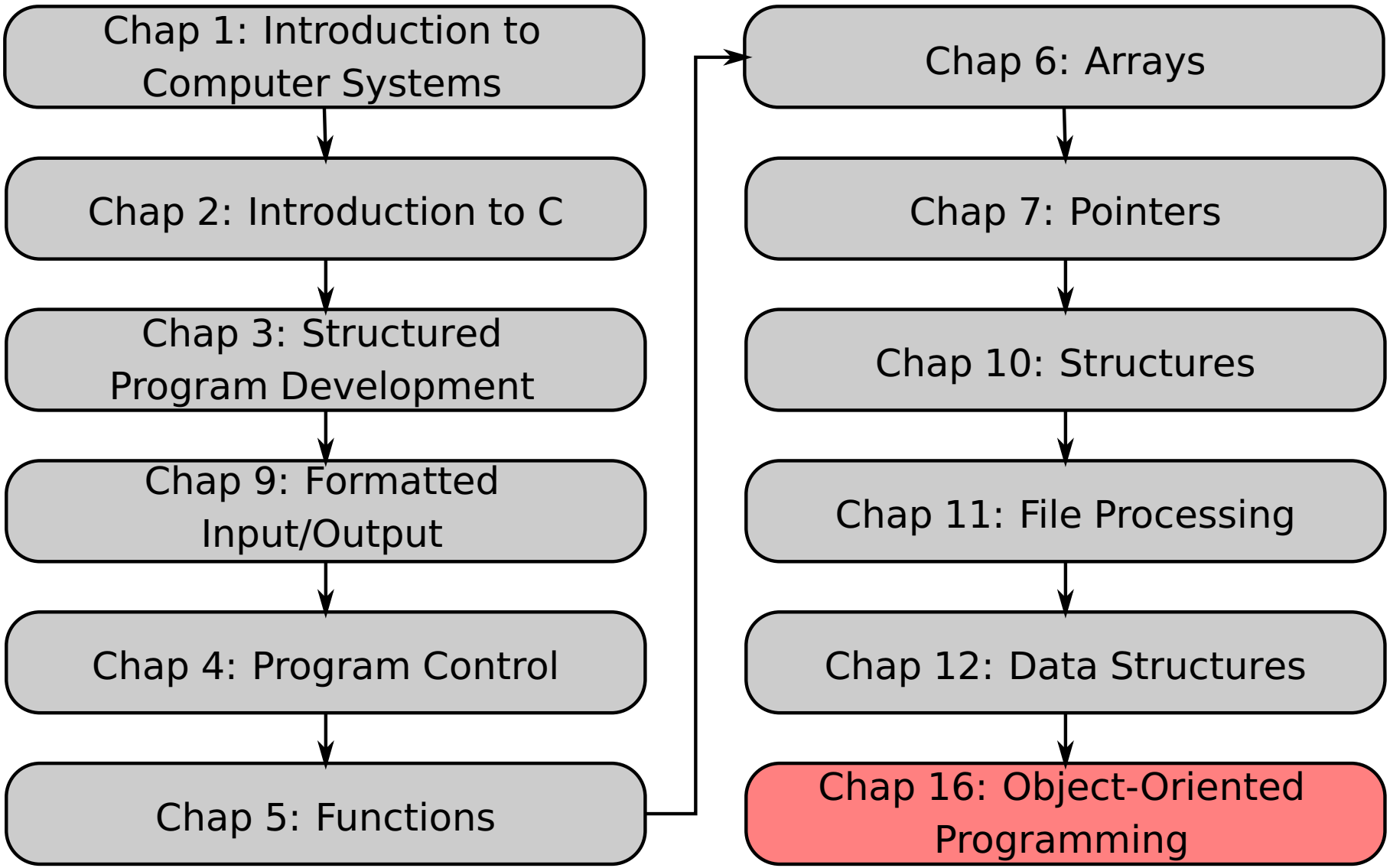
# Computer Programming 143 – Lecture 32
# Object-Oriented Programming II

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. Review: Classes, Data Members and Member Functions (15.14)
2. Member Functions with Parameters (16.3)
3. Introducing Data Members (16.4)
4. Programming: Looking Back
5. Programming: Looking Forward

# 15.14 Review: Classes, Data Members and Member Functions

## Classes

- C++ basic building block is the class
- Class contains data and functions that operate on the data

## Objects

- Instance (“variable”) of a class (“data type”)
- Class is a template from which an object is created

## 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

# 16.3 Member Functions with Parameters

```
#include <stdio.h> /*we use standard C printf*/

class GradeBook   /*GradeBook class definition*/
{
public:
   /*function displays courseName received as parameter*/
   void displayMessage(char courseName[20])
   {
      printf("Welcome to course %s!\n",courseName);
   } /*end function displayMessage*/
}; /*end class GradeBook*/

int main()
{
   GradeBook myGradeBook; /*create an object from the class*/
   /*call member function with parameter*/
   myGradeBook.displayMessage("RP143");
   return 0;
} /*end main*/
```

# 16.4 Introducing Data Members

```
#include <stdio.h>
// GradeBook class definition
class GradeBook
{
public:
   // member function that sets the course name
   void setCourseName( char name[] )
   {  // write to object's courseName member
      sprintf(courseName,"%s",name);
   }
   // member function that displays a welcome message
   void displayMessage()
   {
      printf("Welcome to the grade book for\n %s !\n",courseName);
   }
private:
   char courseName[40]; // datamember: course name in this GradeBook
}; // end class GradeBook
```

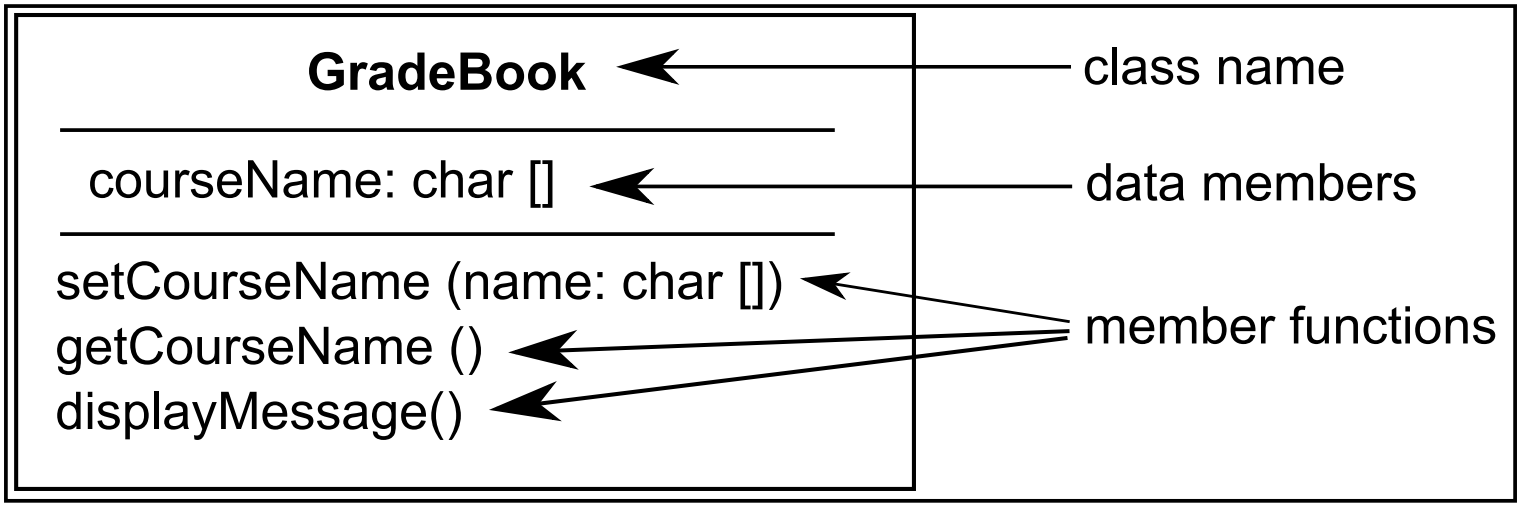
# 16.4 Introducing Data Members

```
int main()
{
   char nameOfCourse[40]; // store the course name in main()
   GradeBook firstGradeBook, secondGradeBook; // create objects

   printf("Please enter the first course's name (no spaces):\n");
   scanf("%39s",nameOfCourse);
   //use member function to write to object's data member
   firstGradeBook.setCourseName( nameOfCourse );
   // display message with new course name
   firstGradeBook.displayMessage();

   printf("Please enter the second course's name (no spaces):\n");
   scanf("%39s",nameOfCourse);
   secondGradeBook.setCourseName( nameOfCourse );
   secondGradeBook.displayMessage();
   return 0; // indicate successful termination
} // end main
```

# 16.4 Class Definition: Explanation

## Class body expanded (from Fig.16.5 in D & D)

- Class now contains 3 member functions — `setCourseName()`,`getCourseName()` and `displayMessage()`
- Data member declared outside member functions but inside class body
- Class contains 1 data member — courseName
- Member functions can access object's data member



# 16.4 Access specifiers

## General use

- **private:** makes members inaccessible outside the object
- Limits user's access to prevent errors
- Both data members and member functions can be either `private` or `public`

## Good practice

- Make data members `private`, member functions `public`
- User affects `private` data members through `public` member functions
- Indirect access promotes maintainability

# 16.4 Class use: Explanation

## Create objects of the class

- Create two objects, `firstGradeBook, secondGradeBook`
- Each object has its own copy of the `courseName` data member
- setCourseName() assigns a unique name to each

```
GradeBook firstGradeBook, secondGradeBook; // create objects

firstGradeBook.setCourseName( nameOfCourse );
firstGradeBook.displayMessage();
```

# Programming: Looking Back

## What have we learned?

- Solving engineering problems by designing structured, procedural algorithms
- C: syntax, control structures, functions and libraries, data and file structures and handling

# Programming: Looking Forward I

## Tiobe Software Index (www.tiobe.com) - September 2020

| Position Sep 2020 | Programming Language | Ratings Sep 2020 | Delta Sep 2020 |
|---|---|---|---|
| 1 | C | 15.95% | 0.74% |
| 2 | Java | 13.48% | -3.18% |
| 3 | Python | 10.47% | 0.59% |
| 4 | C++ | 7.11% | 1.48% |
| 5 | C# | 4.58% | 1.18% |
| 6 | Visual Basic | 4.12% | 0.83% |
| 7 | JavaScript | 2.54% | 0.41% |
| 8 | PHP | 2.49% | 0.62% |
| 9 | R | 2.37% | 1.33% |
| 10 | SQL | 1.76% | -0.19% |

# Programming: Looking Forward II

## Tiobe Software Index (www.tiobe.com) - September 2020

| Position | Programming Language | Ratings | Delta |
|---|---|---|---|
| **Sep 2020** | | **Sep 2020** | **Sep 2020** |
| 11 | Go | 1.46% | 0.24% |
| 12 | Swift | 1.38% | 0.28% |
| 13 | Perl | 1.30% | 0.26% |
| 14 | Assembly language | 1.30% | -0.08% |
| 15 | Ruby | 1.24% | 0.03% |
| 16 | MATLAB | 1.10% | 0.04% |
| 17 | Groovy | 0.99% | -0.52% |
| 18 | Rust | 0.92% | 0.55% |
| 19 | Objective-C | 0.85% | -0.99% |
| 20 | Dart | 0.77% | 0.13% |

# Perspective

## Today

Object-Oriented Programming II

- Class with member function that takes arguments
- Class with member functions and data members

## Next

Prepare for A2

# Homework

1. Study Sections 16.3-16.4 in Deitel & Deitel