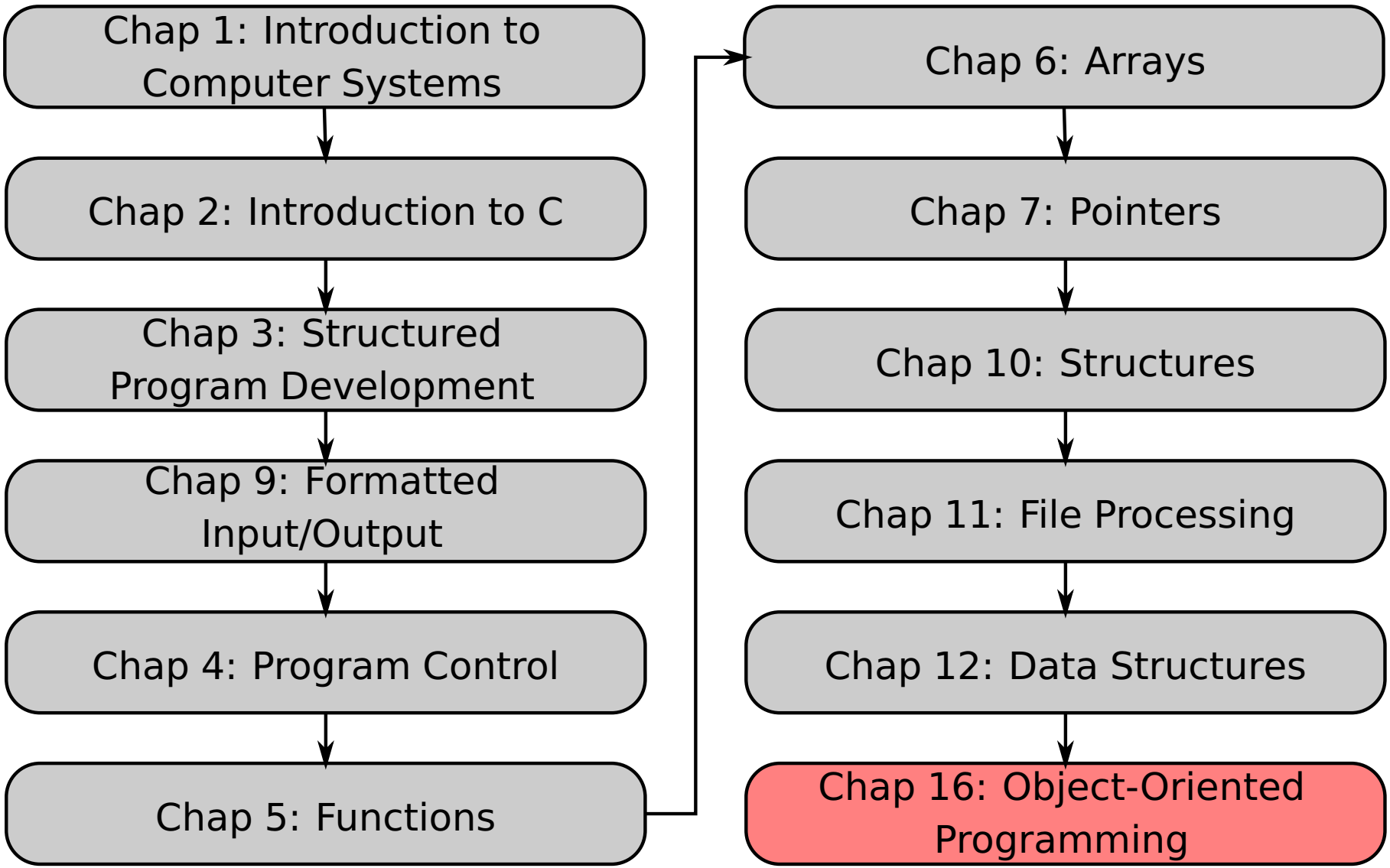
Computer Programming 143 – Lecture 31
Object-Oriented Programming I

Electrical and Electronic Engineering Department
University of Stellenbosch

Prof Johan du Preez
Mr Callen Fisher
Dr Willem Jordaan
Dr Hannes Pretorius
Mr Willem Smit

Module Overview



Copyright & Disclaimer

Lecture Overview

1. Introduction (15.1-15.2)
2. The C++ Language (15.3-15.13)
3. Basic Object Technology Concepts (15.14)
4. Class Analogy (16.2)
5. Class with a Member Function (16.2)

15.1-15.2 Introduction

C++

- Chapters 15-24 cover C++
- C++ is an improved C with:
  - object-oriented programming (OOP) capabilities
  - generic programming capabilities
- Makes software more reusable and higher quality

What is our focus?

- Introduction to OOP concepts
  - Classes, encapsulation and objects

What do we omit?

- Advanced OOP and generic programming concepts
  - Operator overloading, inheritance, polymorphism, class and function templates
- C++ specific syntax

15.3-15.13 The C++ Language

New syntax

- C syntax can be used as is in C++
- C++ introduces new syntax for many actions
- Writing to screen, reading from keyboard, standard libraries, function arguments

Our focus

- Concepts, not syntax
- We keep to known C syntax as far as possible
- Only introduce new syntax where necessary

15.14 Introduction to object technology

Real world objects

- Real world consists of objects
- Objects (eg. a ball) posses

  Attributes: colour, shape, size, weight

  Behaviour: bounce, roll, inflate, deflate

Object-oriented design

- Software design uses familiar concepts
- Intuitive: Focus is on things (objects) rather than actions (functions)
- Allows for reuse of object attributes and behaviour
- Encapsulates attributes and behaviour into objects

15.14 Reuse of Object Attributes and Behaviour

Class relationships

- Similar classes of objects can share properties
- E.g. *car*, *truck*, and *train* share *seat*, *steering wheel*, *wheels*

Inheritance relationships

- New class of objects easily created
- Absorb existing properties of similar class
- Only add unique characteristics

15.14 Classes, Data Members and Member Functions

Classes

- C basic program building block is the function
- C++ basic building block is the class
- Class contains data and functions that operate on the data
- Class is a more self contained building block
- Improves software re-use

Data members and member functions

- Data inside class = data members
- Functions inside class = member functions (also called methods)
- Data members give a class attributes
- Member functions give behaviour

16.2 Class Analogy

Motor car

- Accelerator, steering wheel
- User steps on accelerator
- Hides mechanisms
- One design drawing
- Many cars built
- All possess the same attributes
- Values of attributes unique to each

C++ Class

- Member functions
- User calls member functions
- Interface encapsulates detail
- One class definition
- Create many objects
- Data members determined by class definition
- Contents unique to each object

Compare C function with C++ Class

Why is a class a better building block than a function?

C function

- Gets information as input parameters or global variables
- Prevents effective re-use of building block:
  - *Parameters:* Many parameters hamper readability
  - *Global variables:* Readability improves, but falls outside of building block

C++ Class

- Class functions have access to all data members
- Data members are like static variables: retain values between function calls
- Fewer parameters are passed (data part of building block)
- Hides inner working: encapsulates, easily re-usable

Compare C struct with C++ class

A Class can be seen as a struct that also contains member functions.

C struct

- Different data members
- Is just a definition
- Requires declaration of variables before use

C++ Class

- Different data members and member functions
- Is also just a definition
- Requires declaration of objects before use

16.2 Class with a Member Function: Code

```
#include <stdio.h> /*we use standard C printf*/
class GradeBook   /*GradeBook class definition*/
{
public:
   /*function that displays a welcome message to the GradeBook user*/
   void displayMessage()
   {
      printf("Welcome to the Grade Book!\n");
   } /*end function displayMessage*/
}; /*end class GradeBook*/

int main()
{
   GradeBook myGradeBook; /*create an object from the class*/
   myGradeBook.displayMessage(); /*call object's member function*/
   return 0;
} /*end main*/
```

16.2 Class with a Member Function: Explanation

1) Class Definition — class GradeBook{}

- Body of class follows inside brackets {}
- **void displayMessage()** — member function
- **public:** — makes member function visible for class users (main)
- Only one definition of a class

2) Create objects of class — GradeBook myGradeBook;

- Class is like design specification
- Must build object before use
- Can build many different objects from one class definition

3) Use objects — myGradeBook.displayMessage()

- Dot operator: object.memberFunction()
- Use object's member functions and data members

Perspective

Today

Object-Oriented Programming I

- Introduction
- Classes, objects, member functions and member data
- Defining a class with a member function

Next lecture

Object-Oriented Programming II

- Member functions with arguments
- Member data

Homework

1. Study Sections 15.1-15.2, 15.14, 16.1-16.2 in Deitel & Deitel