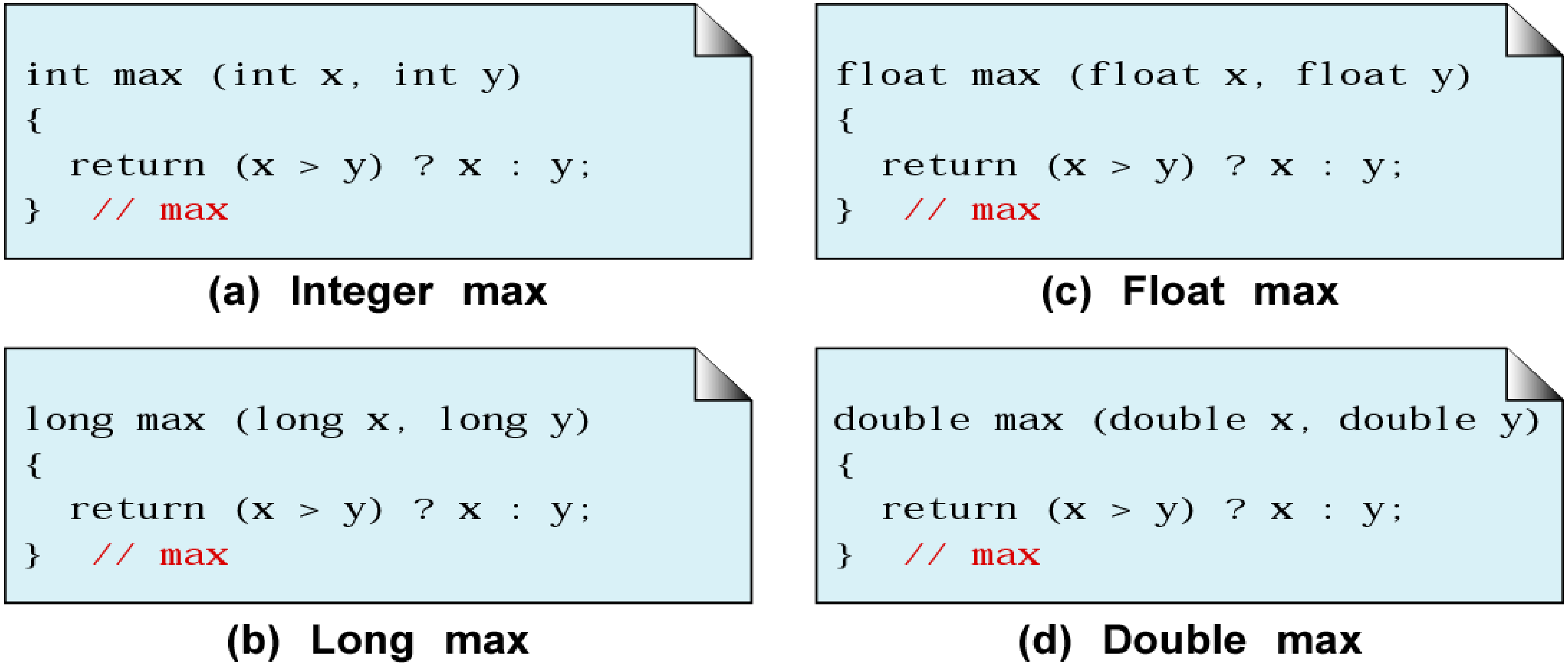
(b) Class template

Brooks/Cole
Thomson Learning™

13.1

FUNCTION TEMPLATES

Figure 13-2 Multiple max functions



```
int max (int x, int y)
{
  return (x > y) ? x : y;
}  // max
```

(a) Integer max

```
long max (long x, long y)
{
  return (x > y) ? x : y;
}  // max
```

(b) Long max

```
float max (float x, float y)
{
  return (x > y) ? x : y;
}  // max
```

(c) Float max

```
double max (double x, double y)
{
  return (x > y) ? x : y;
}  // max
```

(d) Double max

Note:

A function template can create multiple functions, each with potentially different arguments and return types.

Figure 13-3 Function template operation

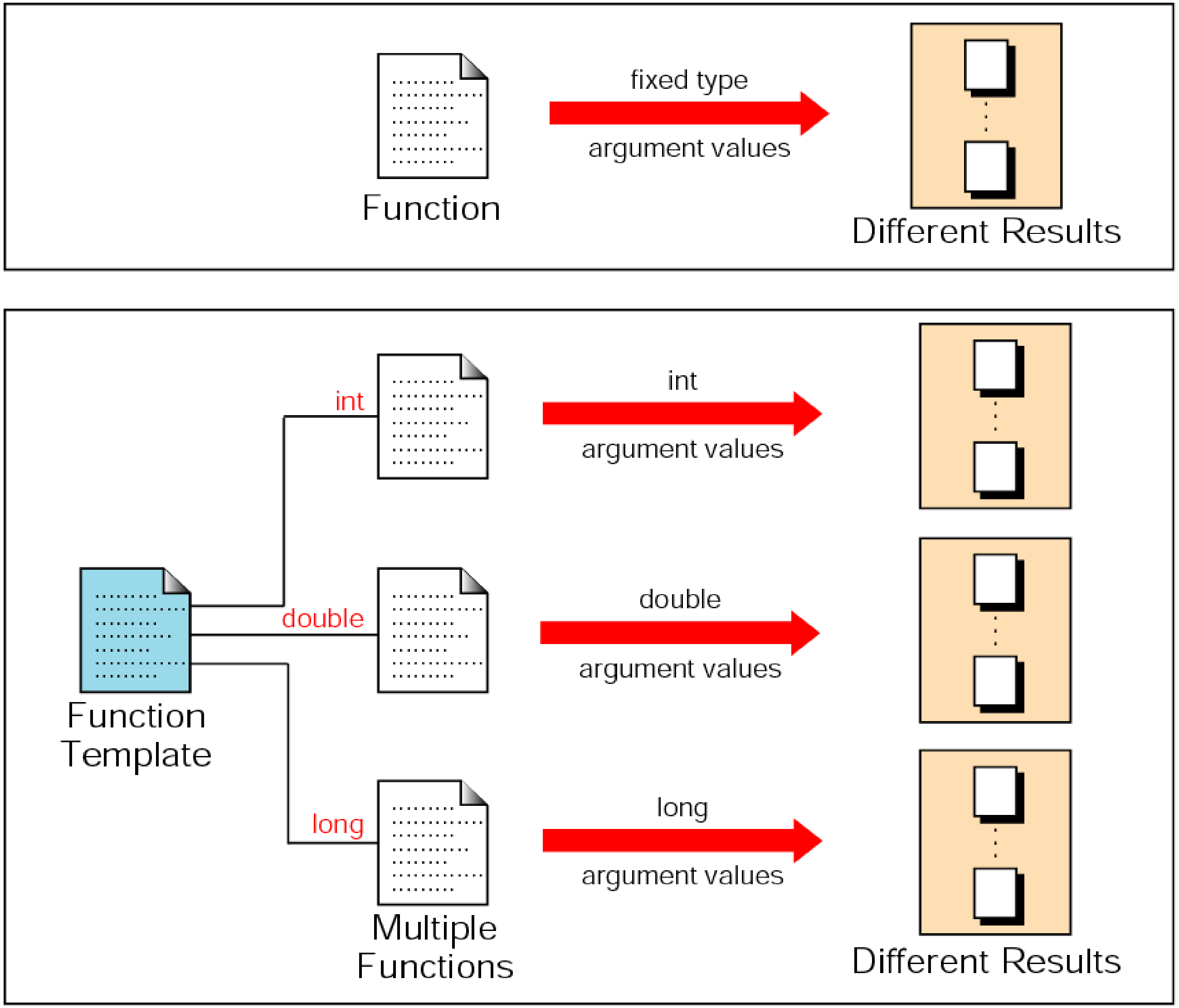
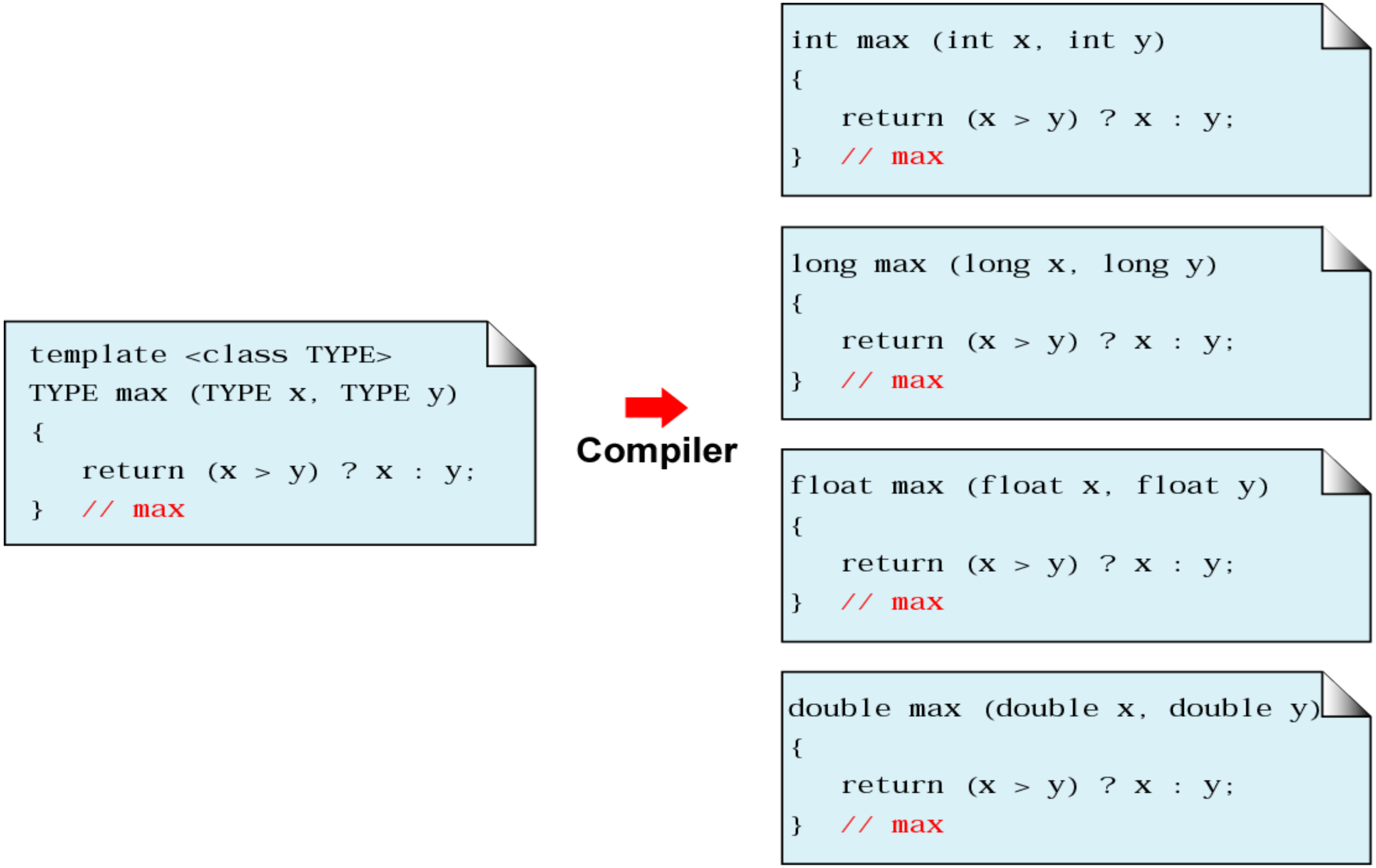


Figure 13-4 Generated functions



```
template <class TYPE>
TYPE max (TYPE x, TYPE y)
{
   return (x > y) ? x : y;
}  // max
```

```
int max (int x, int y)
{
   return (x > y) ? x : y;
}  // max
```

```
long max (long x, long y)
{
   return (x > y) ? x : y;
}  // max
```

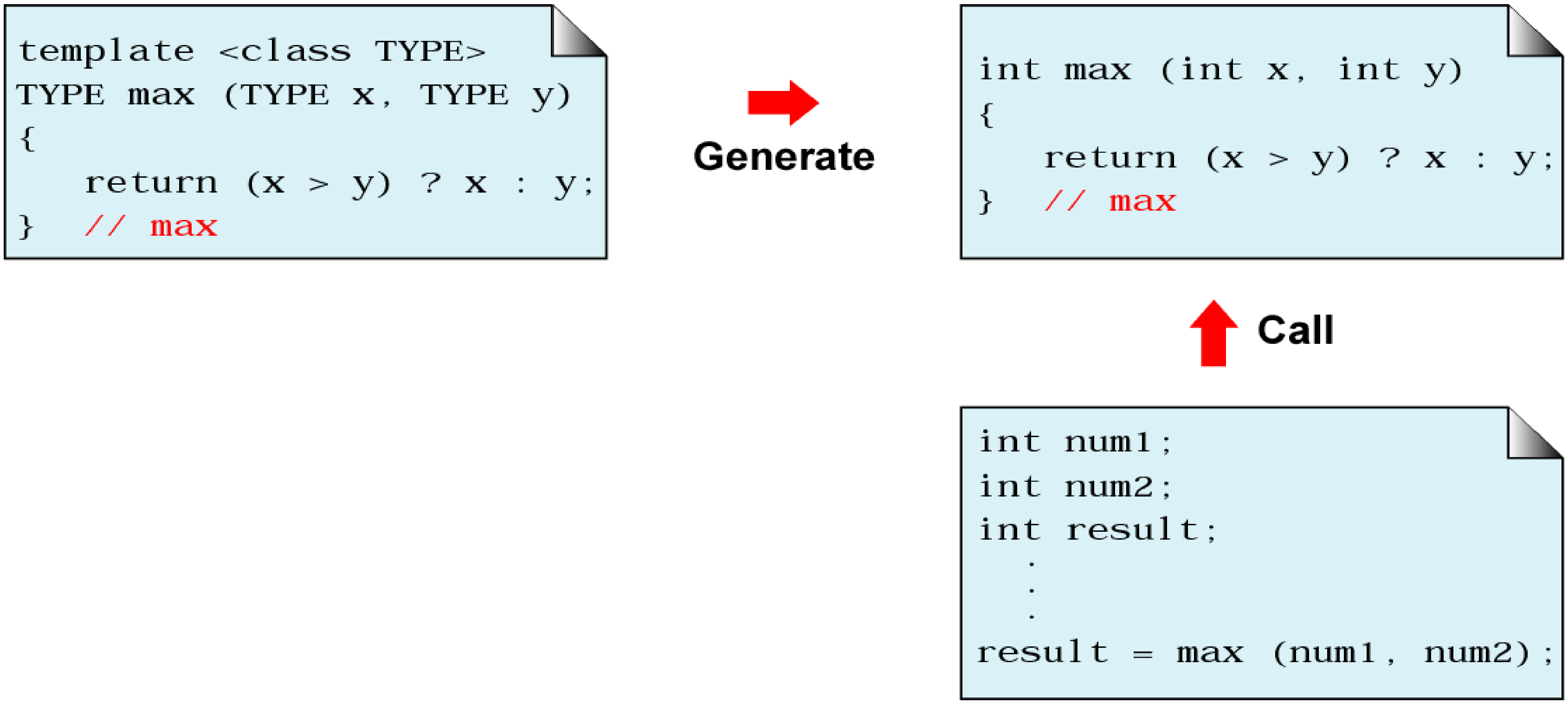
```
float max (float x, float y)
{
   return (x > y) ? x : y;
}  // max
```

```
double max (double x, double y)
{
   return (x > y) ? x : y;
}  // max
```

Note:

Function templates allow us to write a single function for a whole family of similar functions.

Figure 13-5 Function template generation

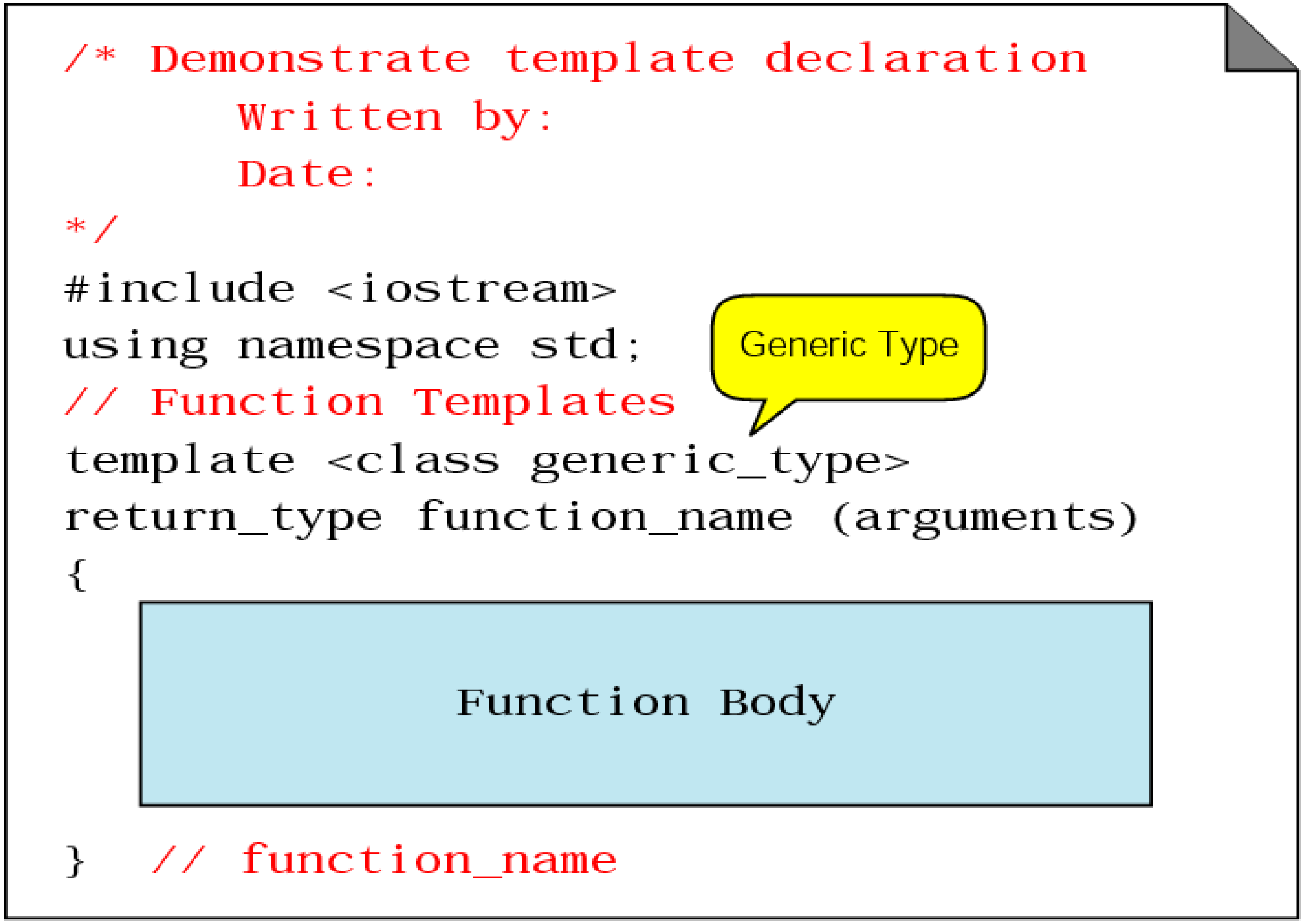


```
template <class TYPE>
TYPE max (TYPE x, TYPE y)
{
   return (x > y) ? x : y;
}  // max
```

```
int max (int x, int y)
{
   return (x > y) ? x : y;
}  // max
```

```
int num1;
int num2;
int result;
   .
   .
   .
result = max (num1, num2);
```

Figure 13-6 Function declaration

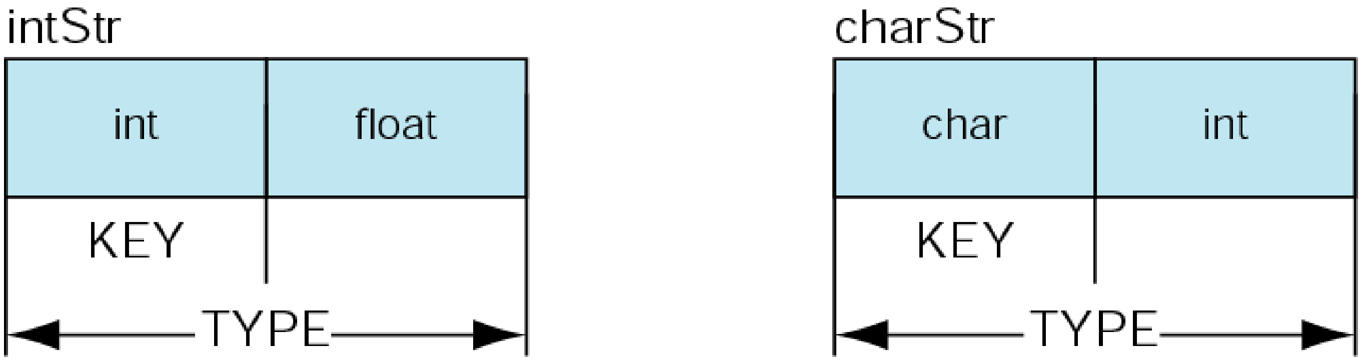


```
/* Demonstrate template declaration
      Written by:
      Date:
*/
#include <iostream>
using namespace std;
// Function Templates
template <class generic_type>
return_type function_name (arguments)
{
            Function Body

}  // function_name
```

Generic Type

Figure 13-7 Structure design for search template


intStr

int	float
KEY	

TYPE

charStr

char	int
KEY	

TYPE

TYPE → intStr

KEY → int

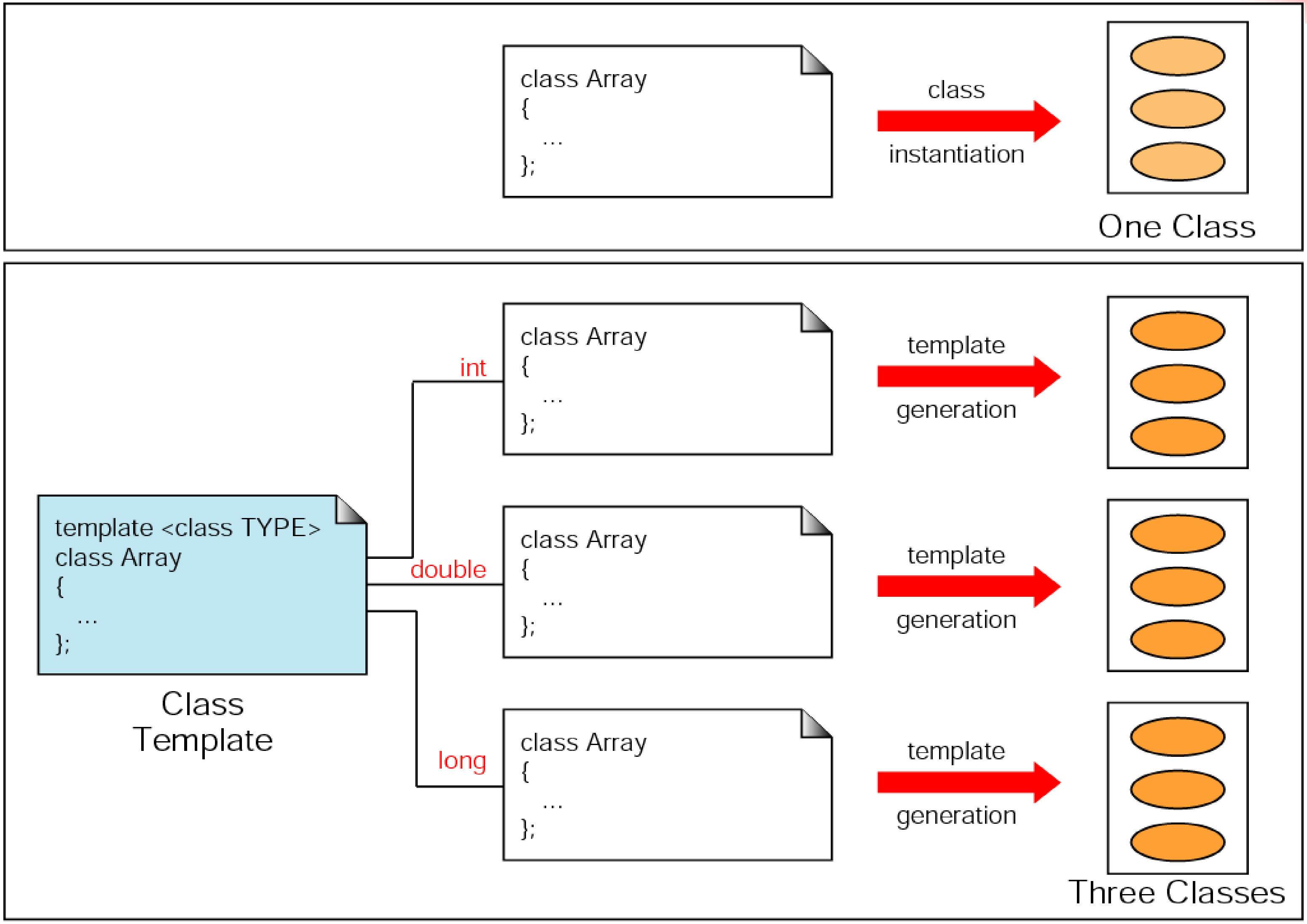
TYPE → charStr

KEY → char

13.2

Class Templates

Figure 13-8 Class template operation



13.3

Software Engineering and Programming Style

Brooks/Cole
Thomson Learning™

Note:

Atomic Data Type

1. A set of values

2. A set of operations on values

Note:

Data Structure

1. A combination of elements, each of which is either an atomic type or another data structure

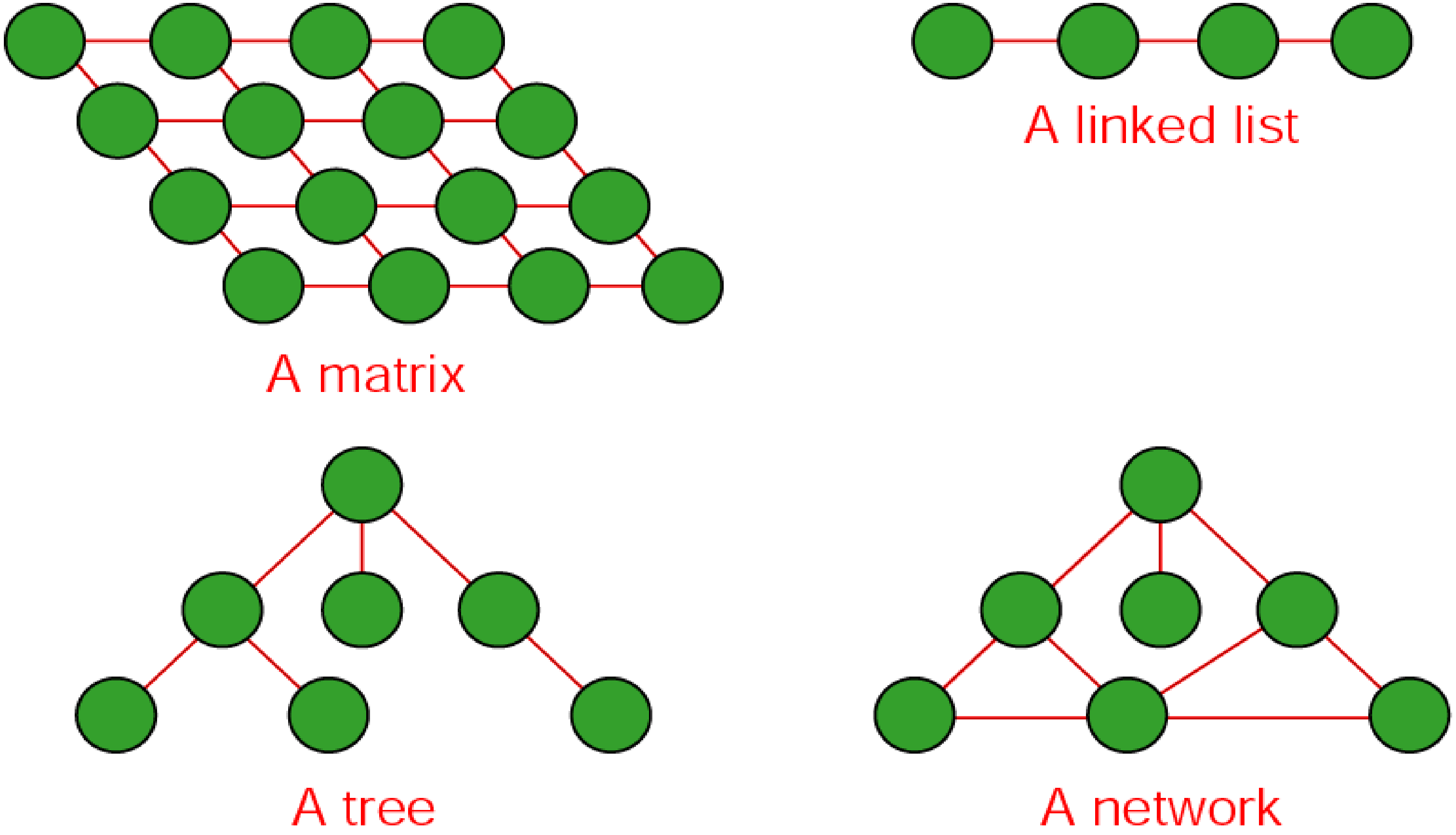
2. A set of associations or relationships (structure) involving the combined elements

Note:

The concept of abstraction means:

- ***We know what a data type can do.***
- ***How it is done is hidden.***

Figure 13-9 Some structures



A matrix

A linked list

A tree

A network

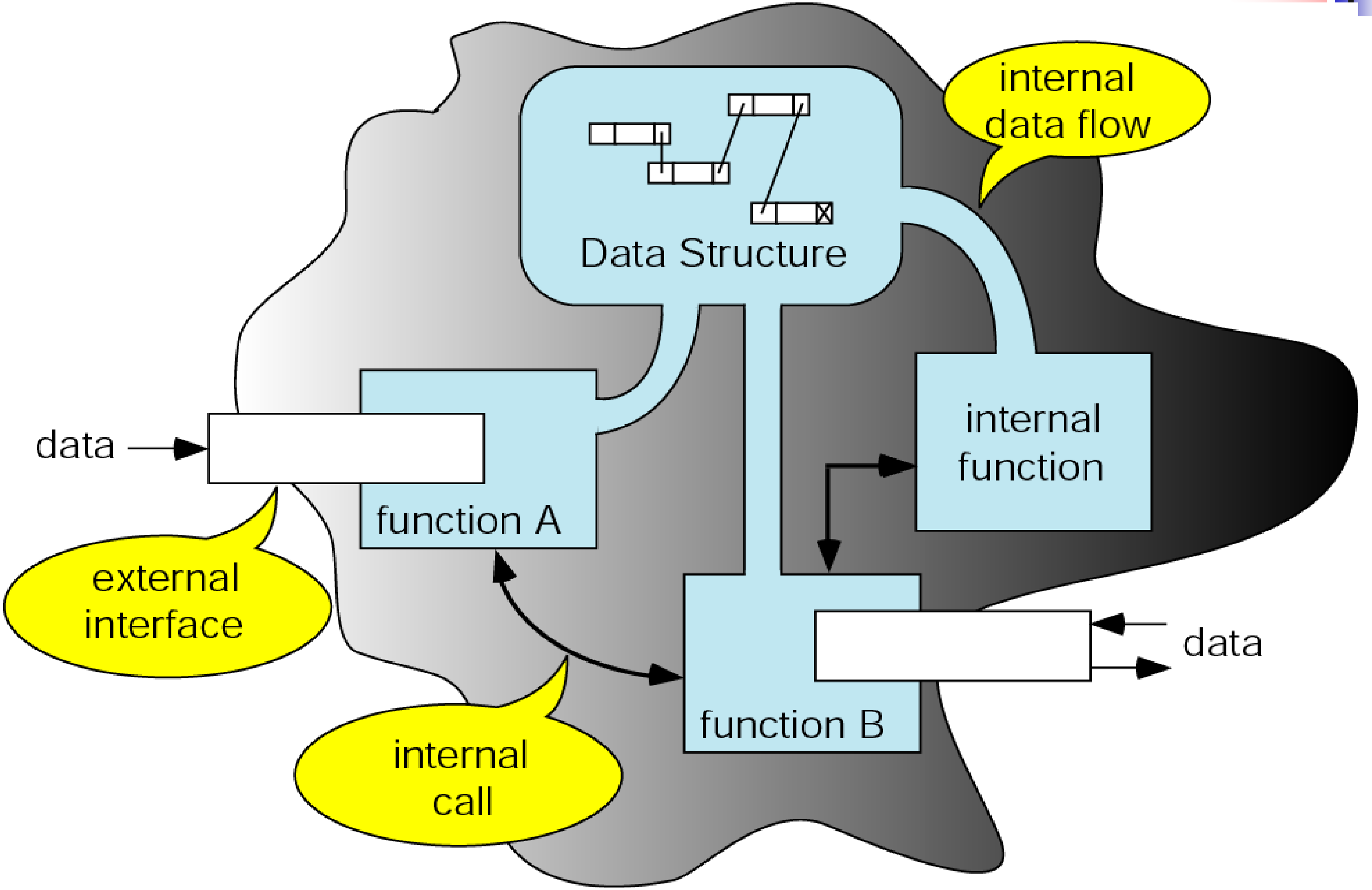
Note:

Abstract Data Type

1. Declaration of data

2. Declaration of operations

Figure 13-11 Abstract data type model


internal data flow
Data Structure
data
function A
internal function
external interface
data
internal call
function B