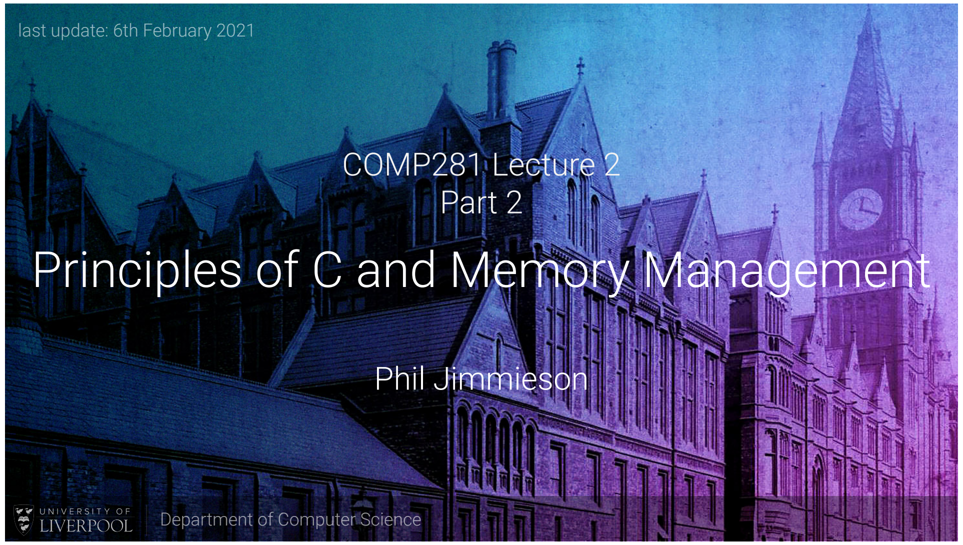


last update: 6th February 2021

COMP281 Lecture 2
Part 2

Principles of C and Memory Management

Phil Jimmieson

UNIVERSITY OF LIVERPOOL — Department of Computer Science

C Language Basics

```
#include <stdio.h>

int main(void)

{

    printf("Hello, World!\n");

    return 0;

}
```

- The main ( ) function
- Statements
- C Skeleton
- Identifiers
- Keywords
- Basic data variables and types
- Constants

The `main()` Function

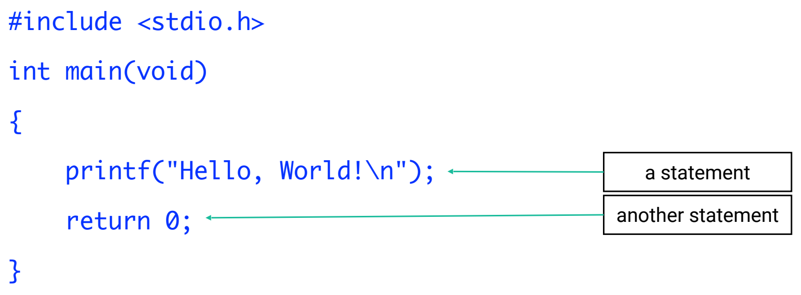
- Identify the start of the program
- Every C program has a `main()`
- “main” is a *C* ***keyword***. We **must not** use it for any other variable.

Statements

- “A specification of an action to be taken by the machine as the program executes.”
- Each statement in C needs to be terminated with semicolon (`;`).

Statements

- A specification of an action to be taken by the machine as the program executes.
- Each statement in C needs to be terminated with semicolon (`;`).



```
#include <stdio.h>
int main(void)
{
    printf("Hello, World!\n");
    return 0;
}
```

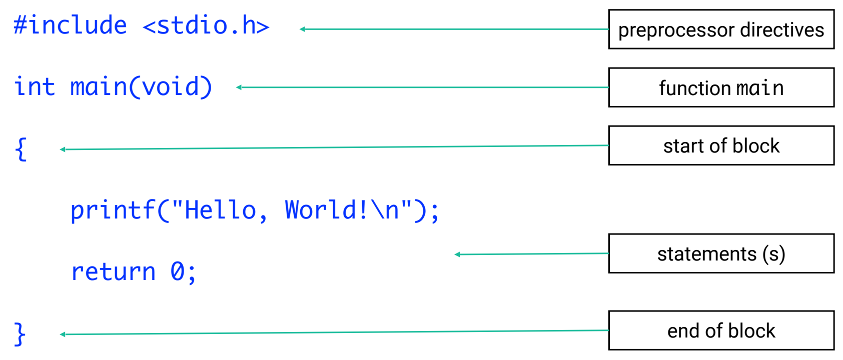
C Program Skeleton

In short, the basic skeleton of a C program looks like this:

```
#include <stdio.h>
int main(void)
{
    printf("Hello, World!\n");
    return 0;
}
```

C Program Skeleton

In short, the basic skeleton of a C program looks like this:



```
#include <stdio.h>

int main(void)

{

    printf("Hello, World!\n");

    return 0;

}
```

Identifiers

Words used to represent certain program entities

(variables, function names, etc.)

E.g.,

- `int bar_baz;`
  - `bar_baz` is an identifier used as a program variable.
- `Void CalculateArea (int radius)`
  - `CalculateArea` is an identifier used as a function name

Identifiers

Rules for naming identifiers:

Rule	**Example**
Can contain a mix of characters and numbers	W3c
Cannot start with a number	2assignments
Must start with a letter or underscore	Number1 _area
Can be of mixed cases	whoAmI
Cannot contain any arithmetic operators	Sm*il
Cannot be any other punctuation marks (separators)	!@#$%^&*(){}
Cannot be a C **keyword/reserved** word	main printf
Cannot contain a space	Oh yay
Identifiers are case sensitive	Happy ≠ happy

Keywords

These are reserved words in C. They may not be used as constants or variables or any other identifier names.

auto	else	long	switch
break	enum	register	typedef
case	extern	return	union
char	float	short	unsigned
const	for	signed	void
continue	goto	sizeof	volatile
default	if	static	while
do	int	struct	_Packed
double			

Basic Data Variables and Types

- All data in C has to have a specified type
- C has several types of variables, but there are a few basic types:
  - Integers
  - Unsigned integers
  - Floating point numbers
  - Chars
- Variables hold data of a particular type only
- Variables must be declared before use

Integers

Whole numbers which can be both ***positive*** and ***negative***.

Defined using:

- `char`
- `int`
- `short`
- `long`
- `long long`

Unsigned integers

Whole numbers which can only be ***positive***.

Defined using:

- `unsigned char`
- `unsigned int`
- `unsigned short`
- `unsigned long`
- `unsigned long long`

Floating point numbers

Real numbers (numbers with a decimal part)

Defined using:

- `float`
- `double`

Chars

Equivalent to 'letters' in English.

Examples:

- Numeric digits: 0 – 9
- Letters: a – z and A – Z
- Space (blank)
- Special characters: !@£$%^&*()

Single character

e.g., `char my_letter = 'E';`

(the declared character must be enclosed within a single quote!)

Constants

- Entities that appear in the program code as fixed values.
- Any attempt to modify a CONSTANT will result in error.
- There are 4 types of constants:
  - Integer constants, e.g., `const int MAX_NUM = 9999;`
  - Floating point constants, e.g., `const double VAL = 1.23e4;` (1.23 x 10^4)
  - Character constants, e.g., `const char letter = 'l';`
  - Enumeration, e.g., `enum City { Manchester, Liverpool, Leeds };`
- **The other way to define "constants is to use"** `#define`

Constants

Example 1

```
#include <stdio.h>
void main()
{
    int i = 1;
    const int x = 2;
    i = 3;
    x = 4;
    printf("i = %d\nx = %d", i, x);
}
```

This creates an error

Constants

Example 2

```
#include <stdio.h>
#define PI 3.14
int main()
{
    int r = 2, area ;
    printf("The radius of circle is %d.\n",r) ;
    area = PI * (r * r) ;
    printf("Area of the circle = %d.\n", area) ;
    return 0;
}
```

Define a constant using `#define` preprocessor directive

Summary

Today

- Compiling C Programs
  - 4 kinds of files to work with: **source code** files, **header** files, **object** files, and binary **executables**
  - The Preprocessor -> `#define`, `#include`
  - The Compiler -> `% gcc foo.c –o foo`
  - The Linker -> `% gcc foo.o bar.o baz.o -o myprogram`

Today

- C Language Basics
  - The `main()` function
  - C Program Skeleton -> segments
  - Identifiers -> naming variables, functions, etc.
  - Keywords -> reserved words, may not be used as identifier names
  - Basic Data Variables and Types
    - Integers, unsigned integers, floating point numbers, chars
  - Constants (fixed values) -> `const int x = 2;` `#define PI 3.14;`

Next

- C Language Basics
  - …
  - Basic I/O
  - Operators
  - Decision Making