COMP1005
Programming and Algorithms

C Basics

Jamie Twycross

Overview

- The C programming language
- Writing and compiling C
- C language standards

The C Programming Language

C Background

- C is a **general-purpose**, **high-level**, **imperative**, **procedural** programming language
- Invented by **Denis Ritchie** (AT&T Bell Labs) around 1972
- Developed out of the B language
- The C Programming Language book written by **Brian Kernighan** and Dennis Ritchie in 1978
- Still ranks as one of the most **widely used** languages:

  `https://www.tiobe.com/tiobe-index/`
- Linux, Windows, Mac kernels written in C
- **Derivatives**: C++, C#, Java, PHP, Perl, Rust, ...

  `https://en.wikipedia.org/wiki/List_of_C-family_programming_languages`

C Features

- C provides a minimal set of inbuilt **constructs**
- More complex tasks handled by **functions**:

  `printf()`, `exit()`, `getchar()`, `fopen()`, ...
- **Standard libraries** (`stdio.h`, `strings.h`, ...) of functions provided to do common tasks
- Create new libraries to do specific tasks
- C instructions can be **efficiently** converted in to machine code
- C offers ways of interfacing with underlying hardware (e.g. pointers)

Programming in C

Writing and Compiling C

- A C **source program** is an ASCII text file(s)
- Use any **text editor** to write a C program: gedit, kwrite, vim, emacs, notepad++, ...
- **Compile** the text **source** program into a binary **executable** program
- In COMP1005, use the **GNU gcc compiler** in a *nix shell
- Can also use an **Integrated Development Environment** (IDE) to edit and compile: Eclipse, Code::Blocks, NetBeans
- Choose a development environment (editor/IDE) which suits your tastes and allows you to program **efficiently**:
  - syntax highlighting
  - autocompletion
  - keyboard shortcuts

Compiling C

- **Basic** usage:

```
$ gcc source.c
```

  - outputs executable `a.out`
- **Better** usage:

```
$ gcc -o myprog source.c
```

  - use the -o **flag** to specific output **executable name**
  - outputs executable `myprog`
- **Standard** usage:

```
$ gcc -Wall -o myprog source.c
```

  - use the -Wall **flag** to show all **compiler warnings**
- compilation **continues** after **warnings**
- compilation **stops** on **errors**

Compiling C in COMP1005

```
$ gcc -Wall -ansi -pedantic-errors -o myprog source.c
```

C Language Standards

Language Standards

- Part of the job of a compiler is to **implement** a programming language
- Programming languages are not static - **evolve** over time
- Often different implementations of a single programming language - **dialects**
- Different implementations provide different **language features**
- A **programming language standard** clearly defines the set of language features
- Standards are important - code **portability**

C Language Standards

- C has several standards:
  - K&R C (1978)
  - ANSI C / C89 (1989)
  - ISO C / C90 (1990)
  - ISO C99 (1999)
  - ISO C11 (2011)
  - ISO C18 (2018)
- C standards apply to **language constructs** and **standard libraries**
- Compilers also often offer their own **non-standard extensions**
- gcc provides a number of non-standard extensions

Controlling gcc Standards

- Use `-std` or `-ansi` flag with gcc to specify the standard to use:

```
$ gcc -std=c89 -o myprog source.c
```

```
$ gcc -ansi -o myprog source.c
```

- Use `-pedantic` flag to switch off gcc extensions:

```
$ gcc -pedantic -o myprog source.c
```

```
$ gcc -pedantic-errors -o myprog source.c
```

The Standard for COMP1005

- We use **ANSI C (C89)/ISO C (C90)** in this module

```
$ gcc -Wall -ansi -pedantic-errors -o myprog source.c
```

- All coursework and lab exercises will enforce this
- Some implications for your source code:
  - comments - no //
  - some standard library functions not available e.g. `snprintf()`

Summary

- Overview of the C programming language:
    - history
    - features
- Writing and compiling C:
    - text editors/IDEs
    - compilers - gcc
- C language standards:
    - ANSI C (C89)/ISO C (C90)

Activities

- Read the **Preface** and **Introduction** to your K&R course book
- Have a look at the Wikipedia page for the C Programming Language:

  `https://en.wikipedia.org/wiki/C_programming_language`
- You do not need to understand everything on the Wikipedia page now