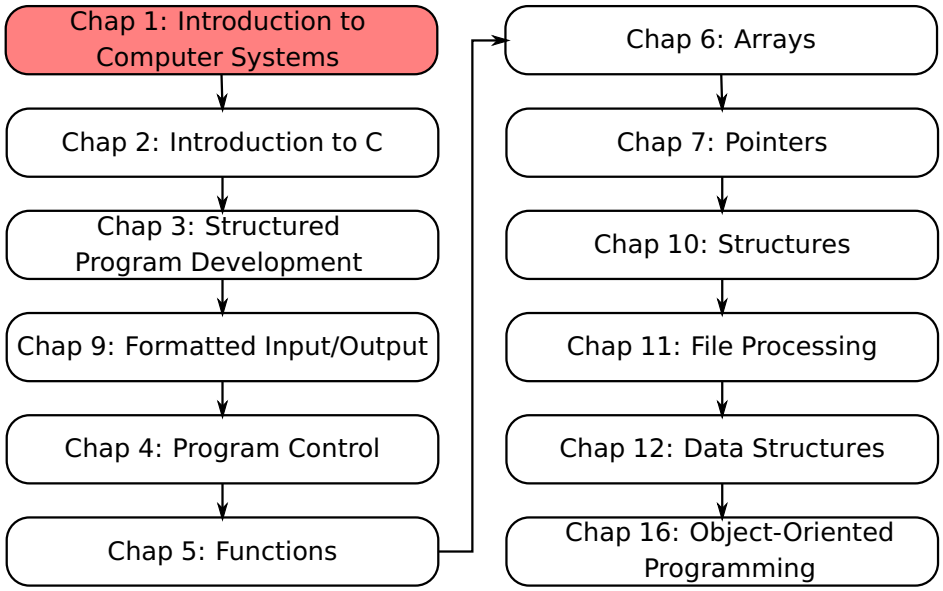
Computer Programming 143 – Lecture 2
Introduction to module

Electrical and Electronic Engineering Department
University of Stellenbosch

19 July 2016

Corné van Daalen
Thinus Booysen
Willie Smit
Willem Jordaan

Module Overview



Lecture Overview

1.1 Introduction

We will learn

- The C programming language
- Structured programming and proper programming techniques

This book also covers

- C++: Chapter 15 – 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 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

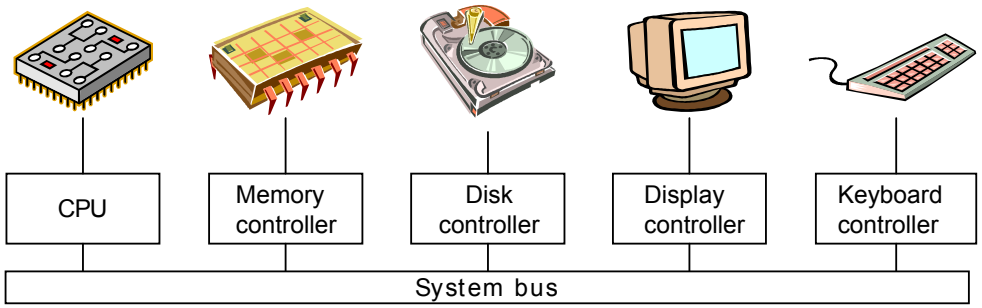
Hardware

- 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 I



1.2.2 Computer Organization 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
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
```


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

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

Fortran, COBOL, Pascal en Ada (Structured Programming)

In the 1960's most software development companies were experiencing major problems during development

- Projects were never completed on time
- Projects always went over budget
- Final products were never reliable

- Research lead to the development of structured programming

What is structured programming?

- It is a disciplined approach to programming where programs can be:
  - more reliable
  - more understandable
  - provable as correct
  - reused

- Pascal (1971) is a language that was developed directly from the research into structured programming

1.8 Key software trend: Object technology I

Writing Code as Objects

- started in the 1990's
- simplified the development of code "objects"
- closely resembles real world objects
- improved code re-usability
- eliminates the process of having to "start from scratch"
- produces code that is easier to understand and maintain

1.9 Typical C program development environment I

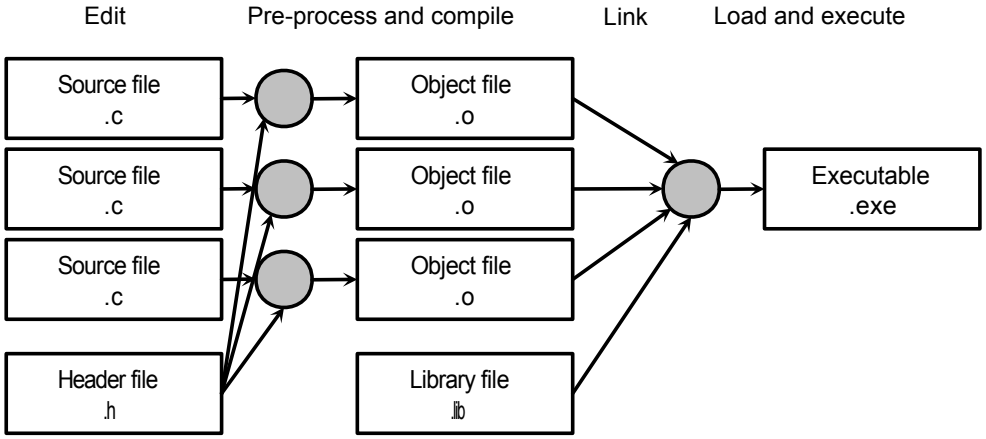
Typical C systems consists of many parts, they include:

- 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



Why 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

Perspective

Today

Introduction to Computer Systems and Programming

- Introduction (1.1)
- Computer Systems (1.3-1.4)
- Programming Languages (1.5-1.10)
- Why C?

Next lecture

Introduction to C

- First program

Homework

1. Read Chapter 1
2. Do Self-Review Exercises 1.1, 1.2
3. Do Exercises 1.4, 1.5