Header Files

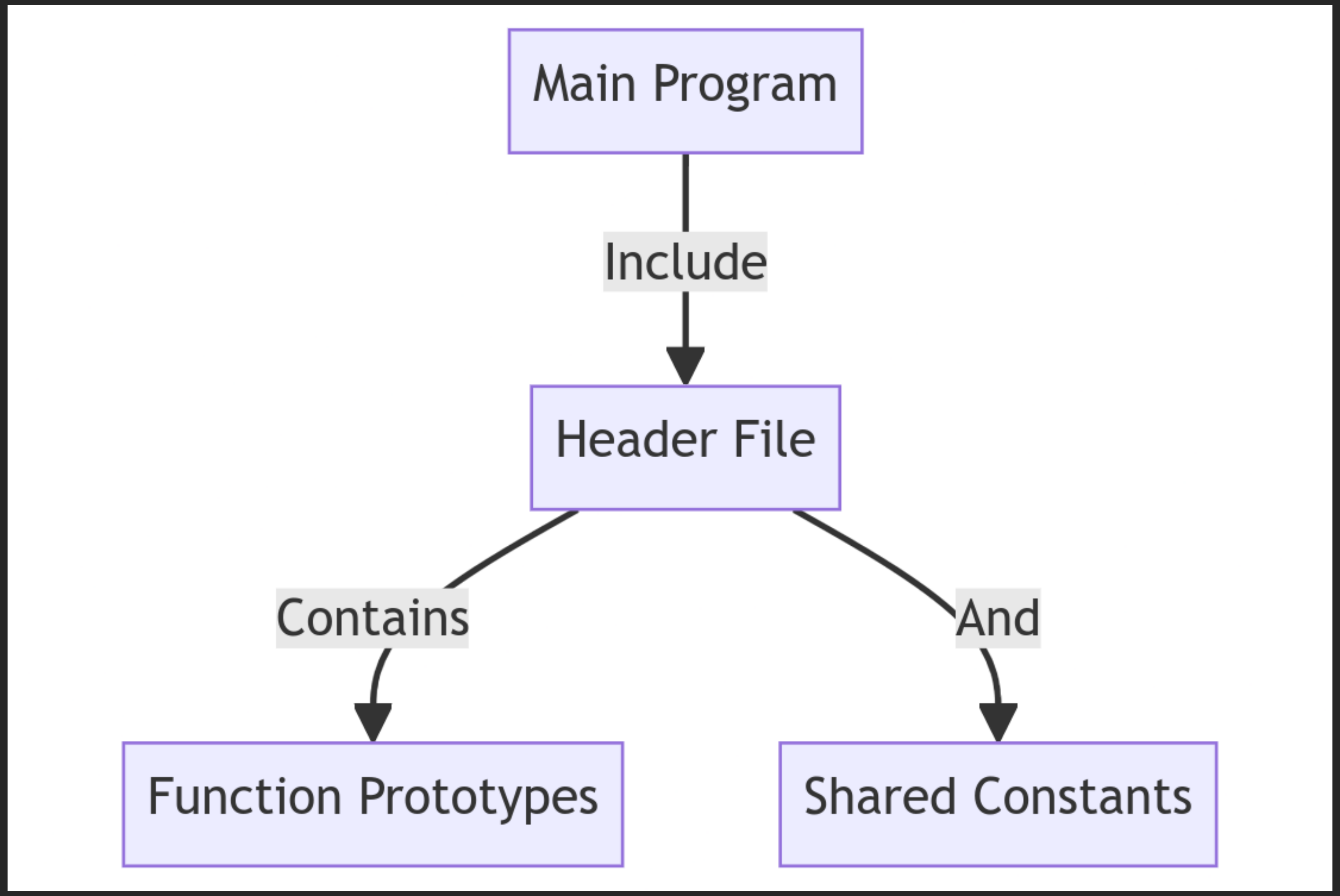
```
module = Module(
    code="ELEE1147",
    name="Programming for Engineers",
    credits=15,
    module_leader="Seb Blair BEng(H) PGCAP MIET MIHEEM FHEA"
)
```

Download as a PDF

Why Use Header Files?

- **Modularity:** Separate interface from implementation.
- **Reusability:** Share functions and data structures.
- **Readability:** Enhance code organisation.
- **Function Prototypes:** Allows the compiler to check function signatures during compilation.
- **Precompiled Headers:** Speeds up compilation by avoiding redundant parsing of headers in multiple source files.

How Does It Work?



Example

```c
// main.c
#include "header.h"

int main() {
    greeter();

    printf("PI: %.5f\n", PI);
    printf("Golden Ratio: %.5f\n", GR);

    struct Student s1 = {"Ada Lovelace", 42, 1.0f};
    printf("Student: %s, ID: %d, Grade: %.1f\n",
        s1.name, s1.studentId, s1.classification);

    return 0;
}
```

Compile command for reference:

```
gcc main.c header.c -o main.exe
```

```c
// header.h
#ifndef HEADER_H // Header guard
#define HEADER_H // Macro

#include <stdio.h> // Other libraries

void greeter(); // Function prototype

#define PI 3.14159 // Shared constant
#define GR ((double)1.61803) // Golden Ratio

// Shared DataStorage
struct Student {
  char name[50];
  int studentId;
  float classification;
};

#endif // HEADER_H
```

```c
// header.c
#include "header.h"

void greeter(){
    printf("Hello World!")!
}
```

What are Header Guards?

- **Purpose:** Prevent multiple inclusions of the same header file.
- **Issue:** Without guards, redefinitions can occur during multiple inclusions.
- **Solution:** Use preprocessor directives to conditionally include the contents.

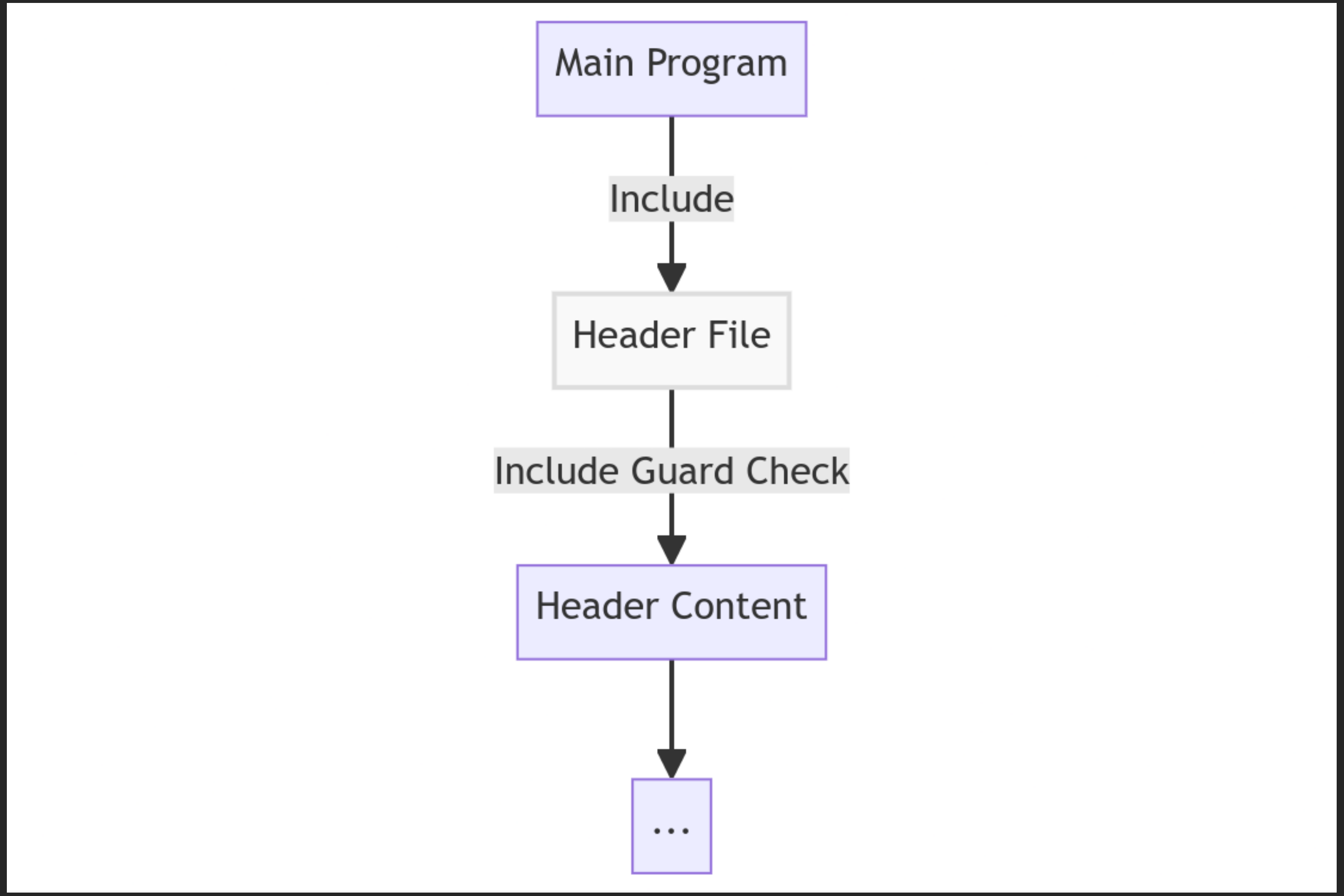
Why Use Header Guards?

- **Avoid Redefinitions:** Prevent compilation errors due to duplicate declarations.
- **Ensure Once-Only Inclusion:** Each header is included only once in a translation unit.
- **Improve Compilation Efficiency:** Reduce redundant parsing of header contents.

```
#ifndef HEADER_H
#define HEADER_H

...

#endif //end of HEADER_H
```



Example

```c
// main.c
#include "header.h"

int main() {
    greeter();

    printf("PI: %.5f\n", PI);
    printf("Golden Ratio: %.5f\n", GR);

    struct Student s1 = {"Ada Lovelace", 42, 1.0f};
    printf("Student: %s, ID: %d, Grade: %.1f\n",
        s1.name, s1.studentId, s1.classification);

    return 0;
}
```

Compile command for reference:

```
gcc main.c header.c -o main.exe
```

```c
// header.h
#ifndef HEADER_H // Header guard
#define HEADER_H // Macro

#include <stdio.h> // Other libraries

void greeter(); // Function prototype

#define PI 3.14159 // Shared constant
#define GR ((double)1.61803) // Golden Ratio

// Shared DataStorage
struct Student {
  char name[50];
  int studentId;
  float classification;
};

#endif // HEADER_H
```

```c
// header.c
#include "header.h"

void greeter(){
    printf("Hello World!")!
}
```

Preprocessor Directive: `#include` `""` vs `<>`

- Use `#include ""`
  - for including header files that are part of your project or are in the current directory.
- Use `#include <>` for
  - including standard library header files or other headers that are part of the system include directories.

Standardised Header Examples

stdio.h:

```c
#ifndef _STDIO_H
#define _STDIO_H    1

#define __GLIBC_INTERNAL_STARTING_HEADER_IMPLEMENTATION
#include <bits/libc-header-start.h>
...

__END_DECLS

#endif /* <stdio.h> included.  */
```

https://code.woboq.org/userspace/glibc/libio/stdio.h.html

Example

```
// main.c
#include "header.h"

int main() {
    greeter();

    printf("PI: %.5f\n", PI);
    printf("Golden Ratio: %.5f\n", GR);

    struct Student s1 = {"Ada Lovelace", 42, 1.0f};
    printf("Student: %s, ID: %d, Grade: %.1f\n",
        s1.name, s1.studentId, s1.classification);

    return 0;
}
```

Compile command for reference:

```
gcc main.c header.c -o main.exe
```

```
// header.h
#ifndef HEADER_H // Header guard
#define HEADER_H // Macro

#include <stdio.h> // Other libraries

void greeter(); // Function prototype

#define PI 3.14159 // Shared constant
#define GR ((double)1.61803) // Golden Ratio

// Shared DataStorage
struct Student {
  char name[50];
  int studentId;
  float classification;
};

#endif // HEADER_H
```

```
// header.c
#include "header.h"

void greeter(){
    printf("Hello World!")!
}
```

Example

```c
// main.c
#include "header.h"

int main() {
    greeter();

    printf("PI: %.5f\n", PI);
    printf("Golden Ratio: %.5f\n", GR);

    struct Student s1 = {"Ada Lovelace", 42, 1.0f};
    printf("Student: %s, ID: %d, Grade: %.1f\n",
        s1.name, s1.studentId, s1.classification);

    return 0;
}
```

Compile command for reference:

```
gcc main.c header.c -o main.exe
```

```c
// header.h
#ifndef HEADER_H // Header guard
#define HEADER_H // Macro

#include <stdio.h> // Other libraries

void greeter(); // Function prototype

#define PI 3.14159 // Shared constant
#define GR ((double)1.61803) // Golden Ratio

// Shared DataStorage
struct Student {
  char name[50];
  int studentId;
  float classification;
};

#endif // HEADER_H
```

```c
// header.c
#include "header.h"

void greeter(){
    printf("Hello World!")!
}
```

Macros

Macros in C are a way to **define** constants or simple functions using the `#define` directive. They are preprocessor directives, meaning they are processed before the actual compilation of the code.

```c
// example_macros.h

#ifndef EXAMPLE_MACROS_H
#define EXAMPLE_MACROS_H

#define PI 3.14159 // Shared Constant
#define SQUARE(x) ((x) * (x)) // Function

#ifdef _MSC_VER
// Code specific to Microsoft Version C/C++
#endif //end of _MSC_VER

#endif // end of EXAMPLE_MACROS_H
```

Standardised Header Examples: [math.h]

```
...
#define    M_E              ((double)2.7182818284590452354)  /* e */
#define    M_LOG2E          ((double)1.4426950408889634074)  /* log 2e */
#define    M_LOG10E         ((double)0.43429448190325182765) /* log 10e */
#define    M_LN2            ((double)0.69314718055994530942) /* log e2 */
#define    M_LN10           ((double)2.30258509299404568402) /* log e10 */
#define    M_PI             ((double)3.14159265358979323846) /* pi */
...
int __signbitl(long double);
__END_DECLS

#endif /* !_MATH_H_ */
```

https://github.com/openbsd/src/blob/master/include/math.h

Example

```c
// main.c
#include "header.h"

int main() {
    greeter();

    printf("PI: %.5f\n", PI);
    printf("Golden Ratio: %.5f\n", GR);

    struct Student s1 = {"Ada Lovelace", 42, 1.0f};
    printf("Student: %s, ID: %d, Grade: %.1f\n",
        s1.name, s1.studentId, s1.classification);

    return 0;
}
```

Compile command for reference:

```
gcc main.c header.c -o main.exe
```

```c
// header.h
#ifndef HEADER_H // Header guard
#define HEADER_H // Macro

#include <stdio.h> // Other libraries

void greeter(); // Function prototype

#define PI 3.14159 // Shared constant
#define GR ((double)1.61803) // Golden Ratio

// Shared DataStorage
struct Student {
  char name[50];
  int studentId;
  float classification;
};

#endif // HEADER_H
```

```c
// header.c
#include "header.h"

void greeter(){
    printf("Hello World!")!
}
```

INCLUDING A HEADER FILE IN C BE LIKE..
TA-DA! JUST SOME DECLARATIONS OUT OF THIN AIR!