CHALMERS
UNIVERSITY OF TECHNOLOGY

Machine-Oriented Programming

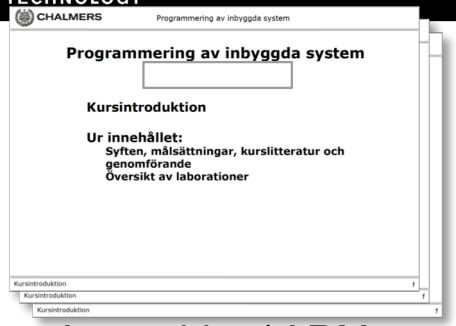
Introduction to C-Programming

Pedro Trancoso
ppedro@chalmers.se

Original slides by Ulf Assarsson

Content

- Topics:
  - Introduction
  - Data types, arrays, scope
  - IDE, .c- / .h/files
  - Preprocessing, compilation, linking
- Exercises:
  - Bitwise operations (AND/OR/XOR)



Assembler / ARM-lectures

5 C-lectures

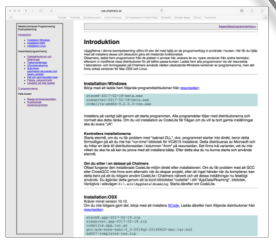
Examples
C – Exercise tasks.
(online)

Same webpage

Textbook

Workbook

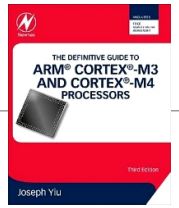
LabPM
(online). 5 labs.



Examples
(exercises online).

If you like:

ARM:



The Definitive Guide to ARM® Cortex® -M3 and Cortex-M4 Processors, Third Edition, Joseph Yiu, ARM Ltd, Cambridge, UK.

C:

- The C programming Language (ANSI-C)
  - https://hassanolity.files.wordpress.com/2013/11/the_c_programming_language_2.pdf
- Vägen till C, Skansholm
- C från början, Skansholm

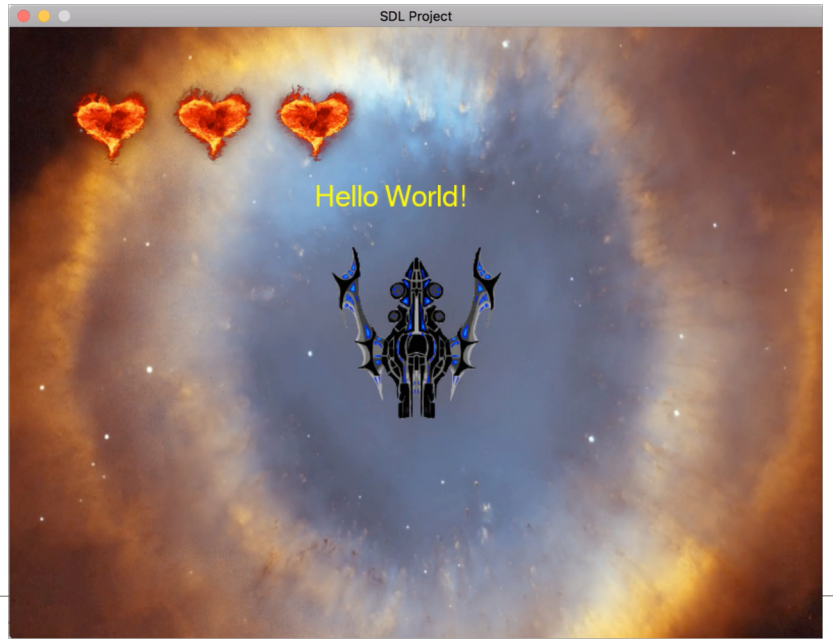
Textbooks?

- The workbook is equivalent to the textbook.
  - Labs – learn how to run things for the target hardware and simulator.
  - Lectures – to understand the workbook (+ lab).
  - Online Example collection:
    - Exercises in assembler/C
- C:
  If you need a book for C:
  - The C programming Language (ANSI-C)
    - https://hassanolity.files.wordpress.com/2013/11/the_c_programming_language_2.pdf
  - Vägen till C, Skansholm, 2011.
  - C från början, Skansholm, 2016.
  - C reference card ANSI (on the course's website under "resurssidan")
    - http://www.cse.chalmers.se/edu/resources/pinsys/ebook/c-refcard.pdf

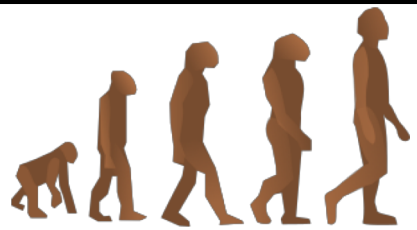
C – Exercises

- Online – see the link on the course homepage for today's C lectures.

  (The first week's exercise also contains links to alternative beginners C exercises online.)



Programming Language Evolution



www.openclipart.org

- 1949: Short Code, 1:st high level language
- 1950s: Autocode, early 50'ies
- 1957: Fortran, IBM
- 1958: Lisp
- 1960: Cobol 60 (Common Business-oriented language)
- 1964: BASIC
- 1960: ALGOL 60 (ALGOrithmic Language 1960)
- 1960s: Simula
- 1969: C
- 1972: Prolog
- 1975: Ada
- 1975: Pascal
- 1978: ML

Top 3 programming languages you used?

Vote View

C Background

C was originally developed by Dennis Ritchie between 1969 and 1973 at Bell Labs, and used to re-implement the Unix operating system

- Modern Languages:
  - C, C++, D, Java, javascript, Objective-C, C#, …
- Machine-oriented programming:
  - Needs a language with pointers to absolute memory addresses
    - Basic, Ada 95 (and older versions), C, C++, C# (with keyword *unsafe*), Objective-C, D, COBOL, Fortran.

    - C - 1969
    - C++ - 1983
    - (Ada– 1995)
    - C# - 2000, strong type checking, garbage collection, obj. oriented, “COOL”.
    - D – 2001
- C is typically the most used language for machine-oriented programming.

Why C (and not stay with Assembly)?

- Fewer lines of code, smaller risk of errors, faster…,
- Why is first C compiled into assembly?
  - Performance optimizations
    - How much faster is a while-loop than recursion?
    - Loop-unroll?
  - Energy consumption
  - security/robustness/risks
    - Code injection
    - Be able to debug
- Able to mix C/asm for programming drivers or performance critical.
  - SIMD: Intel's SSE

After learning C will I still use Assembly? (except for MOP!)

```
a = b – c;

Instead of:
LDR   r3, =b
LDR   r2, [r3]
LDR   r3, =c
LDR   r3, [r3]
SUBS  r2, r2, r3
LDR    r3, =a
STR    r2, [r3]
```

C Overview

- **Lecture 1 - Syntax and Program Structure**:
  - IDE, variables (declaration + assignment), type conversion, ASCII, functions, variable scope, program structure, compiling/linking, arrays
- **Lecture 2 – Pointers and Arrays**:
  - Pointers, absolut addressing (ports), typedef, volatile, #define, arrays of pointers, arrays of arrays
- **Lecture 3 – Structs and Function pointers**:
  - Structs, pointers to structs (arrow notation), array of structs, port addressing with structs, function pointers, structs with function pointers (object-oriented style)
- **Lecture 4 - More program structure as well as Dynamic Memory Allocation**:
  - Scope - static, #extern, (inline), #if/#ifdef, #include guards, enum, union, little/big endian, dynamic memory allocation (malloc/free)
- **Lecture 5 – "Extreme" C**:
  - Real-time loop, C99, obscure C constructions,
  - If we have time: lists, code injection (via stack frames).

C Standards

- 1978, K&R C (Kernighan & Ritchie)
- 1989, C89/C90 (ANSI-C)
- 1999, C99 (Rev. ANSI-standard)
- 2011, C11 (Rev. ANSI-standard)

- 2008, Embedded C (fixed-point arithmetics, basic I/O operations)

Hello world!

```
#include <stdio.h>

int main()
{
   printf(“Hello World!\n");
   return 0;
}
```

C vs. Java

```
struct Course {
    char* name;
    float credits;
    int numberOfParticipants;
};
```

1. C: Does not have classes.
   Has struct for composite data type.
2. Booleans are NOT a type. There is no true/false. 0 is false and a value != 0 is true.
   [you usually define TRUE/FALSE as 1 and 0. Even so, 1 && 1 is not necessarily 1! “implementation-specific for the compiler”]
3. No array boundary checks! 😈😇
4. No exception handling – try/catch
5. No polymorphism or overloading
6. Operations with operands of different types (and type conversion!)
7. Compilation!

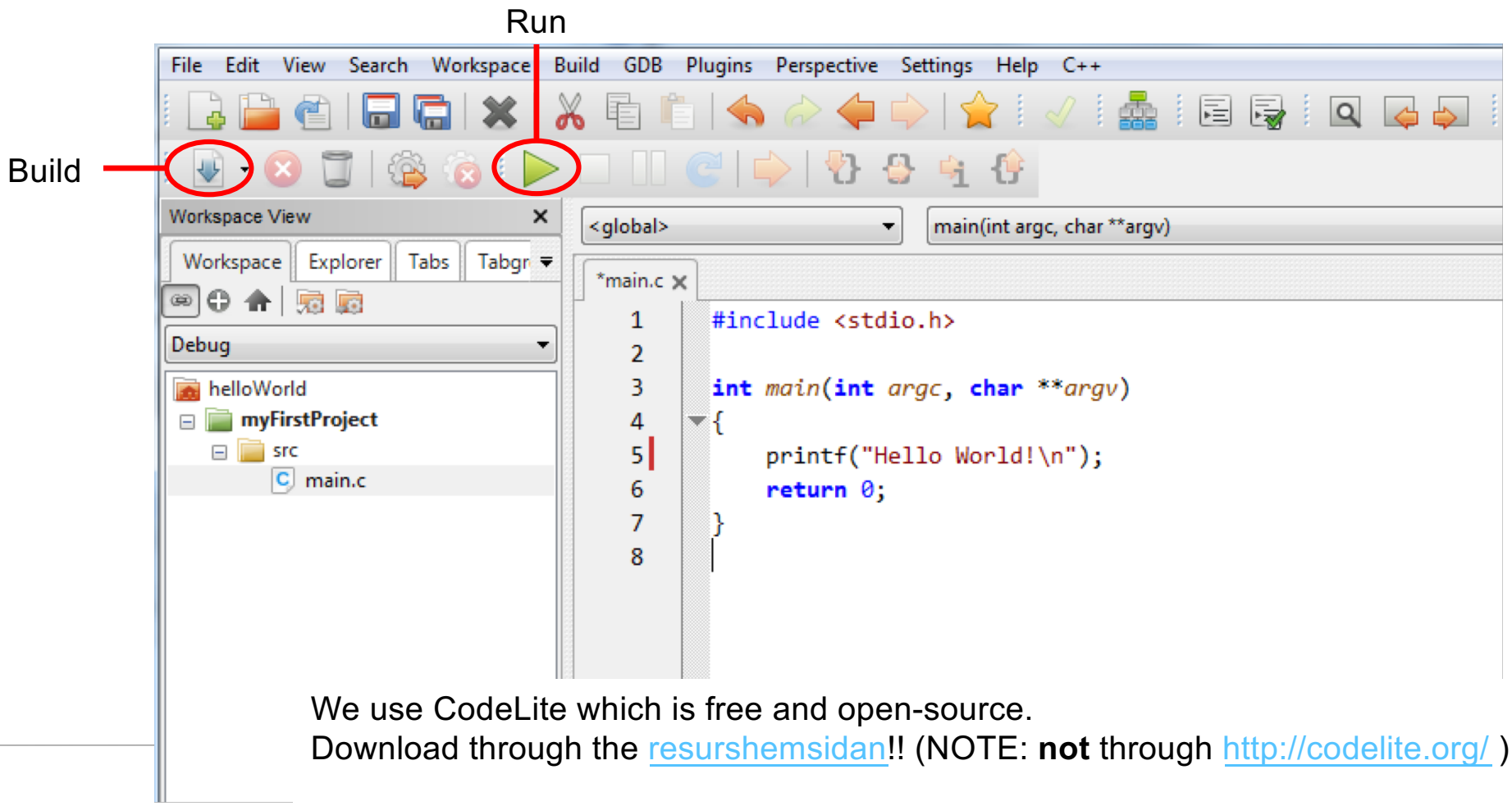
More C

- Type conversion
  - Default: convert narrower format to the wider format (upgrade!)
  - Scope (visibility):
    - global scope,
    - scope in a function
    - Scope in a block.

```
float a;
a = 1.0 / 3;
// a == 0.3333333433 (in MacOS using cc)
```

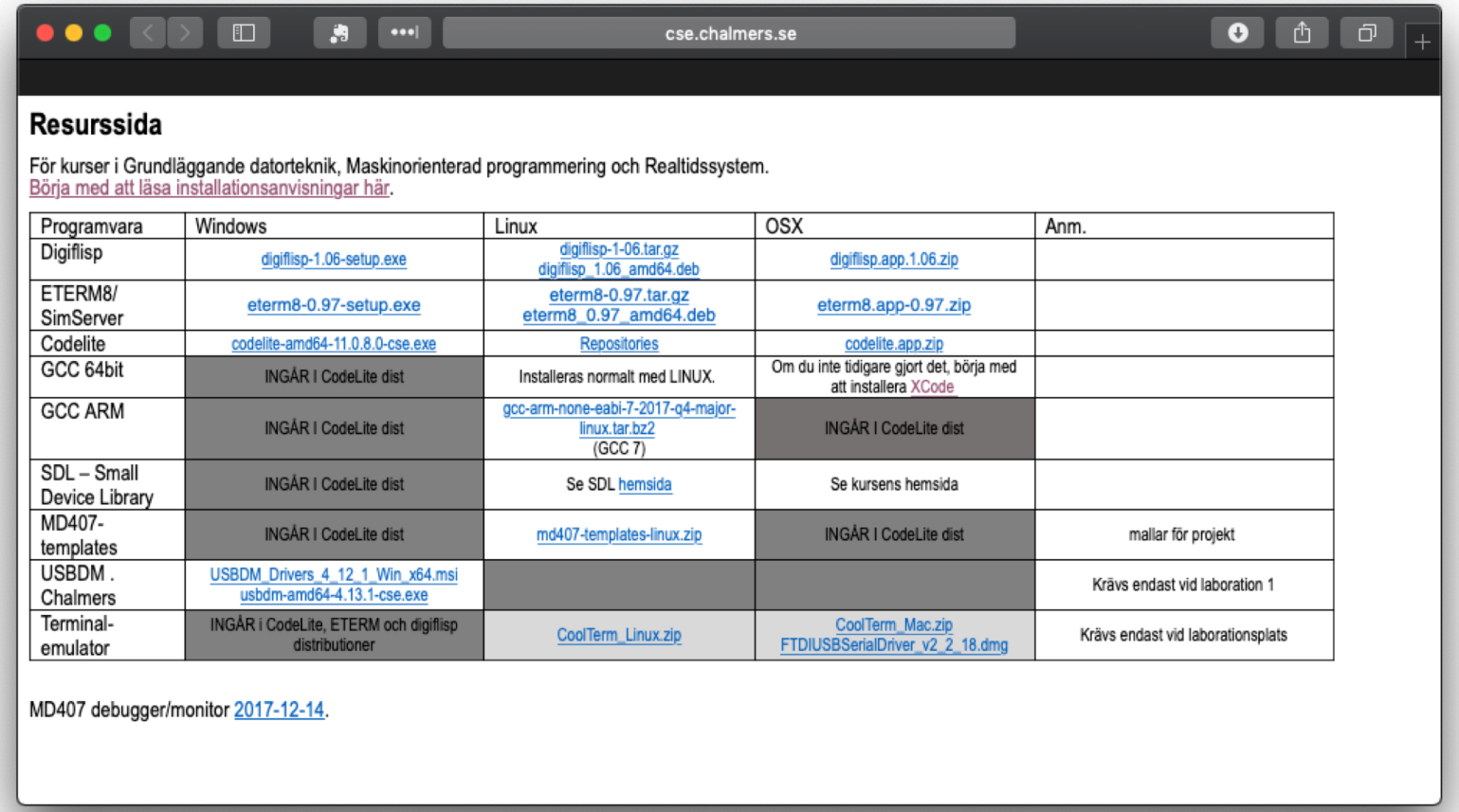
```c
#include <stdio.h>

int x = 10;
int foo(int y)
{
    if( x == y ){
        int z = 4;
        z = z + x + y;
        return z;
    }
    return x;
}
```

Integrated Development Environment (IDE)



We use CodeLite which is free and open-source.
Download through the resurshemsidan!! (NOTE: **not** through http://codelite.org/)

CodeLite – Download through Home->Kursmaterial->Programvaror



Resurssida

För kurser i Grundläggande datorteknik, Maskinorienterad programmering och Realtidssystem.
Börja med att läsa installationsanvisningar här.

Programvara	Windows	Linux	OSX	Anm.
Digiflisp	digiflisp-1.06-setup.exe	digiflisp-1-06.tar.gz digiflisp_1.06_amd64.deb	digiflisp.app.1.06.zip	
ETERM8/ SimServer	eterm8-0.97-setup.exe	eterm8-0.97.tar.gz eterm8_0.97_amd64.deb	eterm8.app-0.97.zip	
Codelite	codelite-amd64-11.0.8.0-cse.exe	Repositories	codelite.app.zip	
GCC 64bit	INGÅR I CodeLite dist	Installeras normalt med LINUX.	Om du inte tidigare gjort det, börja med att installera XCode	
GCC ARM	INGÅR I CodeLite dist	gcc-arm-none-eabi-7-2017-q4-major-linux.tar.bz2 (GCC 7)	INGÅR I CodeLite dist	
SDL – Small Device Library	INGÅR I CodeLite dist	Se SDL hemsida	Se kursens hemsida	
MD407-templates	INGÅR I CodeLite dist	md407-templates-linux.zip	INGÅR I CodeLite dist	mallar för projekt
USBDM . Chalmers	USBDM_Drivers_4_12_1_Win_x64.msi usbdm-amd64-4.13.1-cse.exe			Krävs endast vid laboration 1
Terminal-emulator	INGÅR i CodeLite, ETERM och digiflisp distributioner	CoolTerm_Linux.zip	CoolTerm_Mac.zip FTDIUSBSerialDriver_v2_2_18.dmg	Krävs endast vid laborationsplats

MD407 debugger/monitor 2017-12-14.

From the terminal

```
> gcc -o hello.exe main.c
> hello.exe
Hello World!
>
```


Build


Run

Variables

```c
#include <stdio.h>

int x;

int main()
{
    char y;

    x = 32;
    y = 'a';

    x = x + y;

    printf("x has now the value %d and y has the value %d, the code for character %c\n", x, (int)y, y);
    return 0;
}
```

Output:

```
x has now the value 129 and y has the value 97, the code for character a
```

For printf() parameters, see for example C Reference Guide

Declarations and Assignments

```
#include <stdio.h>

int x;

int main()
{
    char y;

    x = 32;
    y = 'a';

    x = x + y;

    printf("x has now value %d and y has value %d, the code for character %c\n",
    return 0;
}
```

Declarations

Assignments

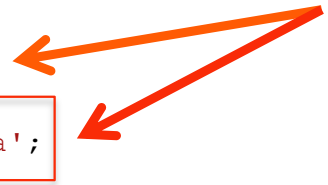
What happens?

```
…
int x;
int y;
y = x + 10;
…
```

A declared variable which has not yet been assigned a value is called **uninitialized**

C89 – Declarations first!

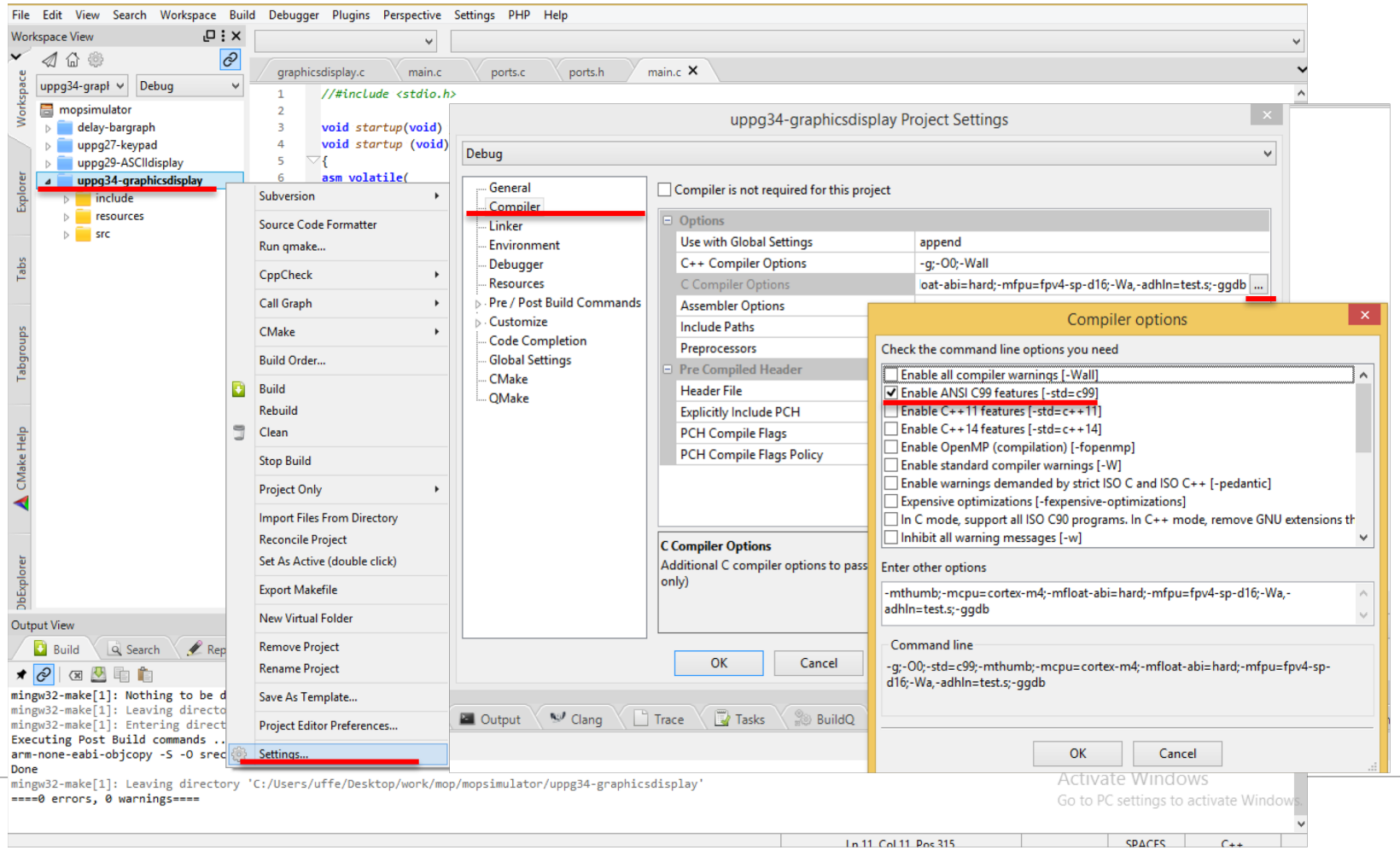
```
#include <stdio.h>

int x;

int main()
{
    x = 32;

    char y = 'a';

    x = x + y;

    printf("x has now value %d and y has value %d, the code for character %c\n", x, (int)y, y);
    return 0;
}
```



In C89 all declarations must come first in the function code.

Sometimes it works anyway (e.g. with gcc), but for our ARM-compiler C99 must be explicitly specified using a compilation flag.

Enable C99


File Edit View Search Workspace Build Debugger Plugins Perspective Settings PHP Help
Workspace View
graphicsdisplay.c main.c ports.c ports.h main.c
//#include <stdio.h>
void startup(void)
void startup (void)
{
asm volatile(
mopsimulator
delay-bargraph
uppg27-keypad
uppg29-ASCIIdisplay
uppg34-graphicsdisplay
include
resources
src
Subversion
Source Code Formatter
Run qmake...
CppCheck
Call Graph
CMake
Build Order...
Build
Rebuild
Clean
Stop Build
Project Only
Import Files From Directory
Reconcile Project
Set As Active (double click)
Export Makefile
New Virtual Folder
Remove Project
Rename Project
Save As Template...
Project Editor Preferences...
Settings...
uppg34-graphicsdisplay Project Settings
Debug
General
Compiler
Linker
Environment
Debugger
Resources
Pre / Post Build Commands
Customize
Code Completion
Global Settings
CMake
QMake
Compiler is not required for this project
Options
Use with Global Settings
append
C++ Compiler Options
-g;-O0;-Wall
C Compiler Options
oat-abi=hard;-mfpu=fpv4-sp-d16;-Wa,-adhln=test.s;-ggdb
Assembler Options
Include Paths
Preprocessors
Pre Compiled Header
Header File
Explicitly Include PCH
PCH Compile Flags
PCH Compile Flags Policy
C Compiler Options
Additional C compiler options to pass
only)
OK
Cancel
Compiler options
Check the command line options you need
Enable all compiler warnings [-Wall]
Enable ANSI C99 features [-std=c99]
Enable C++11 features [-std=c++11]
Enable C++14 features [-std=c++14]
Enable OpenMP (compilation) [-fopenmp]
Enable standard compiler warnings [-W]
Enable warnings demanded by strict ISO C and ISO C++ [-pedantic]
Expensive optimizations [-fexpensive-optimizations]
In C mode, support all ISO C90 programs. In C++ mode, remove GNU extensions th
Inhibit all warning messages [-w]
Enter other options
-mthumb;-mcpu=cortex-m4;-mfloat-abi=hard;-mfpu=fpv4-sp-d16;-Wa,-adhln=test.s;-ggdb
Command line
-g;-O0;-std=c99;-mthumb;-mcpu=cortex-m4;-mfloat-abi=hard;-mfpu=fpv4-sp-d16;-Wa,-adhln=test.s;-ggdb
OK
Cancel
Output View
Build Search
mingw32-make[1]: Nothing to be d
mingw32-make[1]: Leaving directo
mingw32-make[1]: Entering direct
Executing Post Build commands ..
arm-none-eabi-objcopy -S -O srec
Done
mingw32-make[1]: Leaving directory 'C:/Users/uffe/Desktop/work/mop/mopsimulator/uppg34-graphicsdisplay'
====0 errors, 0 warnings====
Output Clang Trace Tasks BuildQ
Activate Windows
Go to PC settings to activate Windows.

Type Conversion

```c
#include <stdio.h>

int x;

int main()
{
    char y;

    x = 32;
    y = 'a';

    x = x + y;

    printf("x has now value %d and y has value %d, the code for character %c\n", x, (int)y, y);
    return 0;
}
```

Implicit type conversion

Explicit type conversion

Type conversion is also called ***cast***, and you can say that you do ***casting***.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
20	14	[DEVICE CONTROL 4]	52	34	4						
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5						
22	16	[SYNCHRONOUS IDLE]	54	36	6						
23	17	[ENG OF TRANS. BLOCK]	55	37	7						
25	19	[END OF MEDIUM]	57	39	9						
26	1A	[SUBSTITUTE]	58	3A	:						
27	1B	[ESCAPE]	59	3B	;						
28	1C	[FILE SEPARATOR]	60	3C	<						
30	1E	[RECORD SEPARATOR]	62	3E	>						
31	1F	[UNIT SEPARATOR]	63	3F	?						

ASCII –Is there only one ASCII-table?

There are several different ASCII **extensions** (over 128 values), depending on the language.

ISO 8859-1, also called *ISO Latin 1*, ISO 8859-2 for Eastern European languages, and ISO 8859-5 for Cyrillic languages

Demo

```c
#include <stdio.h>

int main()
{
   char x = 7;
   printf("%c\n", x);
   return 0;
}
```

To install CodeLite and create a project:
see Hemuppgifter C-programmering

- For Windows choose
  compiler: GCC
  Debugger: GNU gdb.
- For Mac OS choose
  compiler: GCC
  Debugger: LLDB.

Arrays

```
int   a[] = {3, 2, 1, 0};
int   b[5];
float c[6] = {2.0f, 1.0f};

int main()
{
    a[0] = 5;
    b[4] = a[2];
    c[3] = 3.0f;
    return 0;
}
```

Strings

Strings are null-terminated character arrays

```
#include <stdio.h>

char name1[] = {'E', 'm', 'i', 'l', '\0'};
char name2[] = "Emilia";
char name3[];

int main()
{
    printf("name1: %s \n", name1);
    printf("name2: %s \n", name2);
    printf("sizeof (name2): %d \n", sizeof(name2));

    return 0;
}
```

Output:

```
name1: Emil
name2: Emilia
sizeof(name2): 7
```

7 what?

Exercise:

Have an array where position 0 you have the total number of ‘a’ in a text, in position 1 you have the total number of ‘b’, etc.

```
int   countChars[100];

int main() {
    ...
    if( ch == ‘a’ )
        countChars[0]++;
    else if( ch == ‘b’ )
        countChars[1]++;
    else if( ch == ‘c’ )
    ...
   return 0;
}
```

Better? Single line for all cases?

```
int   countChars[100];

int main() {
    ...
    countChars[ch-’a’]++;
    ...
   return 0;
}
```

Functions

```
#include <stdio.h>

int foo(int x, char y)
{
    int sum = 0;

    while(y > 0) {
        sum += x*y;
        y--;
    }

    return sum;
}
```

Arguments (or parameters)

Return value of return type

Arguments are ”pass-by value”

```
int var1;
char var2 = 7;
var1 = foo(5, var2);
```

var2 still has the value 7
after the function call

Visibility / Scope

- Global visibility (global scope)
- File visibility (file scope)
- Local visibility (e.g. function scope)

Visibility

```
#include <stdio.h>

char x;

int foo()
{
  // x is visible
  // y is not visible
}

char y;
```

Visibility at the function level

```
#include <stdio.h>

char x;

int foo(float x)
{
  // argument x (float) is visible
}
```

Visibility at the function level

```
#include <stdio.h>

char x;

int foo()
{
   int x = 4;
   return x;
}
```

Which visibility has the highest priority?

```
#include <stdio.h>

int x;

int foo(int x) {
    if( x == 0 ){
        int x = 4;
        return x;
    }
    return x;
}

int main() {
    x = 1;
    x = foo(0);
    printf("x is %d\n", x);
    return 0;
}
```

What is the output (value of x)?

Function Prototype

```
#include <stdio.h>

// function prototype
int foo(int x);

int main()
{
   printf("x is %d\n", foo(0));
   return 0;
}

int foo(int x)
{
   // function body
}
```

Where is the prototype for ***printf***?

Program Structure

If you have long programs then you should not put everything in a single main.c file. You should split the functionalities between different files. Header h-files should include function prototypes (and user-defined type definitions). Different c-files can include different functions grouped by topic.

```c
// main.c
#include <stdio.h>
#include "foo.h"

int main()
{
    printf("x is %d\n", foo(0));
    return 0;
}
```

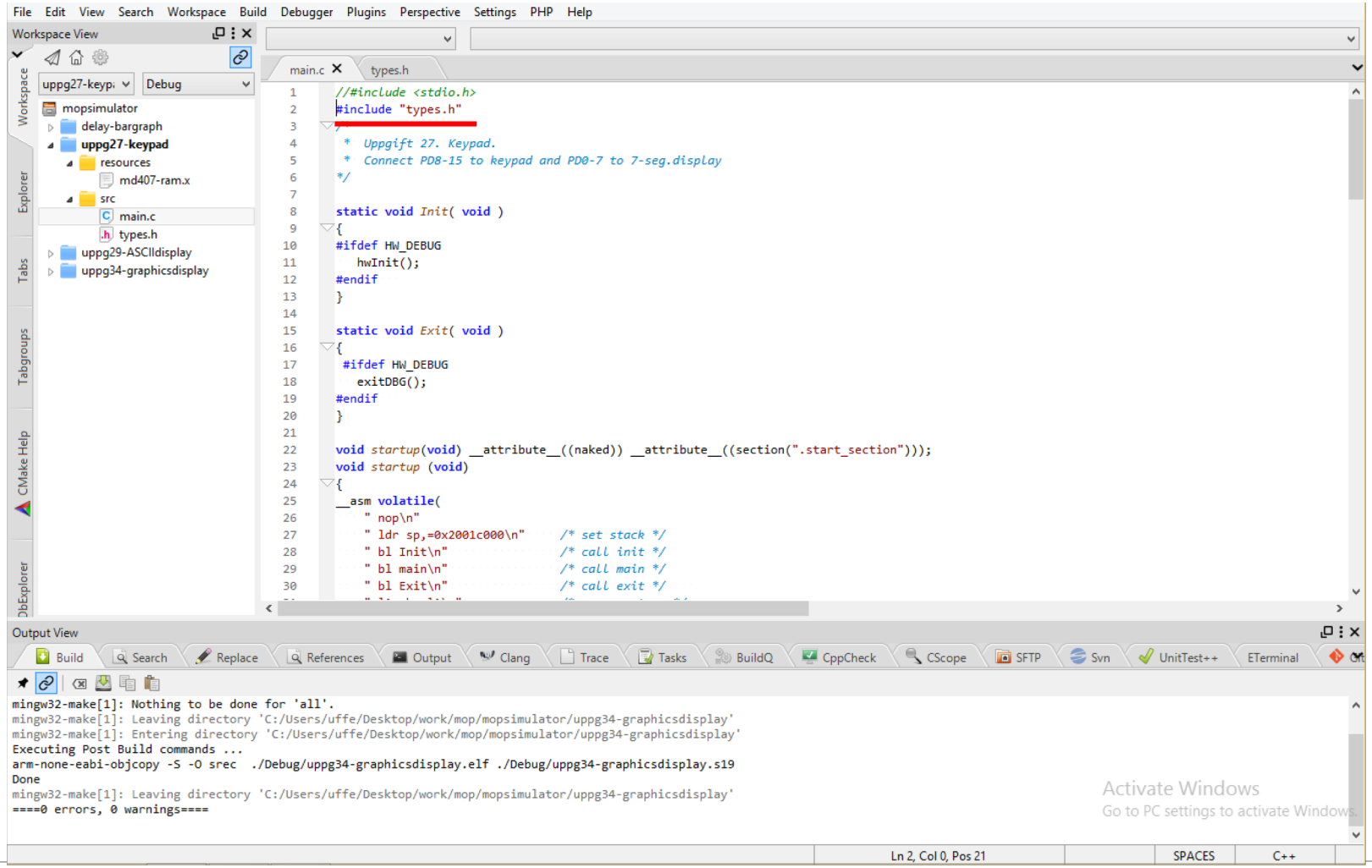
c-file
Includes header files

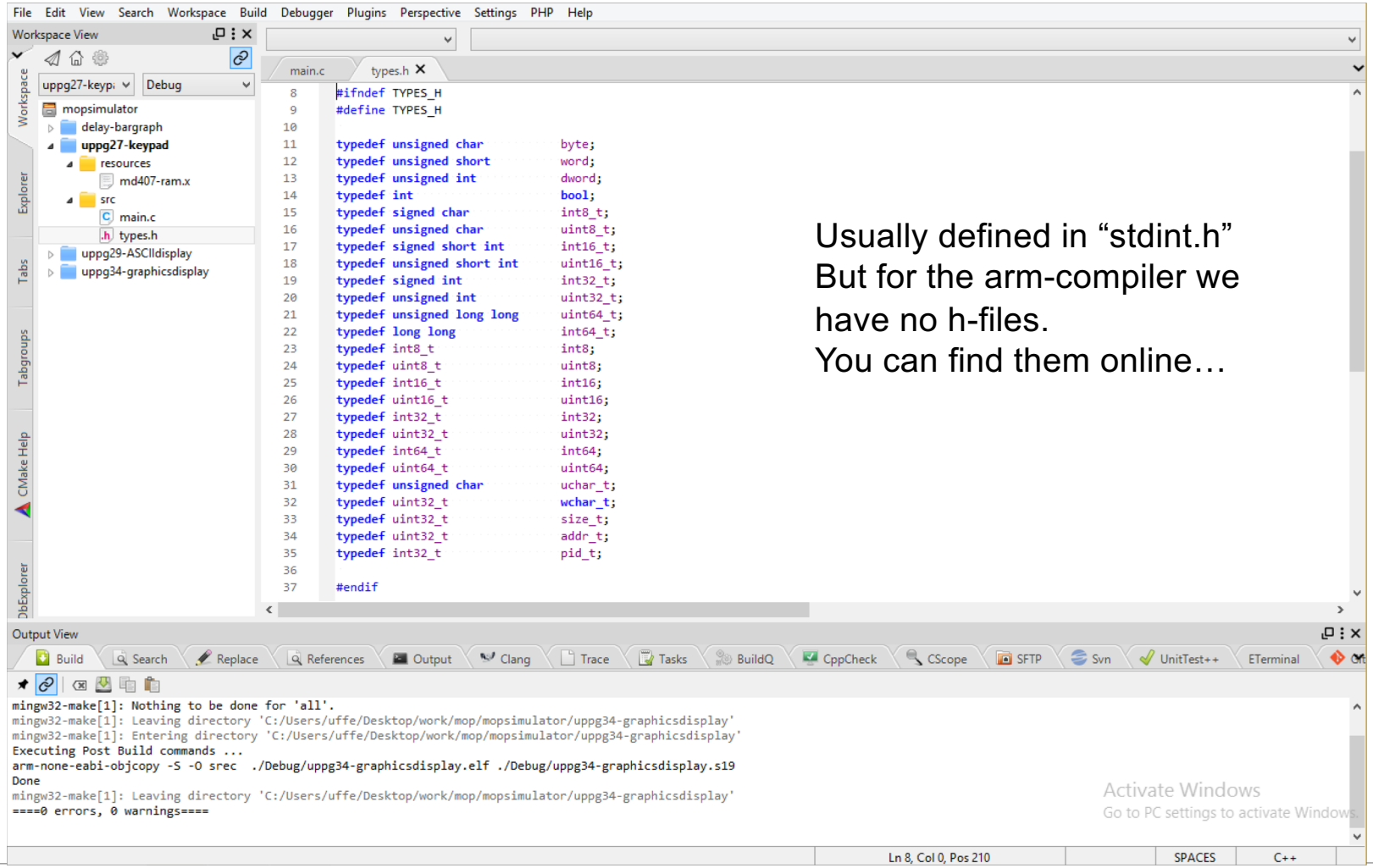
```c
// foo.h

int foo(int x);
```

header-file
Contains the function prototypes

```c
// foo.c
#include <stdlib.h>

int foo(int x)
{
    if( x == 0 ){
        int x = 4;
        return x;
    }
    return x;
}
```

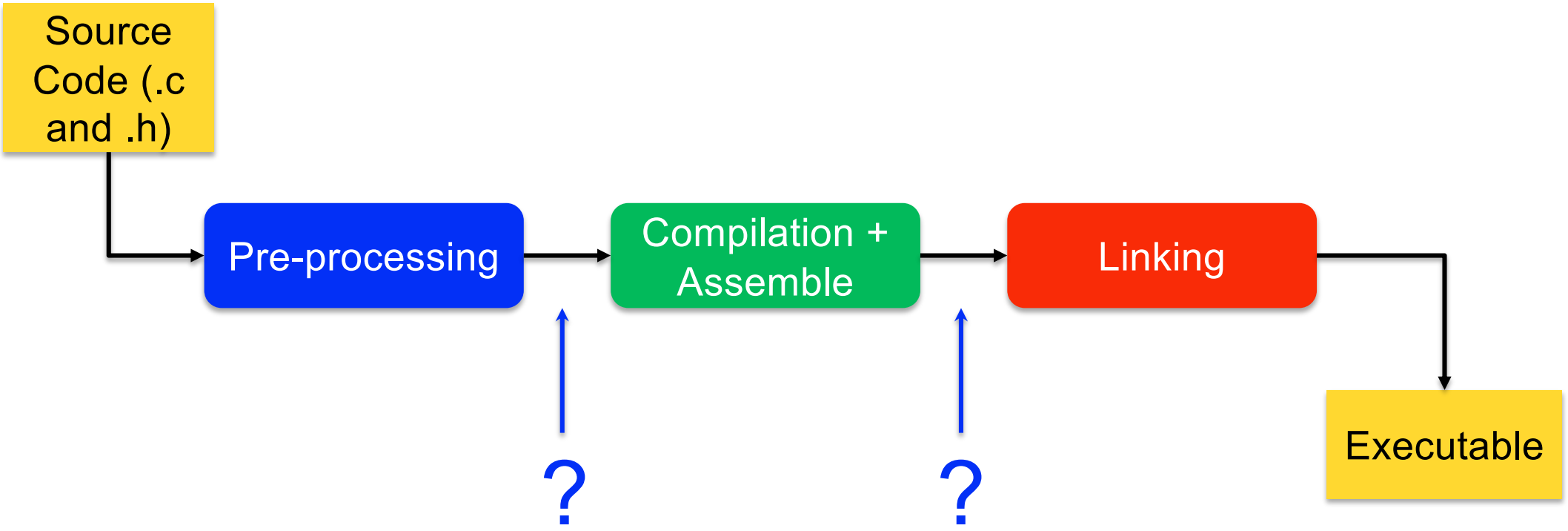
c-file


File Edit View Search Workspace Build Debugger Plugins Perspective Settings PHP Help
Workspace View
uppg27-keypa
Debug
mopsimulator
delay-bargraph
uppg27-keypad
resources
md407-ram.x
src
main.c
types.h
uppg29-ASCIIdisplay
uppg34-graphicsdisplay
main.c
types.h
//#include <stdio.h>
#include "types.h"
*/
* Uppgift 27. Keypad.
* Connect PD8-15 to keypad and PD0-7 to 7-seg.display
*/
static void Init(void)
{
#ifdef HW_DEBUG
hwInit();
#endif
}
static void Exit(void)
{
#ifdef HW_DEBUG
exitDBG();
#endif
}
void startup(void) __attribute__((naked)) __attribute__((section(".start_section")));
void startup (void)
{
__asm volatile(
" nop\n"
" ldr sp,=0x2001c000\n" /* set stack */
" bl Init\n" /* call init */
" bl main\n" /* call main */
" bl Exit\n" /* call exit */
Output View
Build
Search
Replace
References
Output
Clang
Trace
Tasks
BuildQ
CppCheck
CScope
SFTP
Svn
UnitTest++
ETerminal
mingw32-make[1]: Nothing to be done for 'all'.
mingw32-make[1]: Leaving directory 'C:/Users/uffe/Desktop/work/mop/mopsimulator/uppg34-graphicsdisplay'
mingw32-make[1]: Entering directory 'C:/Users/uffe/Desktop/work/mop/mopsimulator/uppg34-graphicsdisplay'
Executing Post Build commands ...
arm-none-eabi-objcopy -S -O srec ./Debug/uppg34-graphicsdisplay.elf ./Debug/uppg34-graphicsdisplay.s19
Done
mingw32-make[1]: Leaving directory 'C:/Users/uffe/Desktop/work/mop/mopsimulator/uppg34-graphicsdisplay'
====0 errors, 0 warnings====
Activate Windows
Go to PC settings to activate Windows.
Ln 2, Col 0, Pos 21
SPACES
C++



Usually defined in “stdint.h”
But for the arm-compiler we have no h-files.
You can find them online…

From source code to Executable



Pre-Processor

```
// main.c
#include <stdio.h>
#include "foo.h"

#define MAX_SCORE 100
#define SQUARE(x)  (x)*(x)

int main()
{
   printf(”Highest possible points are %d\n", MAX_SCORE);
   printf(”Square of 3 is %d\n", SQUARE(1+2));
   printf("x is %d", foo(0));
   return 0;
}
```

Copy-paste from header files

String find-and-replace

The pre-processor works on the source code at the text-level

Note: If you want to check the output of the pre-processor phase run gcc –E!

Compiling

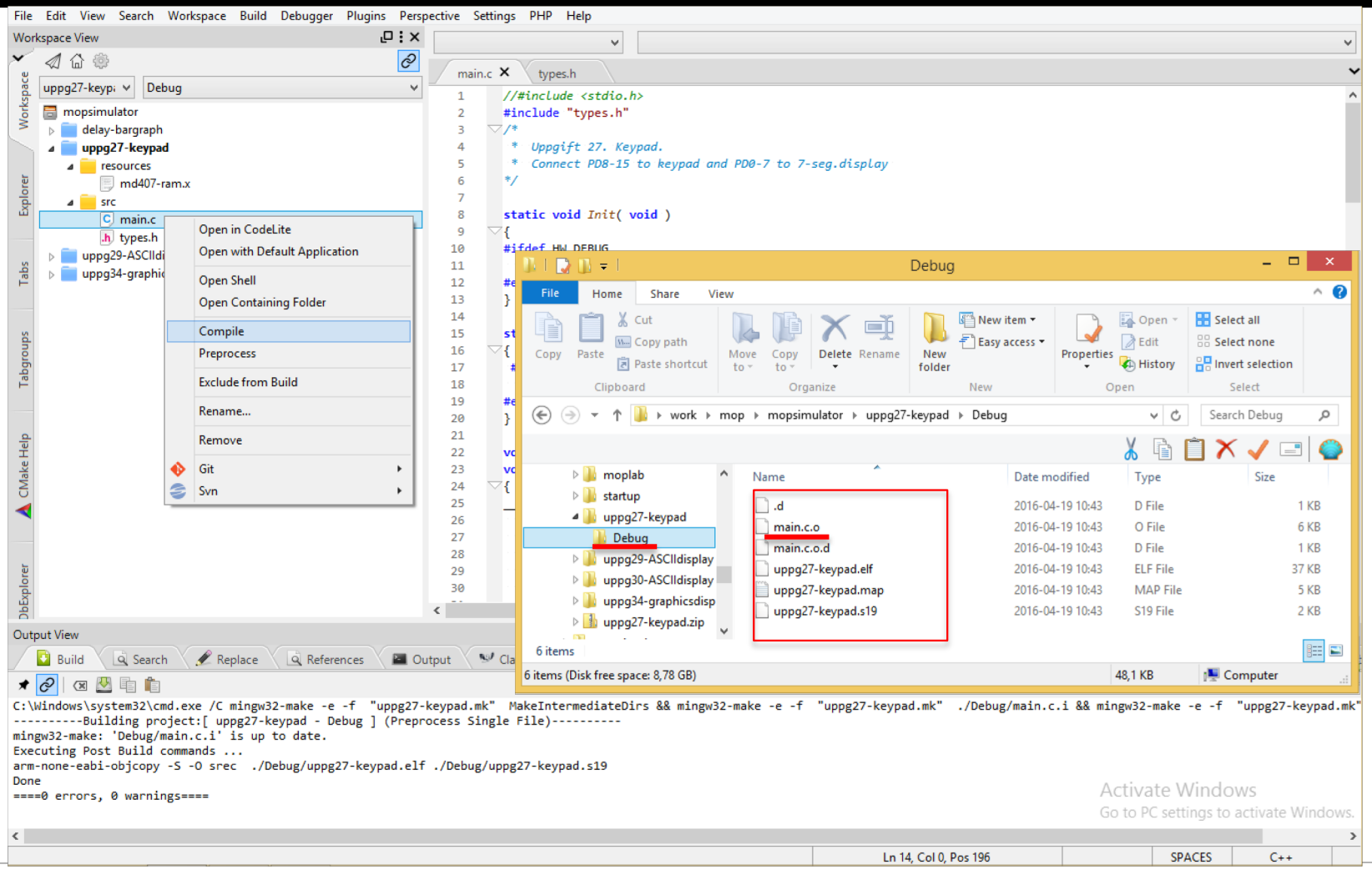
What happens if in file **one.c** function main calls function **foo** that is in file **two.c**?

- Processes one .c-file at a time:
  - Creates an object file (.o-file) (e.g. gcc -c), per .c-file, containing:
    - Machine code instructions
    - Symbols for addresses
      - For functions/variables in the object file.
      - For functions/variables in another object file/library.

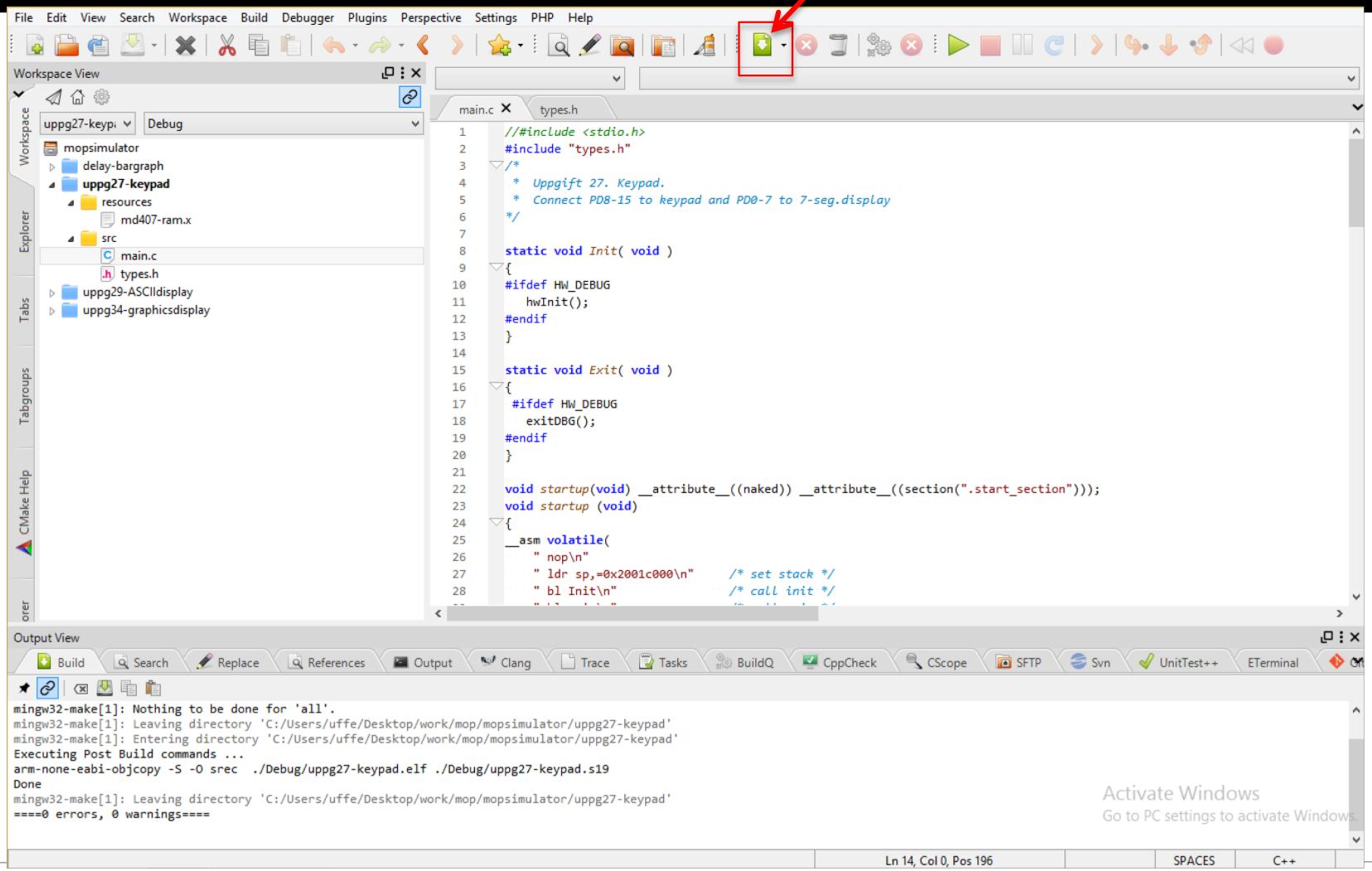
Note: .o file is in binary format so you you need to generate the assembly explicitly to see the instructions and labels generated (e.g. gcc –S)

main.s:

```
myFkn:      mov   r3, #0
            …
            bx    lr

main:  bl   myFkn
       bx   lr
```


File Edit View Search Workspace Build Debugger Plugins Perspective Settings PHP Help
Workspace View
mopsimulator
delay-bargraph
uppg27-keypad
resources
md407-ram.x
src
main.c
types.h
Open in CodeLite
Open with Default Application
Open Shell
Open Containing Folder
Compile
Preprocess
Exclude from Build
Rename...
Remove
Git
Svn
main.c
types.h
//#include <stdio.h>
#include "types.h"
/*
* Uppgift 27. Keypad.
* Connect PD8-15 to keypad and PD0-7 to 7-seg.display
*/
static void Init(void)
{
Debug
File Home Share View
Copy Paste Cut Copy path Paste shortcut
Move to Copy to Delete Rename New folder New item Easy access
Properties Open Edit History
Select all Select none Invert selection
work mop mopsimulator uppg27-keypad Debug
Search Debug
moplab
startup
uppg27-keypad
Debug
uppg29-ASCIIdisplay
uppg30-ASCIIdisplay
uppg34-graphicsdisp
uppg27-keypad.zip
Name
.d
main.c.o
main.c.d
uppg27-keypad.elf
uppg27-keypad.map
uppg27-keypad.s19
Date modified Type Size
2016-04-19 10:43 D File 1 KB
2016-04-19 10:43 O File 6 KB
2016-04-19 10:43 D File 1 KB
2016-04-19 10:43 ELF File 37 KB
2016-04-19 10:43 MAP File 5 KB
2016-04-19 10:43 S19 File 2 KB
6 items
6 items (Disk free space: 8,78 GB)
48,1 KB
Computer
Output View
Build Search Replace References Output
C:\Windows\system32\cmd.exe /C mingw32-make -e -f "uppg27-keypad.mk" MakeIntermediateDirs && mingw32-make -e -f "uppg27-keypad.mk" ./Debug/main.c.i && mingw32-make -e -f "uppg27-keypad.mk"
----------Building project:[uppg27-keypad - Debug] (Preprocess Single File)----------
mingw32-make: 'Debug/main.c.i' is up to date.
Executing Post Build commands ...
arm-none-eabi-objcopy -S -O srec ./Debug/uppg27-keypad.elf ./Debug/uppg27-keypad.s19
Done
====0 errors, 0 warnings====
Activate Windows
Go to PC settings to activate Windows.
Ln 14, Col 0, Pos 196
SPACES
C++

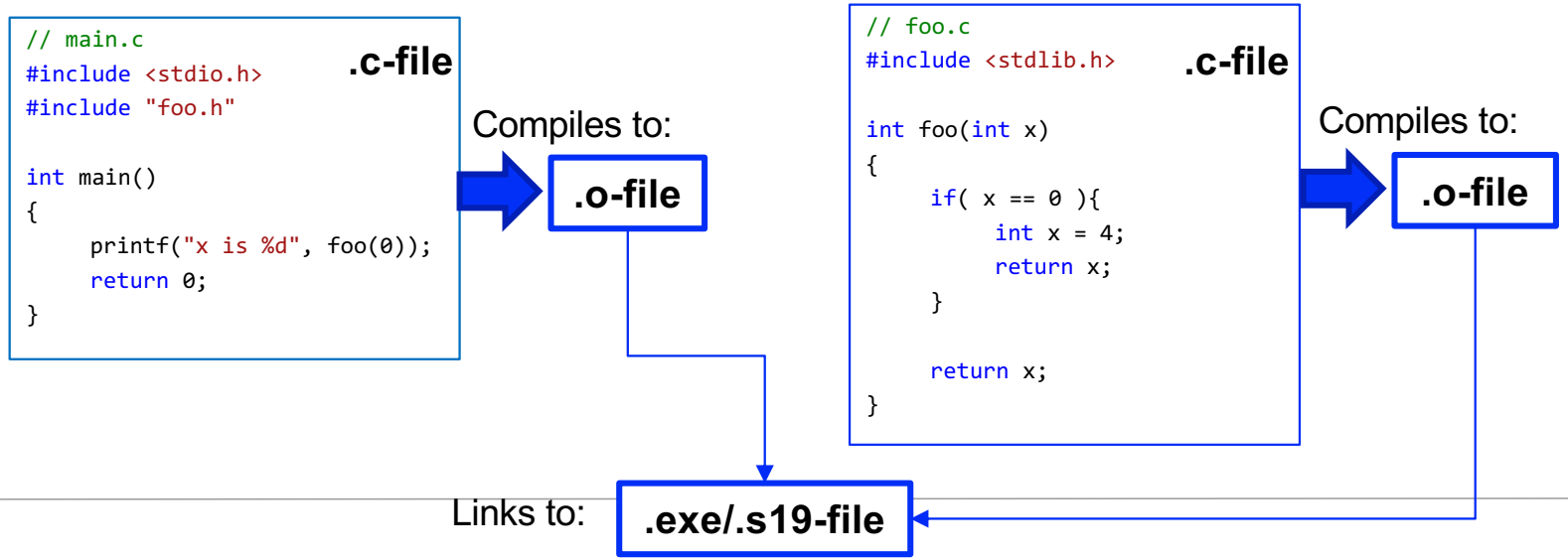
Build

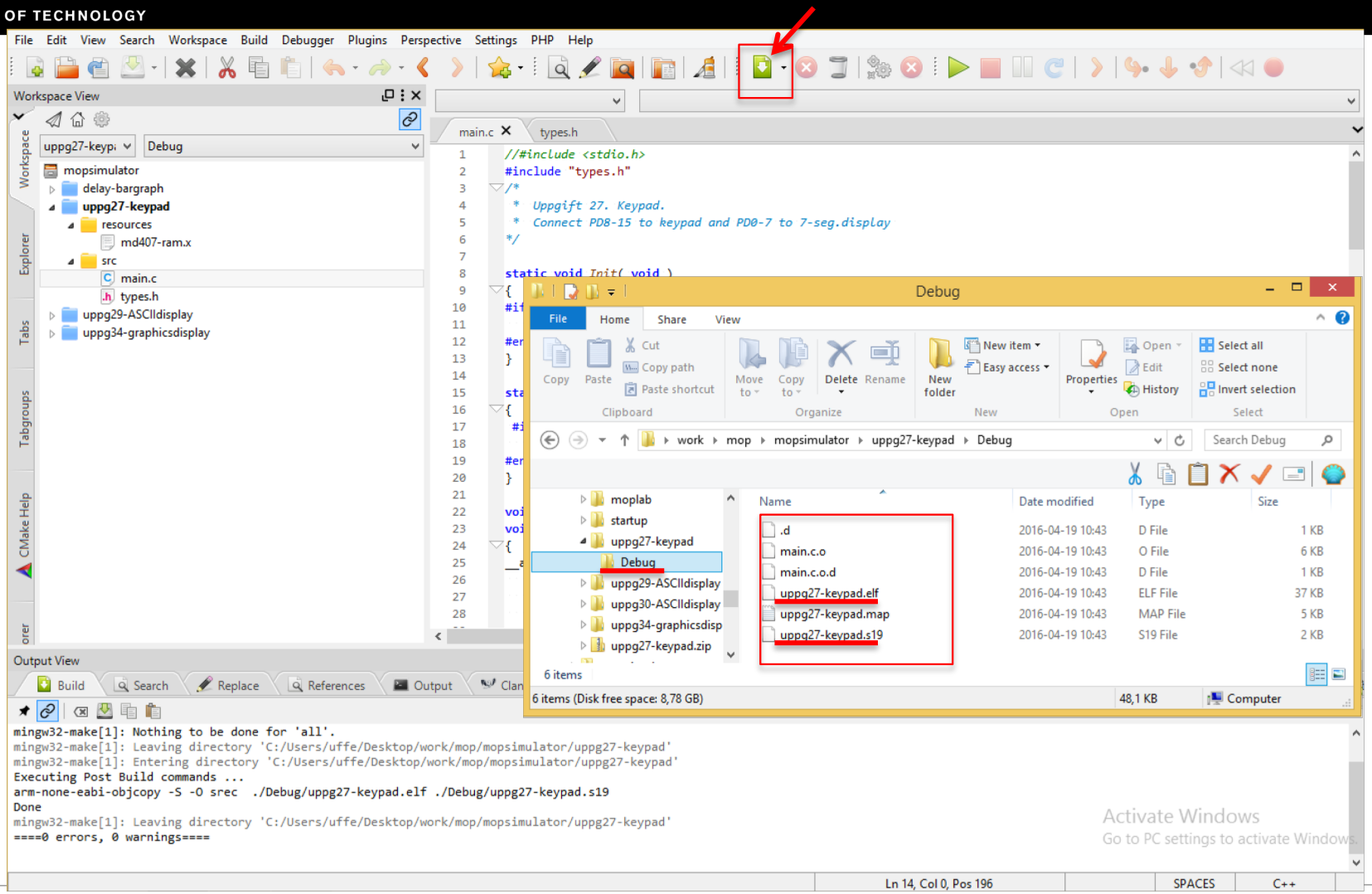

File Edit View Search Workspace Build Debugger Plugins Perspective Settings PHP Help
Workspace View
uppg27-keypa
Debug
mopsimulator
delay-bargraph
uppg27-keypad
resources
md407-ram.x
src
main.c
types.h
uppg29-ASCIIdisplay
uppg34-graphicsdisplay
main.c
types.h
1 //#include <stdio.h>
2 #include "types.h"
3 /*
4 * Uppgift 27. Keypad.
5 * Connect PD8-15 to keypad and PD0-7 to 7-seg.display
6 */
7
8 static void Init(void)
9 {
10 #ifdef HW_DEBUG
11 hwInit();
12 #endif
13 }
14
15 static void Exit(void)
16 {
17 #ifdef HW_DEBUG
18 exitDBG();
19 #endif
20 }
21
22 void startup(void) __attribute__((naked)) __attribute__((section(".start_section")));
23 void startup (void)
24 {
25 __asm volatile(
26 " nop\n"
27 " ldr sp,=0x2001c000\n" /* set stack */
28 " bl Init\n" /* call init */
Output View
Build Search Replace References Output Clang Trace Tasks BuildQ CppCheck CScope SFTP Svn UnitTest++ ETerminal
mingw32-make[1]: Nothing to be done for 'all'.
mingw32-make[1]: Leaving directory 'C:/Users/uffe/Desktop/work/mop/mopsimulator/uppg27-keypad'
mingw32-make[1]: Entering directory 'C:/Users/uffe/Desktop/work/mop/mopsimulator/uppg27-keypad'
Executing Post Build commands ...
arm-none-eabi-objcopy -S -O srec ./Debug/uppg27-keypad.elf ./Debug/uppg27-keypad.s19
Done
mingw32-make[1]: Leaving directory 'C:/Users/uffe/Desktop/work/mop/mopsimulator/uppg27-keypad'
====0 errors, 0 warnings====
Activate Windows
Go to PC settings to activate Windows.
Ln 14, Col 0, Pos 196
SPACES
C++

What is the difference between ***static*** and ***dynamic*** linking?

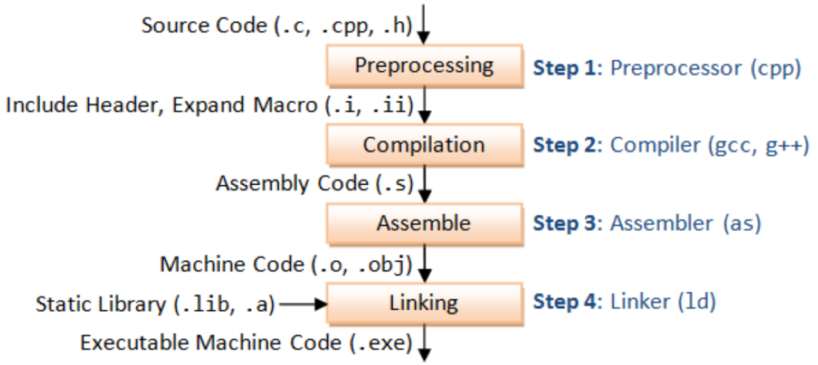
Linking

- Merge multiple object files into one executable file (.exe/.s19)
- Translate the symbols to (relative) adresses




File Edit View Search Workspace Build Debugger Plugins Perspective Settings PHP Help
Workspace View
uppg27-keypa
Debug
mopsimulator
delay-bargraph
uppg27-keypad
resources
md407-ram.x
src
main.c
types.h
uppg29-ASCIIdisplay
uppg34-graphicsdisplay
main.c
types.h
//#include <stdio.h>
#include "types.h"
/*
* Uppgift 27. Keypad.
* Connect PD8-15 to keypad and PD0-7 to 7-seg.display
*/
static void Init(void)
Debug
File Home Share View
Copy Paste Cut Copy path Paste shortcut
Move to Copy to Delete Rename
New folder New item Easy access
Properties Open Edit History
Select all Select none Invert selection
Clipboard Organize New Open Select
work mop mopsimulator uppg27-keypad Debug
Search Debug
moplab
startup
uppg27-keypad
Debug
uppg29-ASCIIdisplay
uppg30-ASCIIdisplay
uppg34-graphicsdisp
uppg27-keypad.zip
Name
Date modified
Type
Size
.d 2016-04-19 10:43 D File 1 KB
main.c.o 2016-04-19 10:43 O File 6 KB
main.c.o.d 2016-04-19 10:43 D File 1 KB
uppg27-keypad.elf 2016-04-19 10:43 ELF File 37 KB
uppg27-keypad.map 2016-04-19 10:43 MAP File 5 KB
uppg27-keypad.s19 2016-04-19 10:43 S19 File 2 KB
6 items
6 items (Disk free space: 8,78 GB)
48,1 KB
Computer
Output View
Build Search Replace References Output
mingw32-make[1]: Nothing to be done for 'all'.
mingw32-make[1]: Leaving directory 'C:/Users/uffe/Desktop/work/mop/mopsimulator/uppg27-keypad'
mingw32-make[1]: Entering directory 'C:/Users/uffe/Desktop/work/mop/mopsimulator/uppg27-keypad'
Executing Post Build commands ...
arm-none-eabi-objcopy -S -O srec ./Debug/uppg27-keypad.elf ./Debug/uppg27-keypad.s19
Done
mingw32-make[1]: Leaving directory 'C:/Users/uffe/Desktop/work/mop/mopsimulator/uppg27-keypad'
====0 errors, 0 warnings====
Activate Windows
Go to PC settings to activate Windows.
Ln 14, Col 0, Pos 196
SPACES
C++

1.4 GCC Compilation Process



GCC compiles a C/C++ program into executable in 4 steps as shown in the above diagram. For example, a "`gcc -o hello.exe hello.c`" is carried out as follows:

1. Pre-processing: via the GNU C Preprocessor (`cpp.exe`), which includes the headers (`#include`) and expands the macros (`#define`).

   ```
   > cpp hello.c > hello.i
   ```

   The resultant intermediate file "`hello.i`" contains the expanded source code.

2. Compilation: The compiler compiles the pre-processed source code into assembly code for a specific processor.

   ```
   > gcc -S hello.i
   ```

   The `-S` option specifies to produce assembly code, instead of object code. The resultant assembly file is "`hello.s`".

3. Assembly: The assembler (`as.exe`) converts the assembly code into machine code in the object file "`hello.o`".

   ```
   > as -o hello.o hello.s
   ```

4. Linker: Finally, the linker (`ld.exe`) links the object code with the library code to produce an executable file "`hello.exe`".

   ```
   > ld -o hello.exe hello.o ...libraries...
   ```

Verbose Mode (`-v`)

You can see the detailed compilation process by enabling `-v` (verbose) option. For example,

```
> gcc -v hello.c -o hello.exe
```


Se: https://www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html

Arithmetic Operators

Basic assignment		a = b
Addition		a + b
Subtraction		a - b
Unary plus (integer promotion)		+a
Unary minus (additive inverse)		-a
Multiplication		a * b
Division		a / b
Modulo (integer remainder)		a % b
Increment	Prefix	++a
	Postfix	a++
Decrement	Prefix	--a
	Postfix	a--

http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B#Arithmetic_operators

Comparison Operators

Equal to	a **==** b
Not equal to	a **!=** b
Greater than	a **>** b
Less than	a **<** b
Greater than or equal to	a **>=** b
Less than or equal to	a **<=** b

http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B#Comparison_operators.2Frelational_operators

Logical Operators

Logical negation (NOT)	**!**a
Logical AND	a **&&** b
Logical OR	a **||** b

http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B#Logical_operators

Compound Assignment Operators

Operator name	Syntax	Meaning
Addition assignment	a **+=** b	a **=** a **+** b
Subtraction assignment	a **-=** b	a **=** a **-** b
Multiplication assignment	a ***=** b	a **=** a ***** b
Division assignment	a **/=** b	a **=** a **/** b
Modulo assignment	a **%=** b	a **=** a **%** b
Bitwise AND assignment	a **&=** b	a **=** a **&** b
Bitwise OR assignment	a **|=** b	a **=** a **|** b
Bitwise XOR assignment	a **^=** b	a **=** a **^** b
Bitwise left shift assignment	a **<<=** b	a **=** a **<<** b
Bitwise right shift assignment	a **>>=** b	a **=** a **>>** b

http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B#Compound_assignment_operators

If-else statements

```
int x = -4;

if( x == 0 ){
    // ...
}

if( x ){
    // ...
}
else {
    // ...
}
```

Evaluates to false so if body does not execute

Evaluates to true so if body is executed (x != 0)

- **Zero** is considered **false**.
- Anything that is **not zero** is considered **true**.

Loops

```
int x = 5;

while( x!=0 )
    x--;
```

```
int x = 5;

while( x )
    x--;
```

```
int x;

for( x=5; x; )
    x--;
```

Three equivalent loops.

```
for(int i=0; i<5; i++ ) {
     if(…)
        continue; // jumps till next iteration
     if(…)
        break; // exists the loop.
     …
}
```

`break` and `continue`

Bitwise Operators

Bitwise NOT	**~**a
Bitwise AND	a **&** b
Bitwise OR	a **|** b
Bitwise XOR	a **^** b
Bitwise left shift	a **<<** b
Bitwise right shift	a **>>** b

http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B#Bitwise_operators

How we use Bitwise operations

- Bitwise OR is typically used to set certain bits.
- Bitwise AND is typically used to reset certain bits.
- Bitwise XOR is typically used to invert certain bits.
- Bitwise NOT used to invert all bits.

Bitwise operations: Examples part 1

isSetLSB, isSetMSB, isSetN

```
int
isSetLSB( int x ) {
  if( x & 0x00000001 )
    return( 1 );
  else
    return( 0 );
}
```

```
int
isSetMSB( int x ) {
  return( x & 0x80000000 );
}
```

Is correct?

```
int
isSetMSB( int x ) {
  return( ( x & 0x80000000 ) == 0 ? 0 : 1 );
}
```

```
int
isSetN( int x, int n ) {
  int mask = 0x00000001;
  mask = mask << n;
  return( ( x & mask ) == 0 ? 0 : 1 );
}
```

```
NOTE:
0x – prefix for hexadecimal
0x0001 << 1 → 0x0002 (0000 0010)
0x0001 << 3 → 0x0008 (0000 1000)
0x0001 << 4 → 0x0010 (0001 0000)

0x0A52 = 0000 1010 0101 0010
0x0A52 << 2 → 0x2948 (0010 1001 0100 1000)
```

Bitwise operations: Examples part 2

countOnes, set mask bits (or reset mask bits)

```
int
countOnes( int x ) {
  int mask = 0x00000001;
  int count = 0;

  for( int i = 0; i < 32; i++ )
    if( x & (mask << i) ) count++;

  return count;
}
```

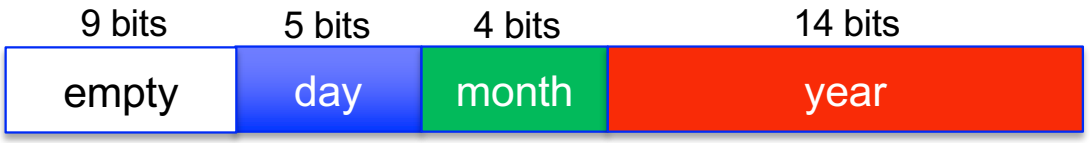
```
#define BIT0 0x00000001
#define BIT1 0x00000002
#define BIT2 0x00000004
#define BIT3 0x00000008
#define BIT4 0x00000010

…

int main( void ) {
  int mask = 0, value1, value2;
  …
  mask = BIT0 | BIT2 | BIT4;
  value1 = value1 | mask; //set
                                  //reset
}
```

Bitwise operations: Assignment!

Packing different values into a single variable:

- Pack and Unpack a date (DAY/MONTH/YEAR) into a word (integer) variable

Max day = 31 -> log2(31) = 4.9 -> 5 bits
Max month = 12 -> log2(12) = 3.6 -> 4 bits
Max year = 9999 -> log2(9999) = 13.3 -> 14 bits



Next C Lecture

- **Pointers, pointers, pointers!**
- **Arrays, Port addressing**

Next Lecture:
- ARM Assembly 1

Assignment:
- Go over online exercises for C for Lecture 1
- Do a program to pack and unpack a date into a single word (4 byte) integer value (do it as teams and submit your solution .c file in canvas)