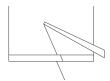
Bring ideas to life

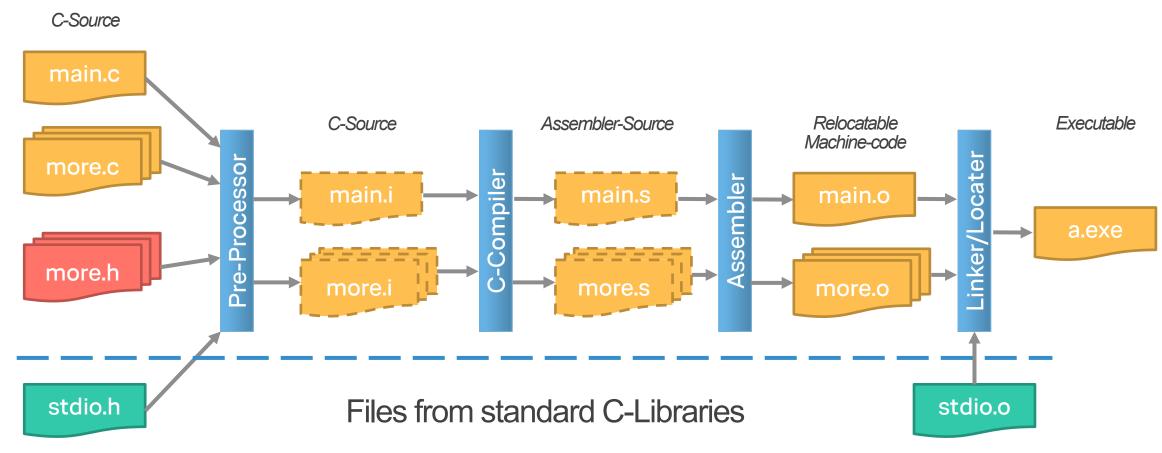
VIA University College



C Program Structure

The Compilation Process





ESW1 C-Program Structure - Lars Bech Sørensen, Erland Larsen, Ib Havn

Source Files

Filenames, use underscore '_' as word delimiter or camelCase



- Header files (*.h)
 - Declaration of functions and global variables
 - Can be compared to Java's interfaces
- C Source files (*.c) (modules)
 - Definition of functions (methods in Java) and variables in module scope or global scope

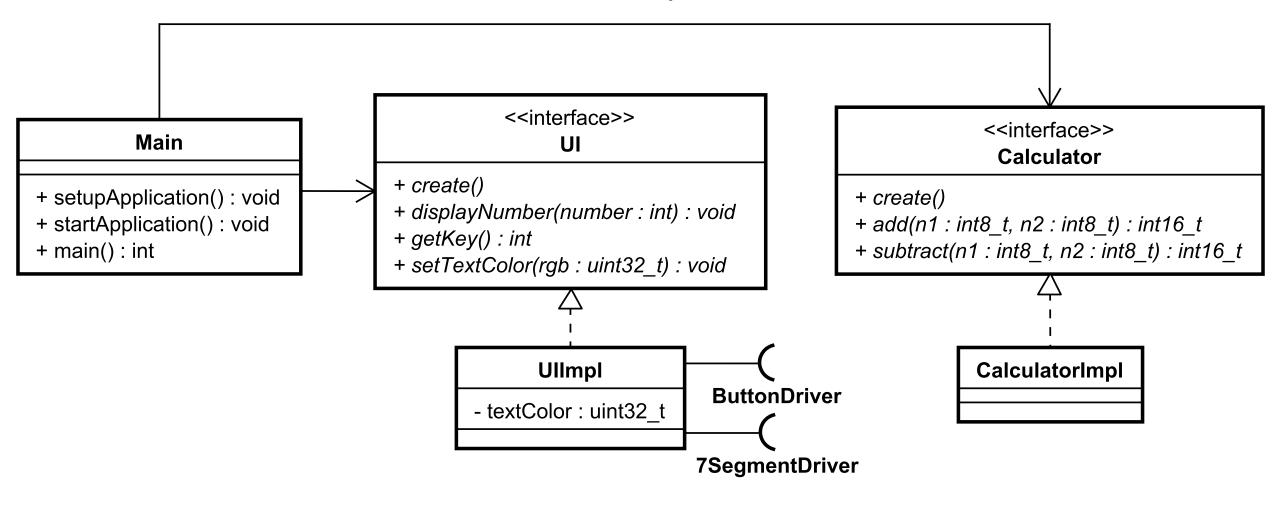


- + sumOfCalculations : uint16_t {declaration}
- + add(n1 : int8_t, n2 : int8_t) : int16_t
- + subtract(n1 : int8_t, n2 : int8_t) : int16_t



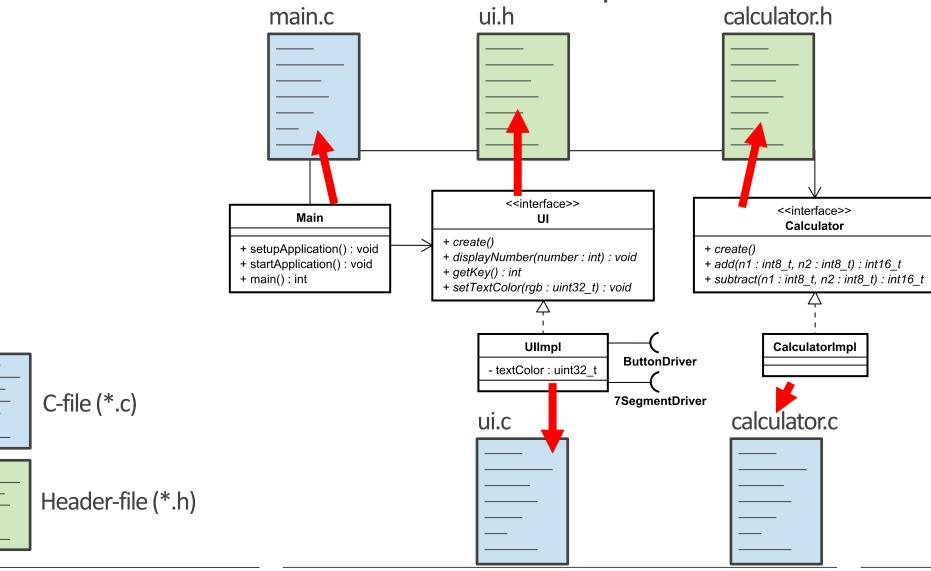
- + sumOfCalculations : uint16 t {definition}
- + create()

Source Files – An Example



ESW1 C-Program Structure - Lars Bech Sørensen, Erland Larsen, Ib Havn

Source Files – An Example



Header Files

E.g. of file buffer.h

In new versions of C-compilers it is ok to just write:

#pragma once

As the first line in the file

Prevents the compiler from including the file more than once!

```
<<interface>>
#ifndef BUFFER H
                                                            Buffer
                                              + BUFFER SIZE : uint8 t = 16 {1..255}
#define BUFFER H
                                                                                    <<enumeration>>
                                                                                    bufferReturnCode
                                              + create() : buffer t
#include <stdint.h>
                                              + delete(self : buffer t) : bufferReturnCode t
                                              + getItem(self : buffer t, item : uint8 t*) : bufferReturnCode
                                                                                   UNKNOWN BUFFER
#include <stdbool.h>
                                              + putItem(self : buffer t, item : uint8 t*) : bufferReturnCode
                                                                                   EMPTY
                                              + isEmpty(self : buffer t) : boolean
                                                                                   FULL
                                              + isFull(self : buffer t) : boolean
                                              + noOfItems(self : buffer t) : uint8 t
#define BUFFER SIZE 16 // 1..255
                                              + clear(self : buffer t) : bufferReturnCode
typedef struct bufferStruct * buffer t;
typedef enum {
     BUFFER OF
                                                             Standard in all
     ,BUFFER UNKNOWN BUFFER
                                                               Header files
     ,BUFFER EMPTY
     ,BUFFER FULL
} bufferReturnCode t;
buffer t buffer create(void);
bufferReturnCode t buffer delete(buffer t/self);
bufferReturnCode_t buffer_getItem(buffer_t self, uint8_t * item);
bufferReturnCode_t buffer_putItem(buffer_t self, uint8_t * item);
bool buffer isEmpty(buffer t self);
bool buffer isFull(buffer t self);
uint8 t buffer noOfItems(buffer t self);
bufferReturnCode_t buffer_clear(buffer_t self);
#endif /* BUFFER H */
```

2022-09-05

Header Files – How to include

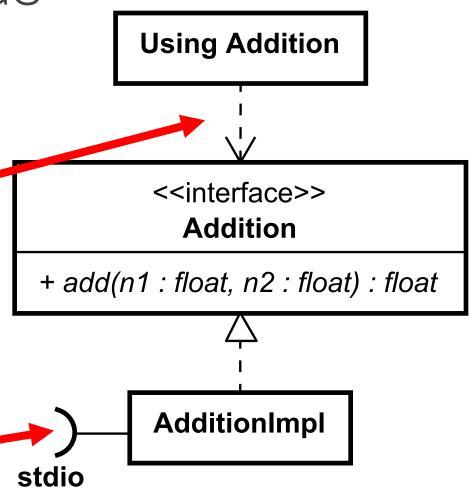
How to include/use header files in C-source files

Including Header files locally in project:

#include "addition.h"

Files that the compiler has an include path to (-I) e.g. Standard Library-files (files outside current project)

#include <stdio.h>



2022-09-05

C-Source Files Implementing Interface (buffer.c)

```
#include "buffer.h"
#include <stdlib.h>
typedef struct bufferStruct {
   uint8_t storage[BUFFER_SIZE];
   uint8 t index;
   uint8 t outdex;
   uint8_t noInBuffer;
}bufferStruct_t;
```

buffer_t buffer_create() {

BufferImpl

stdlib

BufferImpl

storage: uint8_t[BUFFER_SIZE]
index: uint8_t
outdex: uint8_t
noInBuffer: int

Require interface implementation relationship

2022-09-05

Nest relationship

OK

EMPTY

FULL

<<enumeration>>

bufferReturnCode

UNKNOWN BUFFER

return calloc(sizeof(bufferStruct_t), 1);

Input/Output to console

Input from the keyboard (stdio.h)

```
int getchar() // Blocking
// Formatted input from stdin
int scanf( const char* format, ... )
```

<<interface>>

- + getchar() : int
- + scanf(format : char*, ... : void[*]) : int
- + manyMoreInputFunctions()
- + putchar(c : int) : void
- + puts(str : char*) : int
- + printf(format : char*, ... : void) : int
- + manyMoreOutputFunctions()

Output to screen (stdio.h)

```
void putchar(int c) // Single character
int puts( const char* str ) // Zero terminated c-string
// Parameters like printf in Java
int printf( const char* format, ...)
```

Lets shift to Visual Studio

Visual Studio makes your life easier

Short Demo

Exercise Session 2

Follow this video to create your first C-Program in VS 2022: https://www.youtube.com/watch?v=0PUZbgclMzg

ESW1 C-Program Structure - Lars Bech Sørensen, Erland Larsen, Ib Havn

- Do ESW1 Session 1 Exercises
 - but now with Visual Studio 2022
- Do ESW1 Session 2 Exercises
 - can be found in ItsLearning