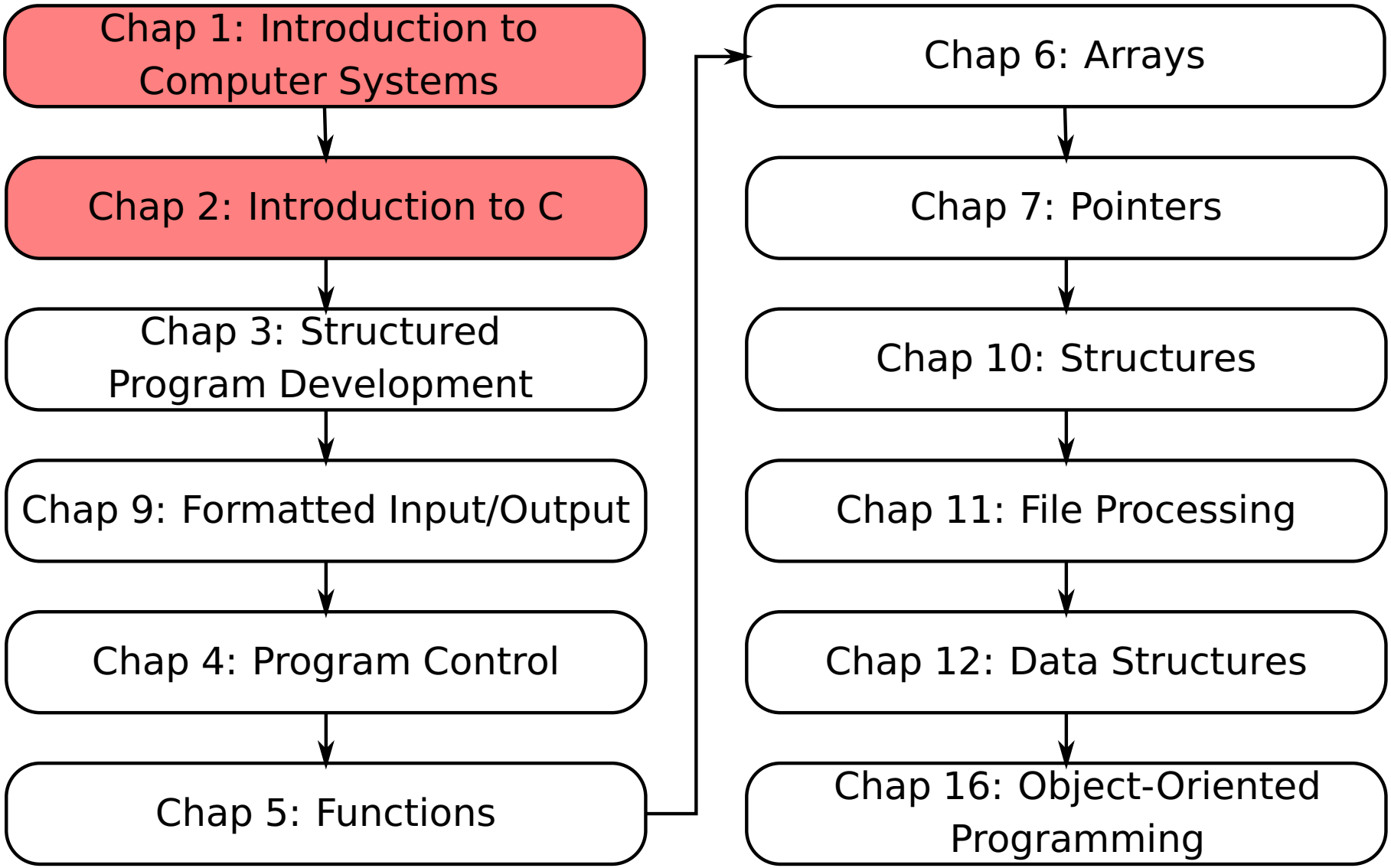
Computer Programming 143 – Lecture 2
Introduction to module

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



Lecture Overview

1. Introduction (1.1)
2. Computer Systems (1.2)
3. Programming Languages (1.4-1.9)
4. Simple C Program 1: Printing Text (2.2)

1.1 Introduction

We will learn

- How to think in a structured way in order to solve problems using computers
- The C programming language
- Proper programming techniques

The textbook also covers

- C++: Chapters 15 to 24 introduce the C++ programming language

This course is appropriate for

- Technically oriented people with little or no programming experience
- Experienced programmers who want a deep and rigorous treatment of the C language

1.2 Computers: Hardware and Software

Computer

- Device capable of performing computations and making logical decisions
- Computers process data under the control of sets of instructions called computer programs

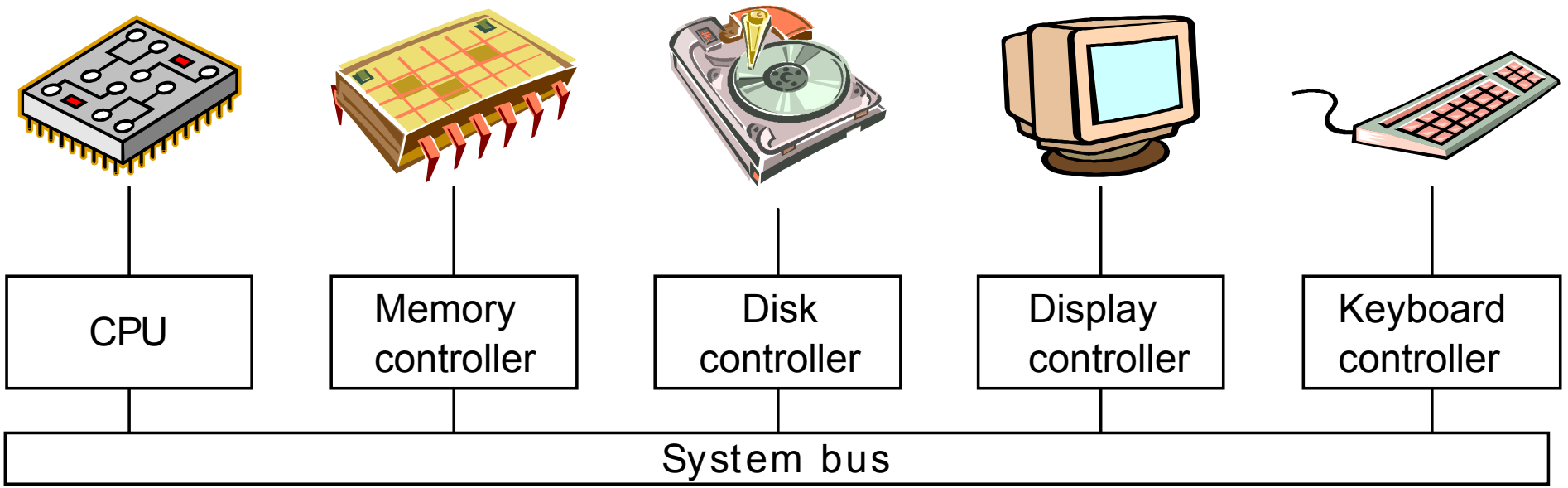
Hardware

- A computer consists of various devices
- Keyboard, screen, mouse, disks, memory, CD-ROM, and processing units

Software

- Programs that run on a computer

1.2.2 Computer Organization (Hardware) I



1.2.2 Computer Organization (Hardware) II

Six logical units in every computer:

1. Input unit (Keyboard, mouse)
   - Obtains user input information from input devices
2. Output unit (Screen, printer)
   - Outputs user or processed information and results
3. Arithmetic Logic Unit (ALU)
   - Performs arithmetic calculations and logic decisions
   - Implemented as part of CPU (below)
4. Central processing unit (CPU)
   - Supervises and coordinates the other sections of the computer
5. Memory unit (RAM)
   - Volatile, rapid access, low capacity, and expensive
   - Stores program and input information temporarily
6. Secondary storage unit (Disks)
   - Cheap, long-term, high-capacity storage
   - Stores inactive programs and data

1.4 Machine Languages, Assembly Languages, and High-level Languages I

Three types of programming languages:

1. Machine languages
   - Strings of numbers giving machine specific instructions
   - Example:

```
+1300042774
+1400593419
+1200274027
```

2. Assembly languages
   - English-like abbreviations representing elementary computer operations (translated via assemblers)
   - Example:

```
LOAD    BASEPAY
ADD     OVERPAY
STORE   GROSSPAY
```

1.4 Machine Languages, Assembly Languages, and High-level Languages II

3. High-level languages
   - Codes similar to everyday English
   - Use mathematical notations
   - Translated via compilers
   - Example:

```
grossPay = basePay + overTimePay
```

This course: C

Why do we use C in this module?

- Programming principles are easy to learn in C
- Provides a good base from which to learn higher level languages (C++/Java)
- Essential for embedded applications
- Many libraries for mathematical and scientific applications available

1.5 History of C

C

- Evolved by Ritchie from two previous programming languages, BCPL and B
- Used to develop UNIX
- Used to write modern operating systems
- Hardware independent (portable)
- By late 1970's C had evolved to "Traditional C"

Standardization

- Many slight variations of C existed, and were incompatible
- Committee formed to create a "unambiguous, machine-independent" definition
- Standard created in 1989, updated in 1999 (C99) and 2011 (C11)

1.6 The C Standard Library

C programs consist of pieces/modules called functions

- A programmer can create his own functions
  - Advantage: the programmer knows exactly how it works
  - Disadvantage: time consuming
- Programmers will often use the C library functions
  - Use these as building blocks
  - Advantages: saves time
  - Disadvantages: must know exactly how the library work
- Avoid re-inventing the wheel
  - If a pre-made function exists, generally best to use it rather than write your own
  - Library functions carefully written, efficient, and portable

1.9 Typical C program development environment I

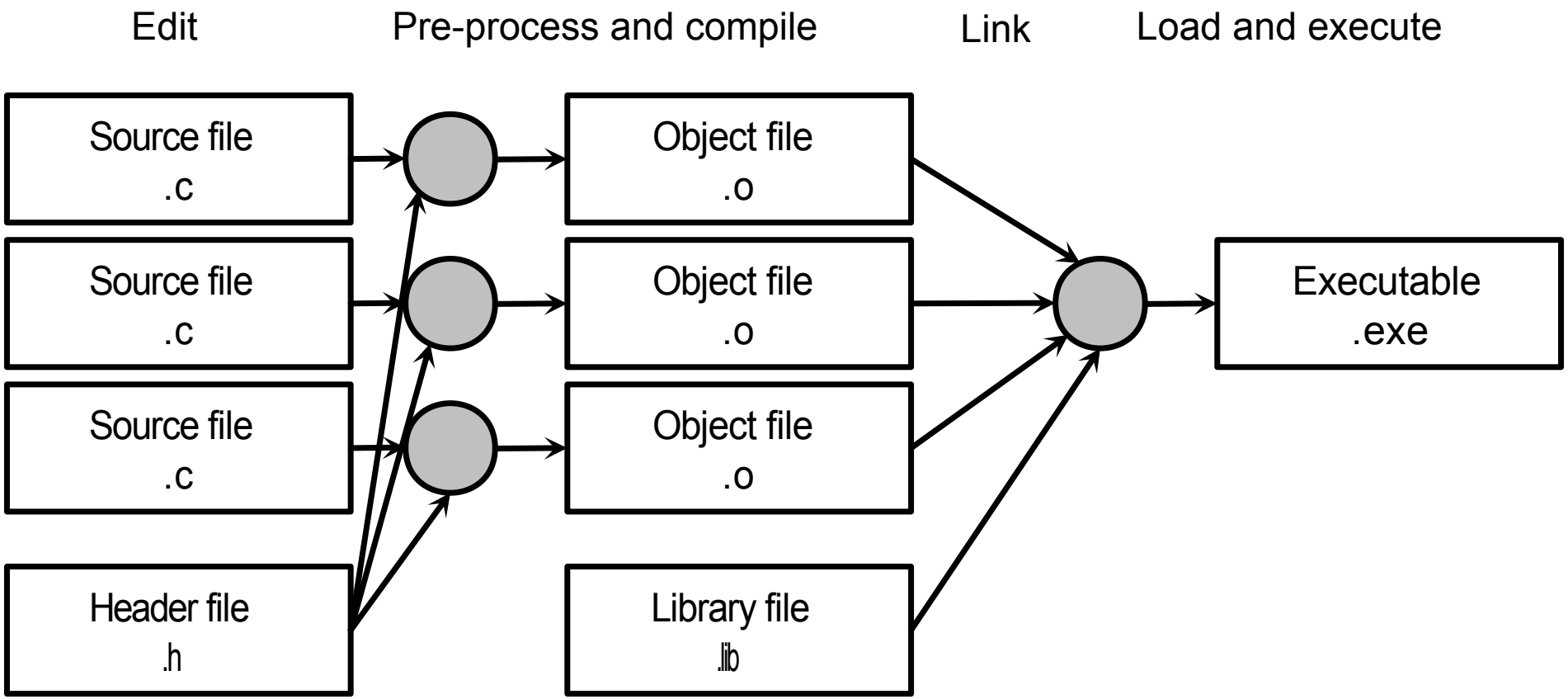
Typical C systems consists of many parts, they include:

- a program development environment
- the language
- the C Standard Library

C programs typically go through six phases to be executed:

- edit
- pre-process
- compile
- link
- load
- execute

1.9 Typical C program development environment II



2.2 Simple C Program: Printing Text I

Fig 2.1 Text printing program

```c
/* filename: helloworld.c
 * description: My first program in C
 * version: 3
 * date: 10/02/2004
 * author: RG and DE
 */
#include <stdio.h>

/* function main() begins program execution */
int main()
{
   printf( "Hello World!\n" );

   return 0; // program ended successfully
}
```

2.2 Simple C Program: Printing Text II

Output

```
Hello World!
```

2.2 Simple C Program: Printing Text III

Comments

- Used to describe program
- Text surrounded by /* and */ is ignored by compiler

```
/* filename: helloworld.c
 * description: My first program in C
 * version: 3
 * date: 10/02/2004
 * author: RG and DE
 */

/* function main() begins ... */
```

- Remainder of line after // is ignored by compiler

```
   return 0; // program ended successfully
```

2.2 Simple C Program: Printing Text IV

`#include <stdio.h>`

- Pre-processor directive
  - Tells the computer to include the contents of the specified file with the source code
- `<stdio.h>` allows standard input/output operations

`int main()`

- C programs always contain one or more functions, exactly one of which must be main
- Parenthesis used to indicate a function
- **`int`** means that main “returns” an integer value to calling function or batch process

2.2 Simple C Program: Printing Text V

Braces { and }

- Braces ({ and }) indicate a block
- The bodies of all functions must be contained in braces

```
{
   //Block contents
}
```

printf("Hello World!\n");

- Instructs computer to perform an action
  - Specifically, prints the string of characters within quotes (" ")
- Escape character (\)
  - Indicates that `printf` should do something out of the ordinary
  - \n is the newline character
- Entire line called a statement
  - All statements must end with a semicolon (;)

2.2 Simple C Program: Printing Text VI

return 0;

- A way to exit a function
- `return 0`, in this case, means that the program terminated normally

Perspective

Today

Introduction to Computer Systems and Programming

- Introduction (1.1)
- Computer Systems (1.2)
- Programming Languages (1.4-1.9)

Introduction to C

- First program (2.2)

Next lecture

Introduction to C continued

- Arithmetic, user input

Homework

1. Read Chapter 1
2. Do Self-Review Exercises 1.1, 1.2
3. Do Exercises 1.4, 1.5
4. Go through first program 2.2