C Program Structure

An Example

```
#include <stdio.h>
#define PI 3.1415926

int main( void ) {
  printf( "Please enter the radius of a circle > " );
  double radius;
  scanf( "%lf", &radius );

  double area = PI * radius * radius;
  printf( "The area of a circle of radius %lf is %lf\n",
          radius, area );
  return 0;
}
```

Header Files

```
#include <stdio.h>
#define PI 3.1415926
```

`stdio.h` is one of a number of header files defined by the C standard library
Header files contain information about functions, constants, types and (occasionally) global variables that exist in a part of the C library
Header files use the same syntax as C source files, only with the ".h" extension
However, header files should never contain actual code, i.e. no functions with bodies, and no global variable definitions (eg those with initializations)

As we've seen, the `#include` directive reads the contents of the included file and flattens it into a temporary file before the proper C compiler runs
In particular, `#include` brings in declarations of many standard I/O related functions (eg `printf` and `scanf`) so that we may use them:

- `printf` is for writing data to standard output
- `scanf` is for obtaining data from the standard input

`#define`

`#define PI 3.1415926` defines a constant

The preprocessor will then automatically replace each occurrence of the text PI in the subsequent code by `3.1415926`

The main C compiler never sees the symbol PI in the code

`main()` Function Definition

```
int main( void ) {
        // BODY
        return 0;
}
```

The entry point for a C program is the `main` function
Every C program (but not every C source file) must contain a `main` function
The remaining lines of the program form the body of the function which is enclosed in braces `{...}`
`main` must return an integer value `int`:

- 0 for success
- non-zero value for failure

If we omit the return statement, `main` will return an undefined value (and most compilers will issue a warning)
`main` either takes no parameters (use `void` as shown above), or an alternative form giving access to command line arguments

Outputting Data

```
printf( "Please enter the radius of a circle > " );
```

Use `printf`

```
printf( "The area of a circle of radius %lf is %lf\n",
        radius, area );
```

Here, the `printf()` format control string contains two uses of a long float, the first one prints the `radius`, and the second one prints the `area`

Inputting Data

```
double radius;
scanf( "%lf", &radius );
```

Here, we declare a double-precision floating point variable called `radius`
Then we call `scanf()` to read a long float (`%lf`, i.e. a double) from the standard input, passing `&radius` (the address of radius) so that `scanf()` can store the floating-point value that was read into `radius`

Variable Declarations

```
double area = PI * radius * radius;
```

Here, we declare a double variable called `area` and initialise it

Identifiers and C Keywords

There are words classified as reserved words, standard identifiers (e.g. `printf` and `scanf`) and user-defined identifiers (`radius` and `area`)

Reserved words:

- They are in lowercase
- Have a defined meaning in C
- Cannot be used for other purposes

The list of ANSI C reserved words are:

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