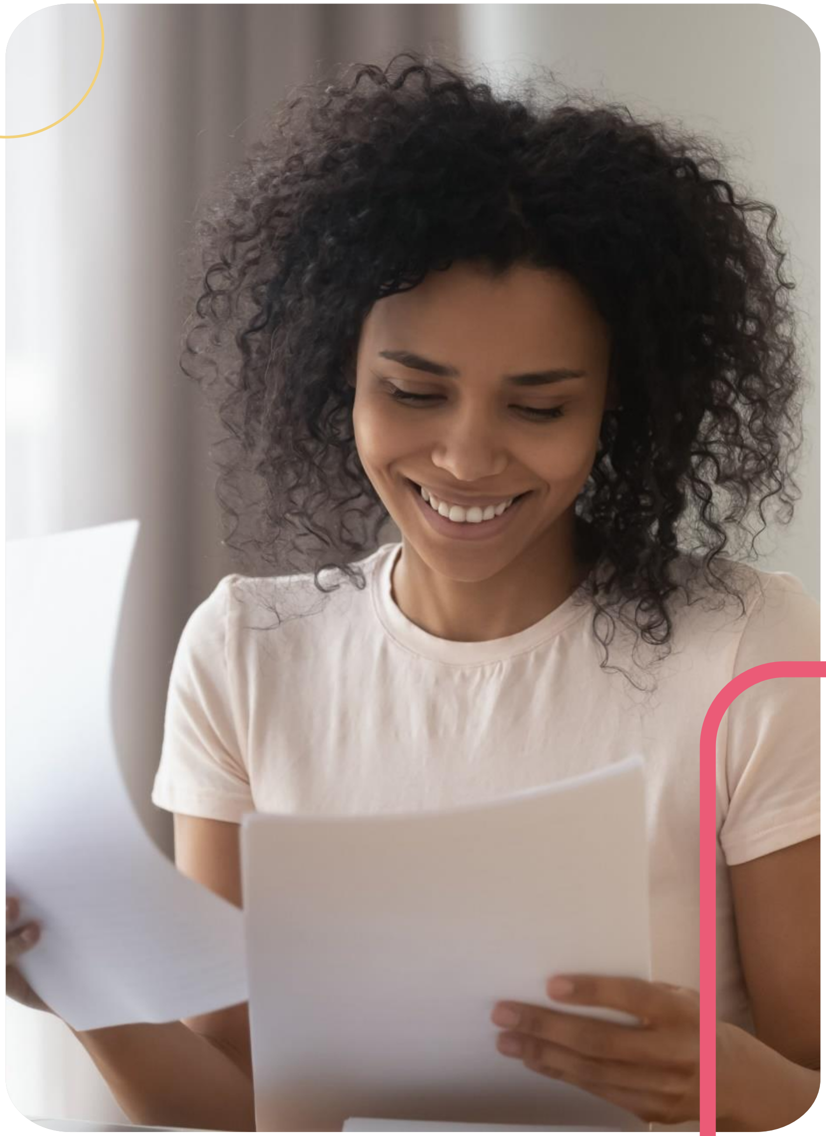
Diploma in

Computer Science

Pre-processors, Header Files and Libraries

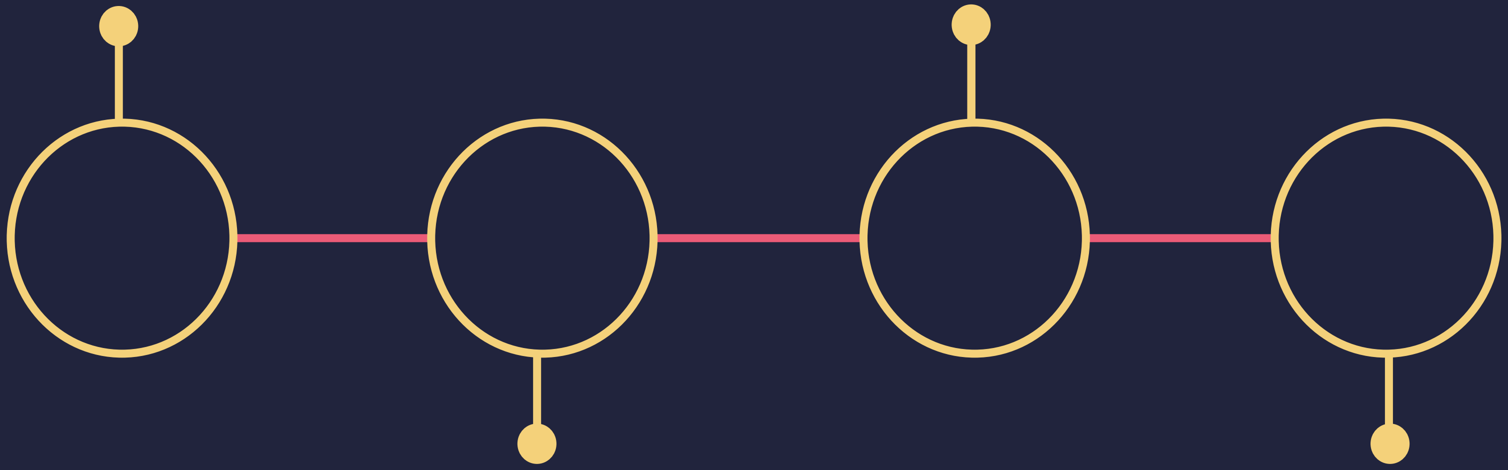


Objectives

- Appreciate the role of the pre-processor
- Explore the contents of header file
- Examine the role of a library in programming
- Comprehend the concept of APIs

The compilation process (again!)

Pre-processor directives



Input: high-level language code
Pre-processor
Output: pure high-level language code

Compiler
Output: relocatable code

Assembler
Output: assembly code

Linker and loader
Output: machine code

Pre-processor

Pre-processor processes directives indicated by the hash symbol

Hash symbol directs the pre-processor to process whatever comes after it

Pre-processor

Separate program from the compiler that does preparatory work

C grammar not strictly implemented in the pre-processor

Other kinds of text files can also be handled when needed

Pre-processing defined by four main stages

Stage 1: Trigraph replacement

- Pre-processor replaces trigraph sequences with the characters that they represent
- Trigraphs - sequences of 3 characters that should be treated as 1 character
- Used when no keyword to represent that particular set of characters

Stage 2: Line splicing

- Physical source lines are spliced to form logical lines

Stage 3: Tokenisation

- Pre-processor breaks the resulting code into tokens
- Comments are removed and replaced with white space

Stage 4: Macros & directives

- Lines with pre-processor directives are executed
- Files that are not part of the code are lined up for compilation

Types of directives

The pragma directive

- Provides a way to request special behaviour from the compiler
- Used for large programs that require the compiler to carry out compilation in a certain way
- Usually followed by a single token

```
#pragma token
```

Syntax

- Only tokens from the set of permitted are recognised in this directive

The pragma directive

From C99 onwards, you can embed the pragma directive in a macro, like this:

```
#pragma exit
```

Example

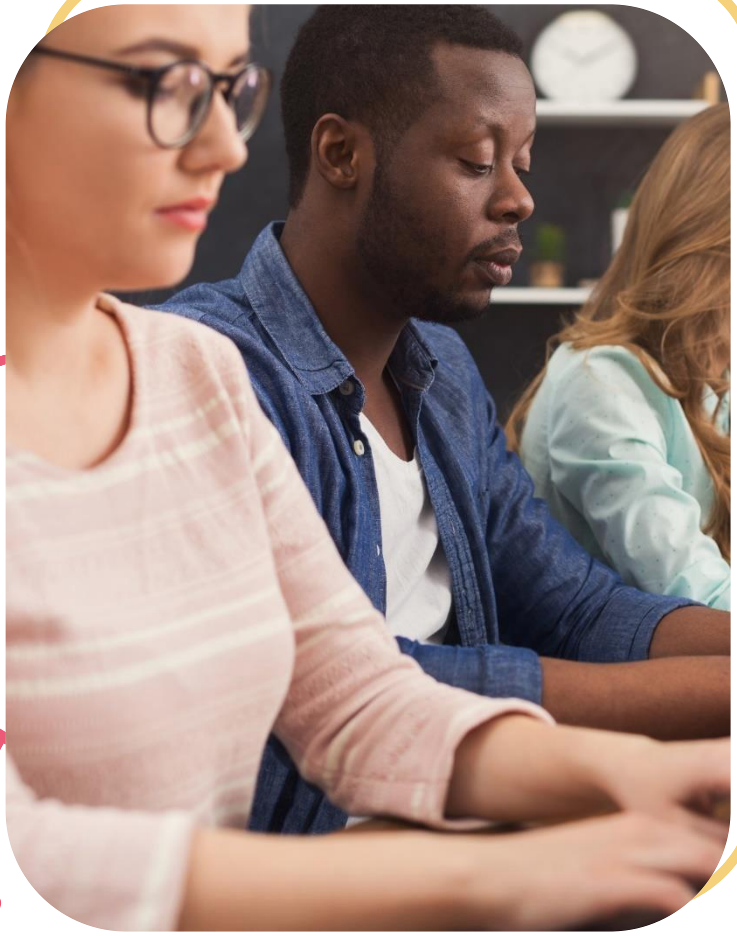
```
#include<stdio.h>
int pragmaTest();

#pragma startup pragmaTest
#pragma exit pragmaTest

int main() {
  printf("\nI am in the main function");
  return 0;
}
int pragmaTest() {
  printf("\nI am in the test function");
  return 0;
}
```

Sample

Where do header files fit in?

If the filename is enclosed in angle brackets, the computer knows that it has to look for the specified file in the standard C ‘include’ folder.

- Computer replaces tag #include filename
- Text is then treated as code
- A header file is characterised by the extension .h
- Could encounter header files that carry the following extensions: .hpp, .def and .xbm
- Header files should only be included once



Header files contain all the functions and expressions you need!

Libraries in C

C standard library

- A collection of header files that contain functions and procedures to perform various tasks
- Started in 1970s where many people used C on Unix across multiple architectures
- Standard was formally established in 1984
- Standard specifically for C followed

Understanding the role of the standard library

- All you need to know is the syntax plus parameters
- Many common tasks have already been partially solved

- Libraries shave off development time

AND

- Allow interoperability of code among various systems

Using libraries...

```
#include<math.h>.
```

Syntax

Did you know?

The first C compiler ran on a PDP 11 with only 4 kilobytes of RAM.

What’s in the C library?

Assert.h

A simple way to halt programme with debugging information if provided assumption does not hold

Useful for quick test runs

Calls the abort function to quit and dump the core

Can be used multiple times without causing an error

Can enable and disable the assert macro

Ctype.h

Useful for testing and mapping characters

All functions accept it as a parameter but value must be presented as an unsigned character

Contains functions that can compare many types of characters

Errno.h

Global integer variable

Stored in errno by certain library functions whenever they detect an error

float.h, limits.h, and math.h

Each has a set of mathematical functions that allow you to carry out special mathematical operations

Using non-standard header files may save you a lot of coding but it is detrimental to the portability of your program

Float is used for floating point operations

Locale.h

When you want to tell the computer how to handle regional customs

C uses this ASCII text and American formatting

Setjmp.h

Allows you to use a sort of saveable go-to statement to jump to places you've already been

Also allows you to jump from one function to another

A kind of exception handling for a C programmer

Signal.h

Utilises the Unix technique for inter-process communication to asynchronously call a handler function or take default aciton

Primarily provided Unix functionality and not portable

Stdarg.h

Carries a number of assumptions:

Variadic function must declare at least one fixed argument

Function must call va_end before returning

Register variables, functions and arrays can’t be returned by va_arg

If a type widens with default argument promotions, va_arg should request the widened type

Stddef.h

Carries a number of assumptions:

Defines various variable types and macros

C can be compiled in either a hosted or a freestanding environment

Freestanding environment reserved for embedded programming where there isn’t enough space to store the entire standard library

Stdio.h

Provides the standard functions for input and output on the host system

These functions will interface with the operating system

Outputs not limited to display characters from the screen

Stdlib.h

Defines four variable types, several macros, and various functions for performing general functions

Also contains functions for allocating and deallocating memory

String.h

Used for string manipulation

Contains different sets of string manipulation functions

Time.h

Contains many to handle conversions of time and date

Limited to nothing smaller than seconds of precision

User defined header files

- C allows you to add nonstandard header files
- Instead of angle brackets, enclose the name of the header file in quotes

C standards

ANSI

- First C standard to be developed
- Supported by most widely used compilers such as GCC and Clang
- Any source code written in standard C without any hardware dependent assumptions is guaranteed to compile correctly on any platform

ISO

- In 1990 ANSI C adopted by ISO
- Iterations C99, C11 and C18 were released
- Five aspects covered in the standard:
    - Representation of C programs
    - Syntax and constraints of the C language
    - Semantic rules for interpreting C programs
    - Representation of input data to be processed by C programs
    - Representation of output data produced by C programs

POSIX

- Contains header files that are not available in C standard library
- POSIX = portable operating system interface
- A super set of the standard C library
- Developed at around the same time as ANSI standard

Challenge

- Which C standard does your compiler conform to?