Secure Coding

CHAPTER 1

Object-Oriented Programming

- 1.1 Introduction
- 1.2 Object-Oriented and Procedural Programming
- 1.3 Classes and Abstract Data Types
- 1.4 The Client/Server Model and Message Passing
- 1.5 Inheritance and Polymorphism
- 1.6 Interfaces and Components

1.1 Introduction

- **Programming Languages**

The genesis of the computer revolution was in a machine. The genesis of our programming languages thus tends to look like that machine.

1.1 Introduction

1 Machine Language:

Binary instruction, which is also called an instruction system, is the only one used directly by computers.

Example： Addition 100101

Subtraction 010011

1.1 Introduction

2 Assembly Language:

The assembler language is a kind of symbolic language. It adopted some mnemonic symbols which can show the instructional functions to present the content of the program.

Example：Addition add

Subtraction sub

1.1 Introduction

3 High-level language:

It is a programming language based on English. Its operators and expressions are similar to ordinary mathematical formulas.

Example：int a,b,c,d;

a=10; b=5; c=8;

d = a + b – c;

FORTRAN，BASIC，PASCAL，C

1.1 Introduction

4 Object-oriented programming languages: They have three characteristics in common: encapsulation, polymorphism and inheritance.

Example: Smalltalk，LISP，C++，Java，C#；

1.1 Introduction

- **Algorithms**

Algorithms are methods for solving problems which are suited for computer implementation.

1.1 Introduction

- **Data Structure**

Data structure is the type and organization of data in a program.

1.2 Object-Oriented and Procedural Programming

- **Procedural Programming**

Procedural programming is associated with a design technique known as top-down design.

In top-down design, a problem is associated with a procedure.

Example: C, Pascal.

1.2 Object-Oriented and Procedural Programming

- **Object-Oriented Programming**

Object-oriented programming is an alternative to procedural programming.

The design technique associated with object-oriented programming is object-oriented design.

In an object-oriented program, the modules are classes rather than procedures.

1.2 Object-Oriented and Procedural Programming

Aristotle was probably the first to begin a careful study of the concept of type; he spoke of “the class of fishes and the class of birds.”

In object-oriented design, a class is a collection of objects.

1.2 Object-Oriented and Procedural Programming

Objects in a class share properties, features, or attributes.

For example, if Human is a user-defined data type in C++, we can defined a variable such as maryLeakey to represent the object who belongs to the class of human beings.

1.2 Object-Oriented and Procedural Programming

In C++, the member variables or fields are called **data members**.

The functions that belong to a class are called **function members**.

In object-oriented languages generally, such functions are called **methods**.

1.3 Classes and Abstract Data Types

- **Information Hiding**

In an object-oriented language, a class is a module that supports information hiding.

In a C++ class, we can use keywords such as public and private to control access to the class's properties and operations.

1.3 Classes and Abstract Data Types

- **Encapsulation**

In procedural programming, data are manipulated by procedures with passing arguments to and returning a value from a function.

1.3 Classes and Abstract Data Types

In object-oriented programming, data and the procedures to manipulate the data can be encapsulated or contained within a class.

1.3 Classes and Abstract Data Types

- **Abstract Data Type**

A data type is abstract if it exposes in its public interface only high-level operations and hides all low-level implementation details.

For example, the classes in C++, which provide information hiding, are abstract data types.

1.4 The Client/Server Model and Message Passing

- **Client/Server Model**

Object-oriented programming is based on client/server model of computing.

This model explains the emphasis placed on information hiding in object-oriented programming.

1.4 The Client/Server Model and Message Passing

Example:

```
#include <string>
using namespace std;
int main()
{
    string s1="Hello World! ";
    int n=s1.length();
    return 0;
}
```

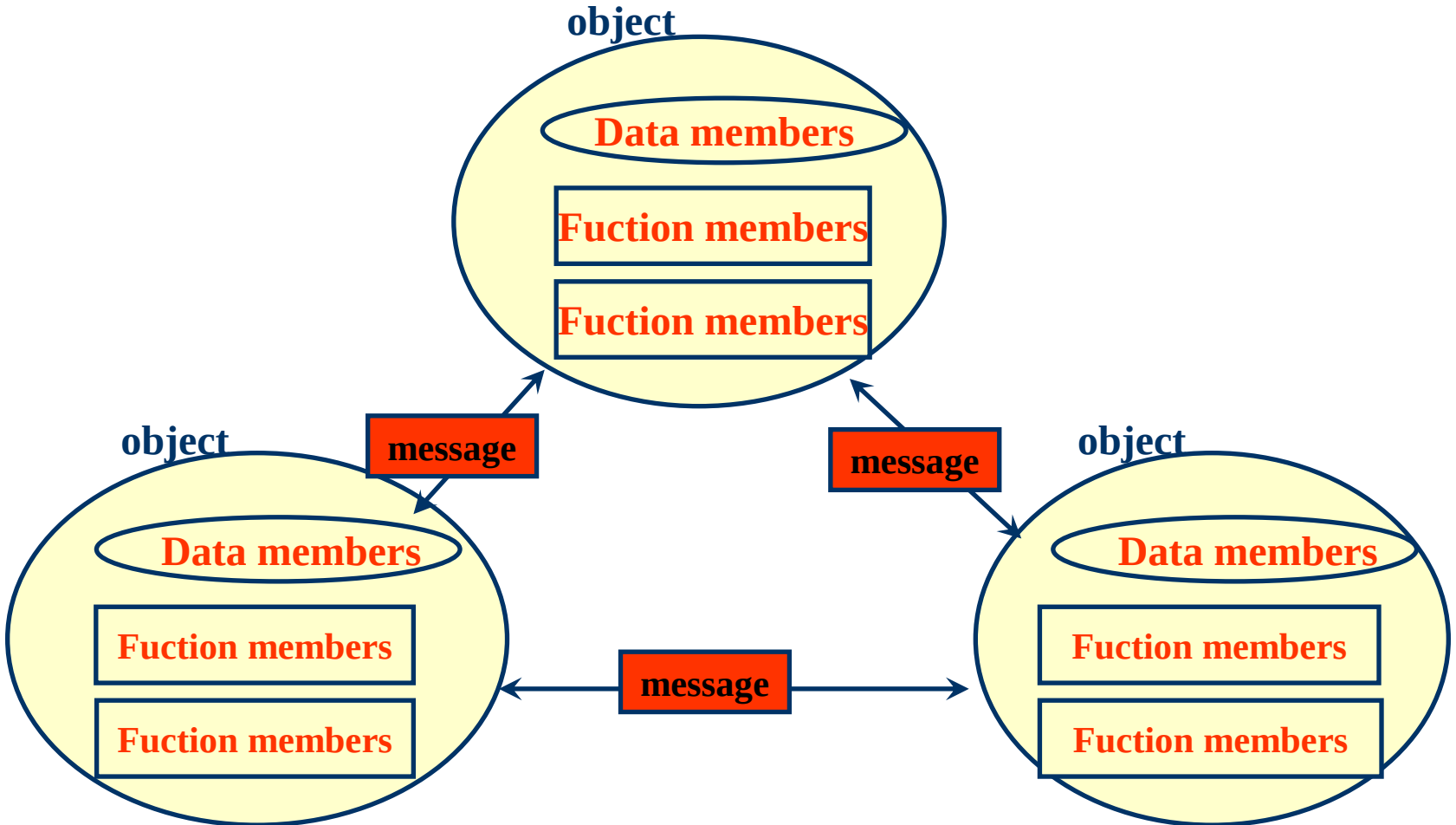
The C++ program that uses the string class is a client. A client requests services from a string object by invoking one of its methods, which is characterized as sending a message to the object.

1.4 The Client/Server Model and Message Passing

- **Message Passing**

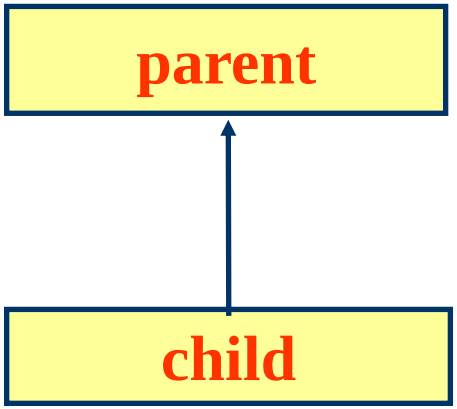
```
#include <string>
using namespace std;
int main()
{
    string s1="Hello World! ";
    int n=s1.length();
    return 0;
}
```

1.4 The Client/Server Model and Message Passing



1.5 Inheritance and Polymorphism

- Classes can be stand-alone or they can occur in inheritance hierarchies, which consist of parent/child relationships among classes.



1.5 Inheritance and Polymorphism

- **Inheritance**

One of the most compelling features about C++ is code reuse.

Inheritance supports a form of code reuse.

1.5 Inheritance and Polymorphism

- **Polymorphism**

The term polymorphism means having many forms.

Polymorphism allows improved code organization and readability as well as the creation of extensible programs that can be "grown" not only during the original creation of the project, but also when new features are desired.

1.6 Interfaces and Components

- **Interfaces**

Interfaces, if well built and shared among classes, are a convenience to programmers.

Learning a shared interface allows a programmer to request services, using the same methods, from a variety of classes.

1.6 Interfaces and Components

- **Components**

A software component is a prebuilt part that can be combined with other such parts to build an application.

A components must expose at least one interface, a component typically expose several interfaces.

From user's viewpoint, a component is the back end of some front-end interface and need not know even the language in which it is written.

Summarize

- Object-Oriented and Procedural Programming
- Classes and Abstract Data Types
- The Client/Server Model and Message Passing