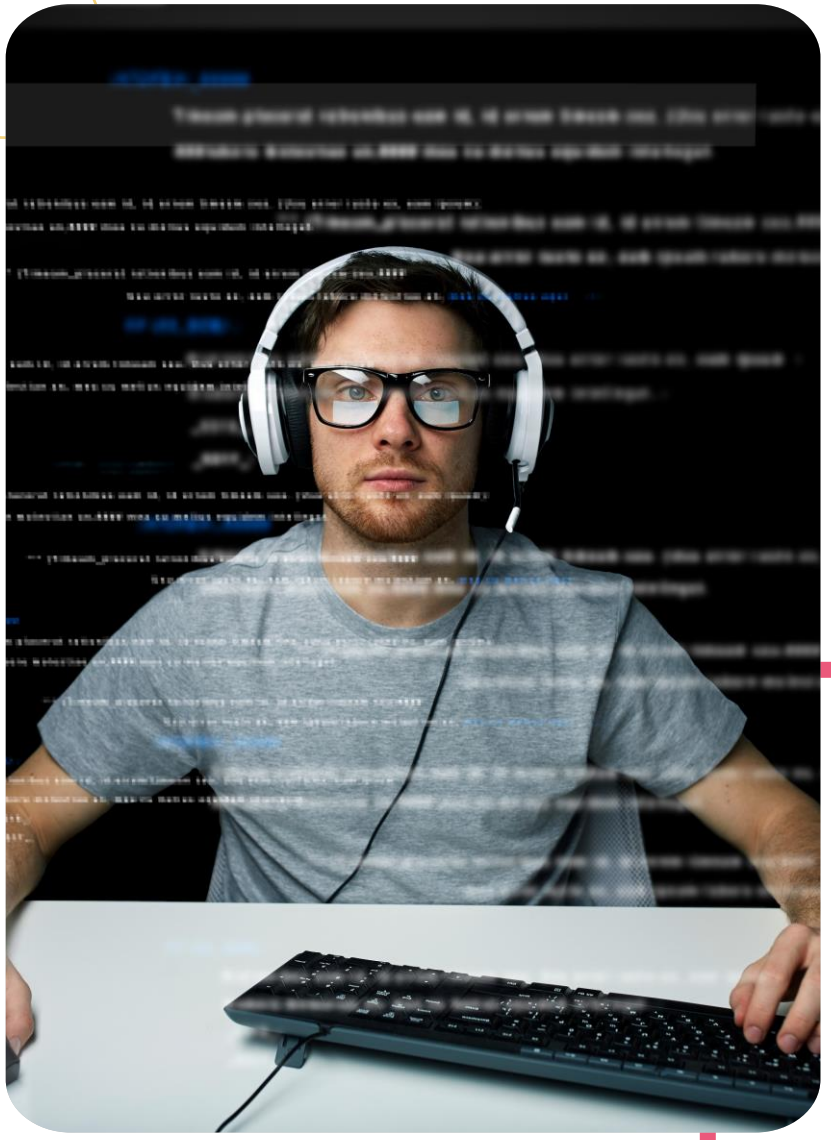
Diploma in

Computer Science

Program Structure



Objectives

- Understand the basic structure of a C program
- Discuss the importance of each part of the program
- Explore the basics of program readability
- Explain the significance of functions as the body of the program

Documentation section

Challenge

Analyse the demo from our last lesson – the ‘Hello world’ demo – and try to see which sections match the basic structure of a C program.

Documentation section

- Contains several comment lines
- Your name and names of others you would have worked with
- Date program created & version number
- Brief description of purpose of program
- Refer to user guide or other reference

Documentation section

Using another programmer's code without permission and/or acknowledgement is considered **plagiarism**.

If you are building commercial software, you need to cite the source.

Did you know?

The ‘C’ programming language had a predecessor called ‘B’.

Guidelines to copying code

- **If using copied code as part of your project:** include a reference as per programmer's preferred format or use conventional citing method.
- **If using copied code as standalone program:** make sure conditions are met and correctly documented and referenced.
- Contact developers if anything is not clear.
- Contact details always included.

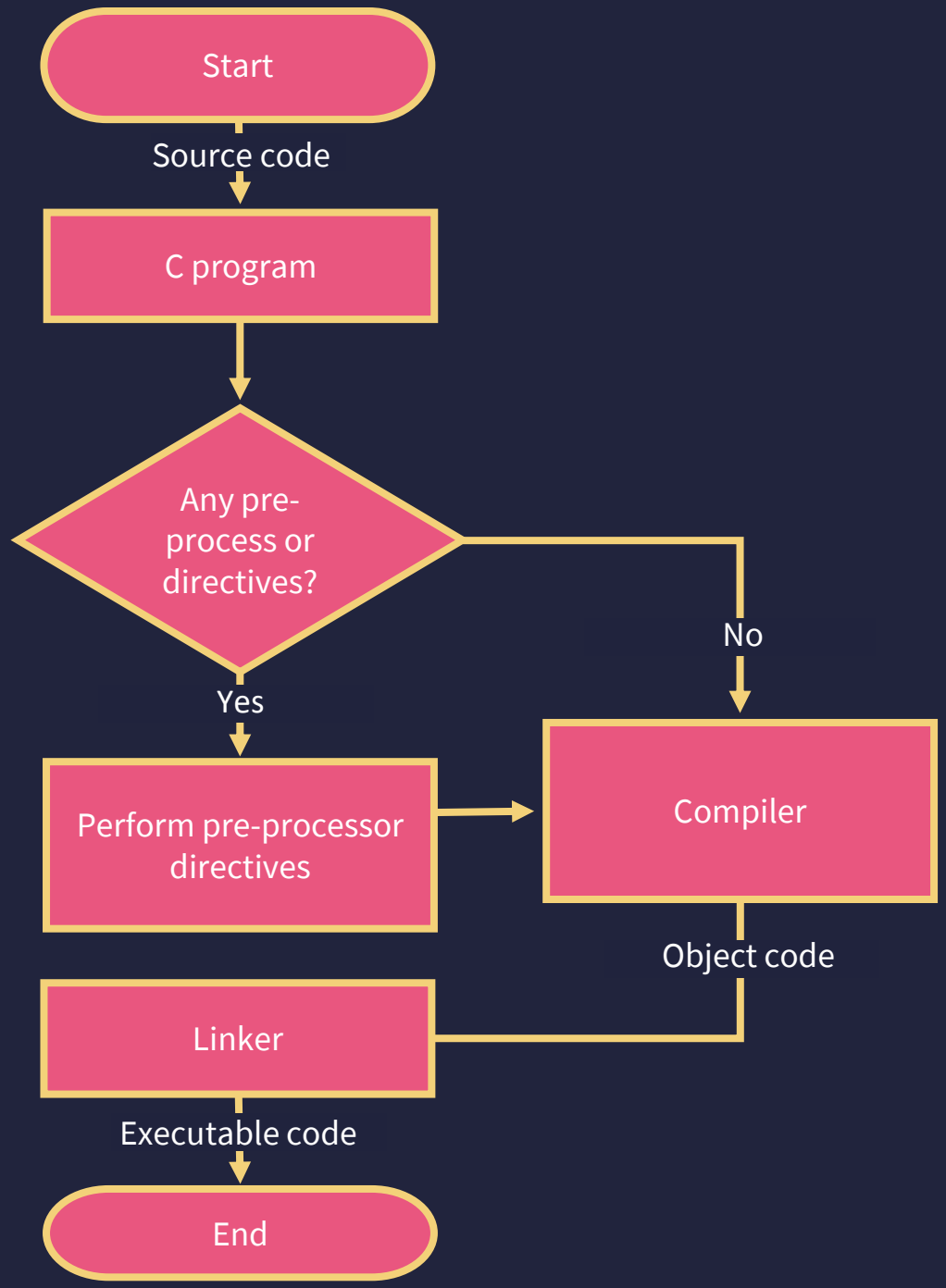
Pre-processor directives

Commands carried out before processing begins

Link section

Definition section

Pre-processor directives



Pre-processor directives

Four main types of pre-processor directives:

- Macros
- File inclusion
- Conditional compilation
- Miscellaneous directives

‘include’ directive

Tells computer to include the specified file in the compilation process

Angular brackets – search for specified file in system area

For example: stdio.h file

‘include’ directive

Can use quotes instead of brackets

Easier to split large program into several files

With quotes, pre-processor will look for file in current directory

Definition section

- Create new objects using macros
- Macros → piece of code with a name
  - Can be simple label
  - Can be an expression
  - Can perform calculations or pass arguments

Definition section

- To define macro, use **#define**
- Will last until **#undef**

Pre-processor replacements can result in unexpected behaviour

- No termination symbol
- Pre-defined macros – for diagnostics and documentation

Conditional definitions

- Only honoured if specified condition is met
- Direct computer to compile a specific portion or code or skip certain section

#ifdef
ifndef
#if
#endif
#else
#elif

Examples

#error will halt compilation if an error is found

Other directives

Compiler-specific directives used in specialised programs

Header files can sometimes cause errors

Header files often paired with code files

Declaration Section

Global declaration

Everything visible is declared

Components will live in memory for as long as program is running

Global variable can have same name as a local variable but local variable will take precedence

Function declaration

Declare your own functions

Functions must be declared above the main function

Compiler will make reference to these functions

Main function

- Main function of the program
- Where execution takes place
- Code is called from within main function to be executed
- Consists of two sections:
  - Declaration section
  - Execution section

Declaration section

- Variables and constants to be used in your program are declared and initialised
- Visible from within the main function

Execution section

- Computer begins, first thing it will look for is the main function
- First point of entry into the program

Return statement

- Once the main function has executed all the statements within it, it needs to exit
- Tells computer to end function and return a value
- Returns zero if executed successfully
- Returns a value other than zero if there was a problem

User-defined functions

Your own functions

Allow you to split your program into sections without having to save it in different files

Type in the name of the function where you want it to be used along with any arguments

User-defined functions

Useful for moving specific routines into functions

For example

In a program to calculate the area of various shapes, you could name your functions *areasqr, areatri, areacircle*.

Did you know?

‘Time to hello world’ (TTHW) is the time it takes to write a ‘Hello, World!’ program in a programming language. This is one measure of a programming language's ease-of-use.

Explaining your code

Comments in coding

Used by programmers to explain code that is not clear at a glance

Can be used anywhere

Don’t affect execution of program-compiler ditches comments before compilation begins

Important part of coding

Types of comments

- Single line comments
- Multi-line comments

In-line comments

Written in just one line – same line as code it relates to

Suitable for short comments

Recognisable by double forward slash //

Multi-line comments

Starts with forward slash and an asterisk /* and ends with asterisk and forward slash */

Suitable for describing code in more detail

Conventional way for C source files is to add asterisk at beginning of each line

Comment conventions

Comments ensure code is readable

Comments placed after code being explained rather than before

Avoid shorthand

For self-explanatory blocks of code, put comment on separate line

No blank lines between code and comment

(unless block of code with comment at beginning)

Sentence case

Check spelling and grammar

Remove code and leave comment in final version of a block of code

Have valid reason for leaving commented-out code

May need to comment-out sections of code during debugging