# C Programming for MSc
## ELE00107M

# Lecture 1: Introduction

*Prof. Stephen Smith*

E-mail: stephen.smith@york.ac.uk

# Why learn to program?

- A programming language is used for instructing computers to solve problems using *algorithms*

- High level languages use *statements* to express a required operation. They are easier to construct and resemble English
  - **area = PI * radius * radius;**

- Assembler (low level) represents machine code operations and data with mnemonics: processor specific
  - **SUB AX,BX**
  - **MOV CX,AX**
  - **MOV DX,0**

- Machine code uses primitive operations (lowest level)
  - purely binary codes: processor specific
  - **0010101111000011**
  - **1000101111001000**
  - **10111010000000000000000**

# Algorithms: numbered steps
## Example: Calculating the Mean

1. allocate memory for `sum, number,count,how_many`
2. set `sum` to zero
3. get `how_many`
4. set `count` to `how_many`
5. while `count` is not `zero` do
    1. get `number`
    2. add `number` to `sum`
    3. decrement `count by one`
6. if `how_many` is not `zero` then
7. print `sum/how_many;`

# Algorithms: described as *pseudocode*

## Example: Calculating the average

```
Allocate memory for sum, how_many, number;
{
   Set sum to zero
   Input how_many numbers to average
   Do how_many times
      Input number
      Add number to sum
   If how_many is Not zero
     Output sum / how_many
}
```

# Algorithms: coded in C
## Example: Calculating the Mean

```
#include <stdio.h>

int main(void)
{
    /* allocate memory */
    double number,sum;
    int count;
    int how_many;

    /* intialise sum and count*/
    sum = 0.0;
    count = 1;

    printf("\nHow many numbers?: ");
    scanf("%d", &how_many);

    while (count <= how_many)
    {
        printf("\nEnter number: ");
        acanf("%lf",&number);
        sum = sum + number;
        count++;
    }
    if (how_many > 0)
        printf("\nThe average is %6.2lf", sum/how_many);

    return 0;
}
```

# Why learn C?

- General purpose
- Widely used
- Lots of embedded C code in electronic systems
- Many hardware devices are programmed in C (e.g. PICs, FPGAs)
- Basis of many other programming language syntax:
  - e.g. C++, C#, Java
- Fast – good for numerical computing
- Good for your CV

# Top Paying and Most Popular Programming Languages in 2020

**Top Paying and Most Popular Programming Languages in 2020**

| Rank by Average Salary | |
|---|---|
| **1.** Python | $119,000 |
| **2.** JavaScript | $117,000 |
| **3.** Java | $104,000 |
| **4.** C | $103,000 |
| **5.** C++ | $102,000 |
| **6.** C# | $97,000 |
| **7.** PHP | $94,000 |
| **8.** SQL | $92,000 |

| Rank by Volume of Job Openings | |
|---|---|
| **1.** Python | 50,000 |
| **2.** SQL | 50,000 |
| **3.** Java | 45,000 |
| **4.** JavaScript | 38,000 |
| **5.** C++ | 29,000 |
| **6.** C# | 21,000 |
| **7.** PHP | 13,000 |
| **8.** C | 9,000 |

https://www.codeplatoon.org/best-paying-most-in-demand-programming-languages-2020/

# Integrated Development Environment (IDE)

- It is a piece of software that allows you to:
  - write your programs with a text editor
  - *compile* your programs
  - link in predefined libraries and other programs
  - run your programs
  - manage your programs
  - debug your programs
  - provides help about the language
- We are using the Code::Blocks IDE
- We are using the MinGW C compiler
- Both available for download

# Installing software on your PC

- Visit the following Wiki page (login required):

  https://wiki.york.ac.uk/pages/viewpage.action?spaceKey=software&title=Electronics

- Refer to the “Code:Blocks” section and follow the link to the relevant Google Drive:

  https://drive.google.com/drive/folders/1IF_YCwz9H3rkBBjNVrR8okWYkQ3Qn3Vp?usp=sharing

- Install both "Code:Blocks" IDE and the "MinGW" C compiler as follows

# Installing MinGW Compiler

- The MinGW folder contains the compiler
  - as well as libraries to support graphics and audio
- Download the folder to your PC
  - by selecting the folder and then right-clicking on “Download”
- Run 'MinGW.msi’
  - And follow the on-screen instructions

# Installing Code::Blocks on your PC

- Code::Blocks IDE
  - download both the .msi and .cab files to the same directory, then run the .msi file.
- Once the installation has completed
  - start Code::Blocks
  - select the 'Settings' menu in the toolbar
  - then select 'Compiler...'
  - ensure that the 'Selected compiler' reads 'GNU GCC Compiler'
  - Select the 'Toolchain executables' tab
  - Set 'Compiler's installation directory' to 'C:\APPS\MINGW' and click OK.

# Troubleshooting

- If you encounter problems installing the software please email:

itsupport@york.ac.uk

  - you will get help from our Departmental, Embedded IT Support Team who are ready to help you.

# Still having problems?

- If you still can't get Code::Blocks to work
  - or if you have a Mac
- Use the University's Virtual Desktop Service (VDS)
- Please follow the instructions at:

  https://www.york.ac.uk/it-services/services/vds/
- These are in two pools; one for Teaching in timetabled sessions and one for general Coursework.

# Structure of the module

- Lectures, labs and workshops
- 9 Lectures: programming concepts
  - pre-recorded
- 9 Laboratories: programming exercises in C
  - self-paced
- 9 Workshops: to support lecture and labs
  - Via Zoom
- 1 Assessment
  - Programming assignment and report

# Laboratory work

- Laboratory-led learning
- Graphics exercises
  - relate to the creation and manipulation of graphical objects
  - using a library of graphics defined in a file called graphics_lib.h
  - based on an open source graphics and gaming library called Allegro
- Music/Audio exercises
  - relate to the manipulation of sound files and production of music
  - use a library defined in a file called amio_lib.h
  - based on an open source libraries libsndfile, portaudio and portmidi
- You need to attempt BOTH graphics and audio exercises

Many exercises are directly related to the coursework assignment

# Times & locations

- Lectures
  - video-on-demand
  - released weekly
- Lab sessions
  - self-paced - working off campus
  - timetabled slot 9am-12noon Wednesdays
  - provides opportunity to ask for help assistance
- Workshops
  - review of the week's material
  - further opportunity for help/deeper understanding
  - timetabled slots 9am & 2pm Fridays

# Lecture/Lab Timetable

| Lectures | Laboratories |
|---|---|
| **Week 2: Introduction** | Intro to Code::Blocks, compiling, running, debugging, variables, statements, comments, assignment, **`printf,`** graphics, midi functions |
| **Week 3: Conditional statements** | Conditional statements (**`if`** and **`switch`**)<br>Getting input from user: **`scanf`**<br>Relational operators, compound statements, logical operators |
| **Week 4: Iteration** | Repetition and iteration: **`do`**, **`while`** and **`for,`** shorthand operators**: `++, --`** |
| **Week 5: Functions and the C Preprocessor** | Functions, writing your own, calling library functions |
| **Week 6 : Arrays & Strings** | Arrays and Strings. Initialising arrays, string terminator |
| **Week 7: Data structures and defining new types** | Structures and defining your own types<br>**`struct, typedef`,** dot operator |
| **Week 8: Pointers** | Pointers and passing by reference<br>`int *x, y  y = &x;` |
| **Week 9: Arrays & pointer and memory allocation** | Arrays and Strings revisited. Memory allocation<br>**`malloc`**, **`calloc, free`** |
| **Week 10: Advanced topics in programming** | Constructing a Software Project<br>Practical Software Design |

# Lab 1

- Learning how to use the IDE - Code::Blocks
- Compiling, debugging and running example C programs
- Modifying these example programs
- Using the graphics library for drawing pictures
- Using the music library for producing sounds

# Module Wiki

https://wiki.york.ac.uk/pages/viewpage.action?pageId=210830815

- Contains:
  - lab scripts
  - lectures – slides and videos
  - supporting reading material

# On-line resources

- It is not essential to buy any books. Many full-text books are available on-line through the University library
- https://yorsearch.york.ac.uk/
  - my search returned 3,578 full-text on-line results!
- Reference
  - B. W. Kernighan and D. M. Ritchie. The C Programming Language. Prentice-Hall. 1988
  - largely for historical interest!
- Much other online material also available:
  - tutorials, Wikis, videos and community helps sites
  - YouTube has many channels
  - stackoverflow.com is probably the best known community