Chapter 3: Data Abstraction: The Walls

Data Abstraction & Problem Solving with C++

Fifth Edition

by Frank M. Carrano

Abstract Data Types

- Modularity
  - Keeps the complexity of a large program manageable by systematically controlling the interaction of its components
  - Isolates errors
  - Eliminates redundancies

Abstract Data Types

- Abstract data type (ADT)
  - An ADT is composed of
    - A collection of data
    - A set of operations on that data
  - Specifications of an ADT indicate
    - What the ADT operations do, not how to implement them
  - Implementation of an ADT
    - Includes choosing a particular data structure

ADT vs Data Structure

- ADT
  - Collection of data
  - Set of operations on the data
  - Example: list (we will define shortly)
- Data Structure
  - Construct within programming language
  - Stores a collection of data
  - Example: array

Abstract Data Types

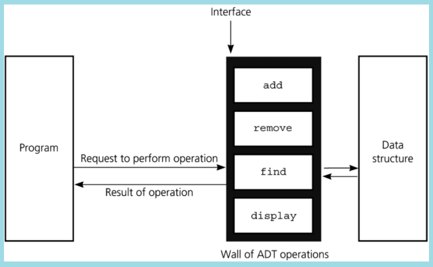


Figure 3-4

A wall of ADT operations isolates a data structure from the program that uses it

C++ Classes

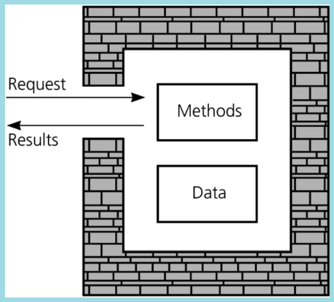


Figure 3-10

An object's data and methods are encapsulated

C++ Classes

- Each class definition is placed in a header file
  - *Classname*`.h`
- The implementation of a class's methods are placed in an implementation file
  - *Classname*`.cpp`

An Array-Based ADT List

- Both an array and a list identify their items by number
  - Using an array to represent a list is a natural choice
  - Store a list's items in an array `items`
- Distinguish between the list's length and the array's size
  - Keep track of the list's length

An Array-Based ADT List

- Header file

```
/** @file ListA.h */
const int MAX_LIST = maximum-size-of-list;
typedef desired-type-of-list-item ListItemType;
class List
{
public:
   . . .
private:
   ListItemType items[MAX_LIST];
   int          size;
}  // end List
```

An Array-Based ADT List

- A list's k^{th} item is stored in `items[k-1]`

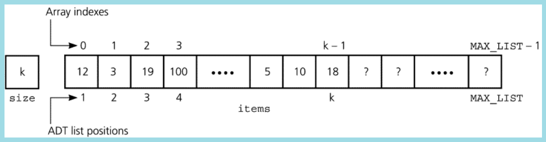


Figure 3-11 An array-based implementation of the ADT list

An Array-Based ADT List

- To insert an item, make room in the array

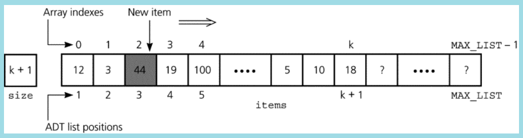


Figure 3-12 Shifting items for insertion at position 3

An Array-Based ADT List

- To delete an item, remove gap in array

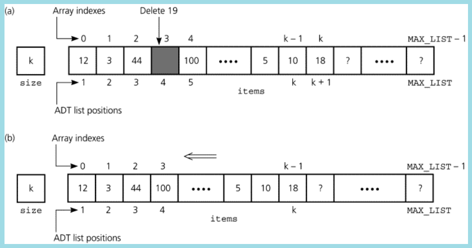
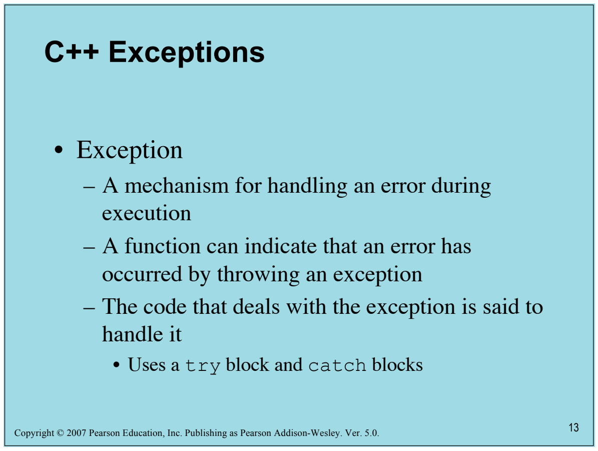
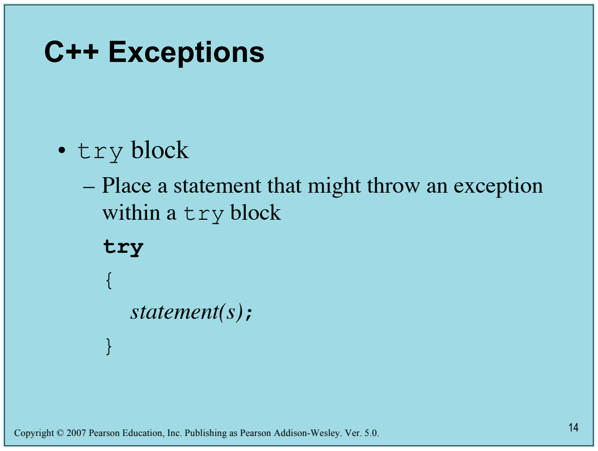


Figure 3-13 (a) Deletion causes a gap; (b) fill gap by shifting



C++ Exceptions

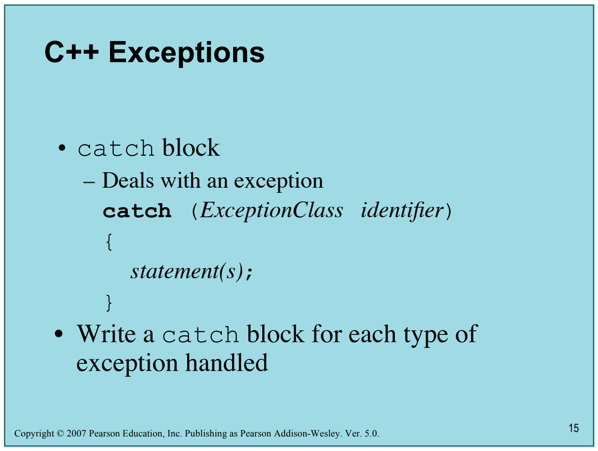
- Exception
  - A mechanism for handling an error during execution
  - A function can indicate that an error has occurred by throwing an exception
  - The code that deals with the exception is said to handle it
    - Uses a `try` block and `catch` blocks



C++ Exceptions

- `try` block
  - Place a statement that might throw an exception within a `try` block

```
try
{
   statement(s);
}
```

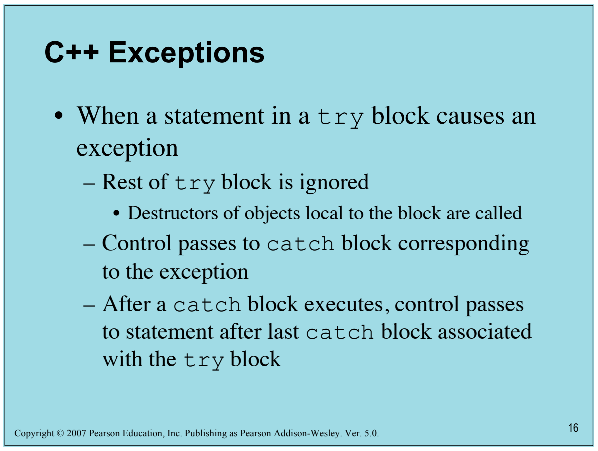


C++ Exceptions

- `catch` block
  - Deals with an exception

```
catch (ExceptionClass identifier)
{
   statement(s);
}
```

- Write a `catch` block for each type of exception handled



C++ Exceptions

- When a statement in a `try` block causes an exception
  - Rest of `try` block is ignored
    - Destructors of objects local to the block are called
  - Control passes to `catch` block corresponding to the exception
  - After a `catch` block executes, control passes to statement after last `catch` block associated with the `try` block

C++ Exceptions

- Throwing exceptions
  - A `throw` statement throws an exception

    throw *ExceptionClass* (*stringArgument*);
  - Methods that throw an exception have a `throw` clause

```
void myMethod(int x) throw(MyException)
{
  if (. . .)
     throw MyException("MyException: …");
  . . .
}  // end myMethod
```

- You can use an exception class in the C++ Standard Library or define your own

An ADT List Implementation Using Exceptions

- We define two exception classes

```
#include <stdexcept>
#include <string>
using namespace std;
class ListIndexOutOfRangeException :
      public out_of_range
{
public:
   ListIndexOutOfRangeException(const string &
                                 message = "")
                               :
     out_of_range(message.c_str())
   {}
}; // end ListException
```

An ADT List Implementation Using Exceptions

```
#include <stdexcept>
#include <string>
using namespace std;
class ListException : public logic_error
{
public:
  ListException(const string & message = "")
                      : logic_error(message.c_str())
   {}
}; // end ListException
```

An ADT List Implementation Using Exceptions

```
/** @file ListAexcept.h */
#include "ListException.h"
#include "ListIndexOutOfRangeException.h"
 . . .
class List
{
public:
   . . .
   void insert(int index,
               const ListItemType& newItem)
        throw(ListIndexOutOfRangeException,
              ListException);
   . . .
}  // end List
```

An ADT List Implementation Using Exceptions

```
/** @file ListAexcept.cpp */
void List::insert(int index,
                  const ListItemType& newItem)
           throw(ListIndexOutOfRangeException,
                 ListException);
{
   if (size > MAX_LIST)
      throw ListException("ListException: " +
                          "List full on insert");
   . . .
}  // end insert
```

Summary

- Data abstraction controls the interaction between a program and its data structures
- Abstract data type (ADT): a set of data-management operations together with the data values upon which they operate
- Axioms specify the behavior of ADT operations in a formal mathematical study of an ADT
- Define an ADT fully before making any decisions about an implementation

Summary

- Hide an ADT's implementation by defining the ADT as a C++ class
- An object encapsulates both data and operations
- A class contains one destructor and at least one constructor
- The compiler generates
  - A default constructor if no constructor is provided
  - A destructor if none is provided

Summary

- Members of a class are private by default
  - Data members are typically private
  - Public methods can be provided to access them
- Define and implement a class within header and implementation files
- Namespace: a mechanism to group classes, functions, variables, types, and constants
- You can throw an exception if you detect an error during program execution. You handle, or deal with, an exception by using `try` and `catch` blocks