

Grundläggande C-programmering del 1

- För maskinorienterad programmering

Ulf Assarsson

Läromoment:

- Datatyper, arrayer, synlighet
- Preprocessing, kompilering, länkning
- IDE, .c- / .h-filer,

Läromoment för hemuppgifterna (=övingsuppgifterna):

- Även bitmanipulering (AND/OR/XOR)

Övningsuppgifter: v1. (se web)



The screenshot shows a web page with a blue header bar containing the text 'Maskinorienterad Programmering / Programming av Intervalkva System' and 'Hemuppgifter C-programmering'. Below the header, there is a main content area with text in Swedish. The text discusses the assignment, which involves implementing a C program to calculate intervals based on user input. It mentions that the assignment is intended for students who have completed the first part of the course and provides instructions for how to submit the assignment. At the bottom of the page, there are two download links: 'hemuppgifter.zip' and 'hemuppgifter.pdf'.

Kursmaterial

CHALMERS
Programmering av inbyggda system

Programmering av inbyggda system

Kursintroduktion

Ur innehållet:
Syften, mätsättningar, kurslitteratur och genomförande
Översikt av laborationer

Kursintroduktion
Kursintroduktion
Kursintroduktion

Assembler / ARM-föreläsningar

Kursbok



Arbetsboken



LaborationsPM
(online). 5 labbar.



Exempelsamling
(övningar online).

CHALMERS
Grundläggande C-programmering
- För maskinorienterad programmering
Ulf Assarsson

Läromoment:
• Datatyper, arrayer, synlighet
• Preprocessing, kompilering, länkning
• IDE, c- / .h-filer,
Övningsuppgifter: v1. (se web)

C-föreläsningar 5 st

Hemuppgifter C-programmering

För denna lista finns de grundläggande hemuppgifterna i C. Lösningshandledning finns till alla uppgifterna (se nedan). Det finns givetvis fler att förfölja sätt att lösa och samma uppgift. Dessa övningsuppgifter är till för att du ska få en bra grund i C och för att du ska kunna lösa andra grundläggande övningar du kan finna på nätet. Förhoppningsvis är de även lite roliga. De syftar till att du på ett förtunat sätt får er vana till tekniken där du förflyttar koden från en plats till en annan. Detta är en bra träningsmetod för att förstå hur programmet fungerar och hur det handlar om kommunikation mellan periferieenheter och koden uträkningsintygget.

Datora, dataprogrammerare är tyvärr mycket unika, men detta ändrar inte att det är väldigt intressant att medverka i projektet. Det är också att förtolka sig uppgifterna med mer diversifiterade, mönsterbaserade och alternativa problemställningar än vad dagens avancerade spel är. Detta säger som sagt minst sagt att det är bra att förflytta koden till en annan plats och att använda olika tekniker för att lösa den. Detta är också att förtolka sig uppgifterna som inte riktigt hör till en sak. Detta är också att förtolka sig uppgifterna som inte saknas i andra kurser och tas för givet att ni kan hantera dem. Men vi ska starta ihop.

Hur som helst finns här andra grundläggande övningar för den som skulle vilja ha:

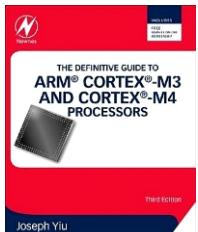
- <http://www.learn-c.org>
- <http://www.tutorialspoint.com/cprogramming/>
- <http://www.cprogramming.com/tutorial.html>

Samma websida

Exempelsamling
C – Övningsuppgifter med facilit.
(online)

Om ni vill:

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

Kursböcker?



- Arbetsboken är motsvarighet till kursbok, dvs huvudmaterialet.
 - Labbarna – känna på att köra mot riktiga hårdvaran istället för simulator.
 - Lektionerna – att kunna förstå arbetsbok (+ lab).
 - Exempelsamling:
 - Övningar i assembler/C
 - Övningar i C inför arb.bok + lab.
- C:

En del behöver troligen ingen kursbok i C men dessa finns om ni vill:

 - The C programming Language (ANSI-C)
 - https://hassanility.files.wordpress.com/2013/11/the_c_programming_language_2.pdf
 - Den var gratis nyss, men verkar just nu inte längre finnas i gratisversion (?). Googla!
 - Vägen till C, Skansholm, 2011. "Mer innehållsrik"
 - C från början, Skansholm, 2016. "Basic".
 - C reference card ANSI (på resurssidan)
 - <http://www.cse.chalmers.se/edu/resources/pinsys/ebook/c-refcard.pdf>

C – Övningsuppgifter

- Online – se länk på kurshemsidan för dagens C-föreläsningar.
(Första veckans uppgifter innehåller även länkar till alternativa nybörjar-C-övningar online.)

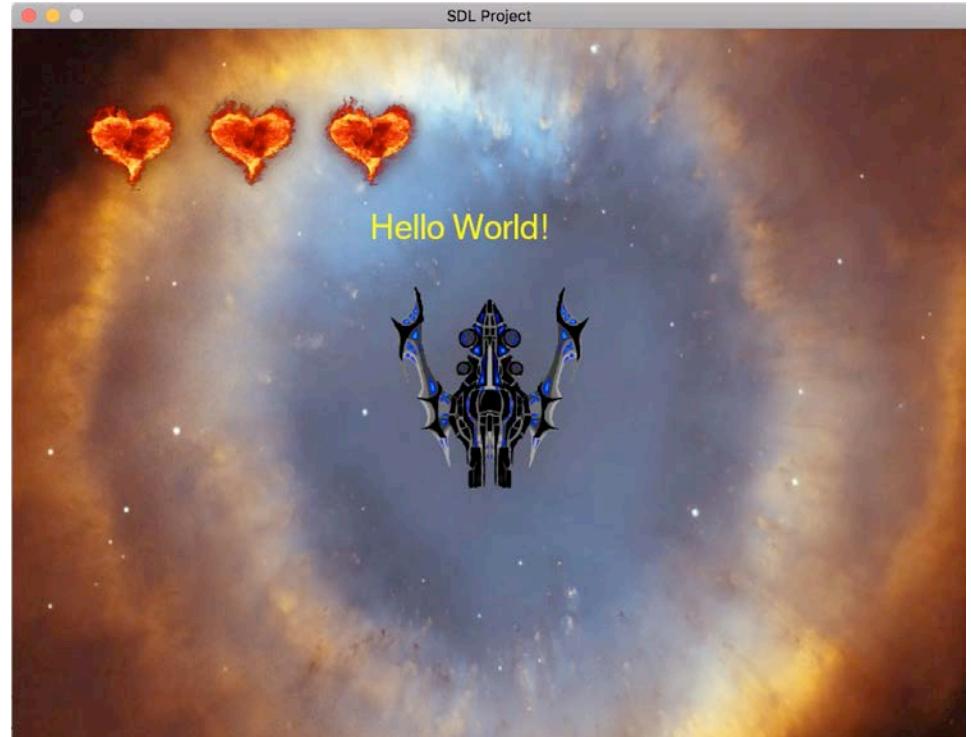
www.cse.chalmers.se/~ulfse/M01/hemuppgifter/hemuppgifter.htm
Maskinorienterad Programmering / Programmering av inbyggda System
Hemuppgifter C-programmering
Förel. 1 Förel. 2 Förel. 3 Förel. 4 Förel. 5

Övningsuppgifter kopplade till C-föreläsningarna
På denna sida finner du grundläggande övningsuppgifter i C. Lösningstillståndet finns till alla uppgifterna (ta reda). Det finns givna ofta ett föreläsnings sätt att lösa en och samma uppgift. Dessa övningsuppgifter är mer orienterade mot vad ni behöver just den här kursen jämfört med andra grundläggande övningar ni kan finna på nätet. Förberedningarna är de avsatta för att ni kan fånga upp det som är viktigt för er sista laboration där ni får implementera ett valfritt spel (eller annan dylik appli) på labdatoren MD407. Tanken där är framförallt att ni på egen hand skall kommunicera med periferienheter och kunna utnyttja interupter.

Doktiga spelprogrammator är ofta mycket attraktiva inom övriga industrier, medan omvänt är mindre spänslar. Det är svårt att föreställa sig applikationer med mer diversifierade, mångfacetterade och allomfattande problemområden än spel. Dagens avancerade spel är inte sällan som små minicomputersystem, med en restriktionskarta, parallellism, multitasking, high-performance computing, rihastning, näverk, säkerhet, ljud, grafik och artificiell intelligens. Det mest av detta omfattar färdigheter som ni tillskansar er i senare kurser. Ni har typiskt heller ännu inte läst om avancerade datatekniker och algoritmer.

Grundläggande i realtidspel är dock att de hanterar en mängd överlappande handelsflöden och har inputs och outputs. Vad domingarna på den här sidan syftar till (förutsett att er är grundskolekänd i C, att adressera minne och manipulera bitar) är att även lära er bemärka hyfat avancerade handelsflöden som inte obudandets följer en rak sekvensell ordning (slaget som ofta saknas i andra kurser och tas för givet att ni kan hantera). Men vi ska starta mycket.

Hur som helst finns här andra grundläggande övningar för dem som skulle vilja ha:
• <http://www.learn-c.org>
• <http://www.tutorialspoint.com/cprogramming/>
• <http://www.cprogramming.com/tutorial.html>





C – Bakgrund

- Short Code, 1949, 1:st high level language
- Autocode, early 50'ies.
- Fortran, IBM, ~57.
- Lisp, 58.
- Cobol 60 (Common Business-oriented language).
- BASIC, 1964.
- ALGOL 60 (**ALGOrithmic Language 1960**).
- Simula, 60:ies.
- C, ~1969.
- Prolog, 1972.
- Ada, ~1975
- Pascal, ~1975.
- ML, 1978.

C – Bakgrund

- Moderna språk:
 - C, C++, D, Java, javascript, Objective-C, C#, ...
- Maskinnära programmering:
 - Behöver språk med pekare till absoluta adresser
 - Basic, Ada 95 (och senare versioner), C, C++, C# (med nyckelordet *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 – Historik

- B, Bell Labs ~1969
- C: Utvecklades först 1969 – 1973 av Dennis Ritchie vid AT&T Bell Labs.
- Högnivå språk med kontakt mot maskinvara.
- Ett utav de mest använda språken.
- C++, D.
- Maskinnära, pekare

C respektive Assembler

- Varför C istället för assembler?
 - Färre rader kod, mindre risk för fel, snabbare...,
 - processorberoende
- Varför förstå hur C kompileras till assembler?
 - prestandaoptimering och resonera kring prestanda (tex för datorgrafik, GPU:er, HPC).
 - Hur mycket snabbare är en while-loop än rekursion?
 - Loop-unroll?
 - energikonsumtion
 - säkerhet/robusthet/risker
 - Kunna debugga
 - Kunna mixa C/asm vid drivrutinsprogrammering eller prestandakritiska förlöpp.

$a = b - c;$

Istället för:

LDR r3, =b

LDR r2, [r3]

LDR r3, =c

LDR r3, [r3]

SUBS r2, r2, r3

LDR r3, =a

STR r2, [r3]

Översikt C – fem lektioner

- Lekt 1 - Syntax och Programstruktur:
 - IDE, variabler (dekl. + tilldeln.), typkonverteringar, ASCII, funktioner, variabelsynlighet, programstruktur, kompilering/länkning, arrayer
- Lekt 2 – Pekare och Arrayer:
 - Pekare, absolutadressering (portar), typedef, volatile, #define, arrayer av pekare, arrayer av arrayer
- Lekt 3 – Structs och Funktionspekare:
 - Structs, pekare till structs (pilnotation), array av structs, portadressering med structs, funktionspekare, structs med funktionspekare (objektorienterad stil)
- Lekt 4 - Mer programstruktur samt Dynamisk minnesallokering:
 - Synlighet - static, #extern, (inline), #if/#ifdef, #include guards, enum, union, little/big endian, dynamisk minnesallokering (malloc/free)
- Lekt 5 – realtidsstyrsystem (spelprogrammering) och Avancerad C:
 - Realtidsloop, C99, obskyra C-konstruktioner,
 - Om vi hinner: Dubbelpekare, code injection (via stack frames).

C – Standarder

- 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 hw addressing)

Hello world! – program

```
#include <stdio.h>

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

C vs Java. Några skillnader:

- C: saknar klasser.
Har dock structs för sammansatta datatyper.
- Booleans är ej egen typ. **true/false** finns ej. 0 är false. Ett värde != 0 är true. (Därför definierar man ofta TRUE/FALSE som 1 resp 0. 1 && 1 = “Implementationsspecifikt för kompilatorn”).

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

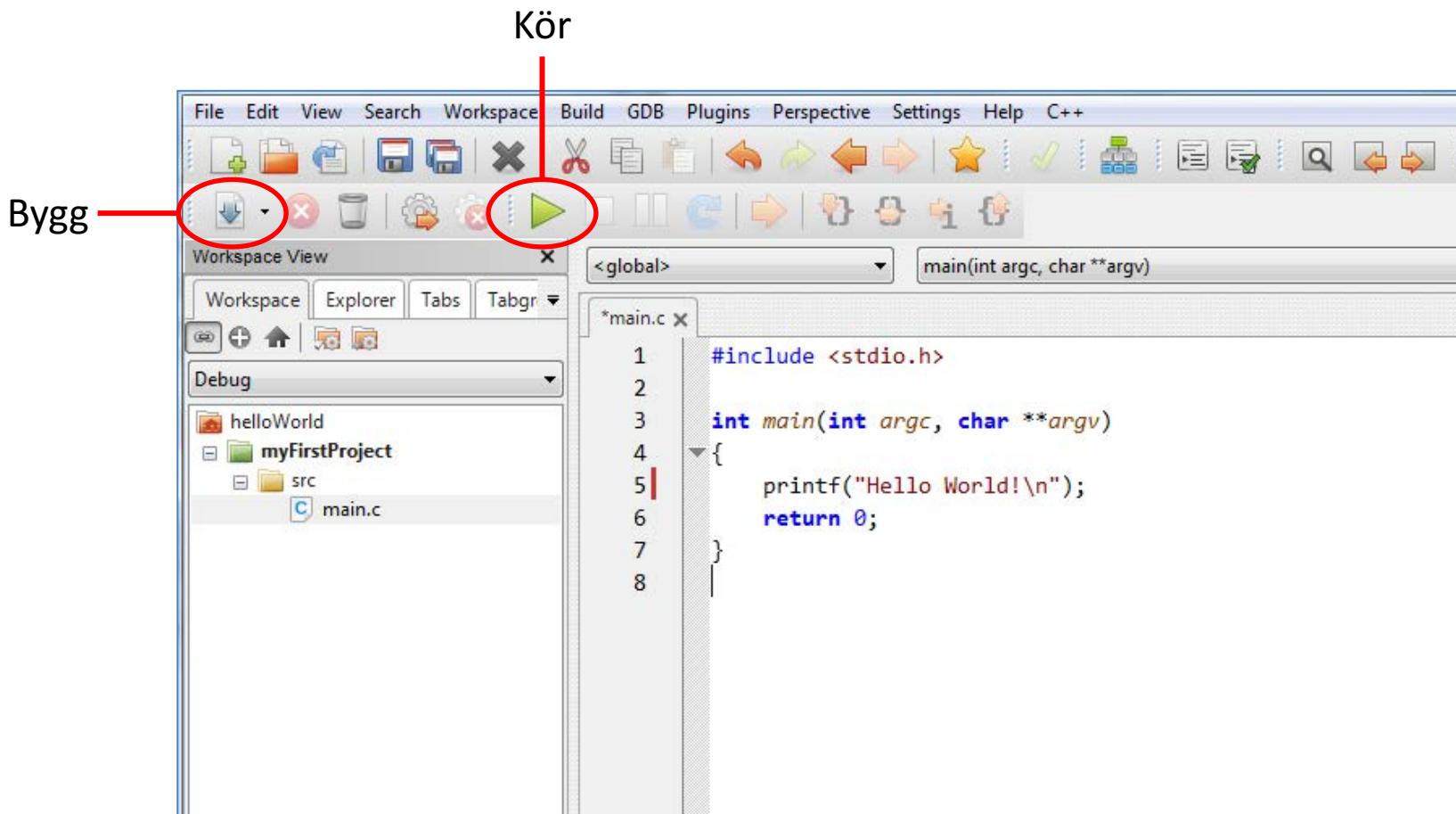
Mer C:

- Type conversion
 - default: convert narrower format to the wider format
 - Synlighet:
 - global synlighet,
 - synlighet i funktion
 - resp scope.

```
float a;  
a = 1.0 / 3;  
// a == 0.33333343
```

```
#include <stdio.h>  
  
int x;  
int foo(int y)  
{  
    if( x == y ){  
        int z = 4;  
        z = z + x + y;  
        return z;  
    }  
    return x;  
}
```

Integrerad utvecklings miljö (IDE)



Vi använder CodeLite som är gratis och open-source.
Hämta via resurshemsidan!! (OBS - ej via <http://codelite.org/>)

Från terminalen

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

Variabler

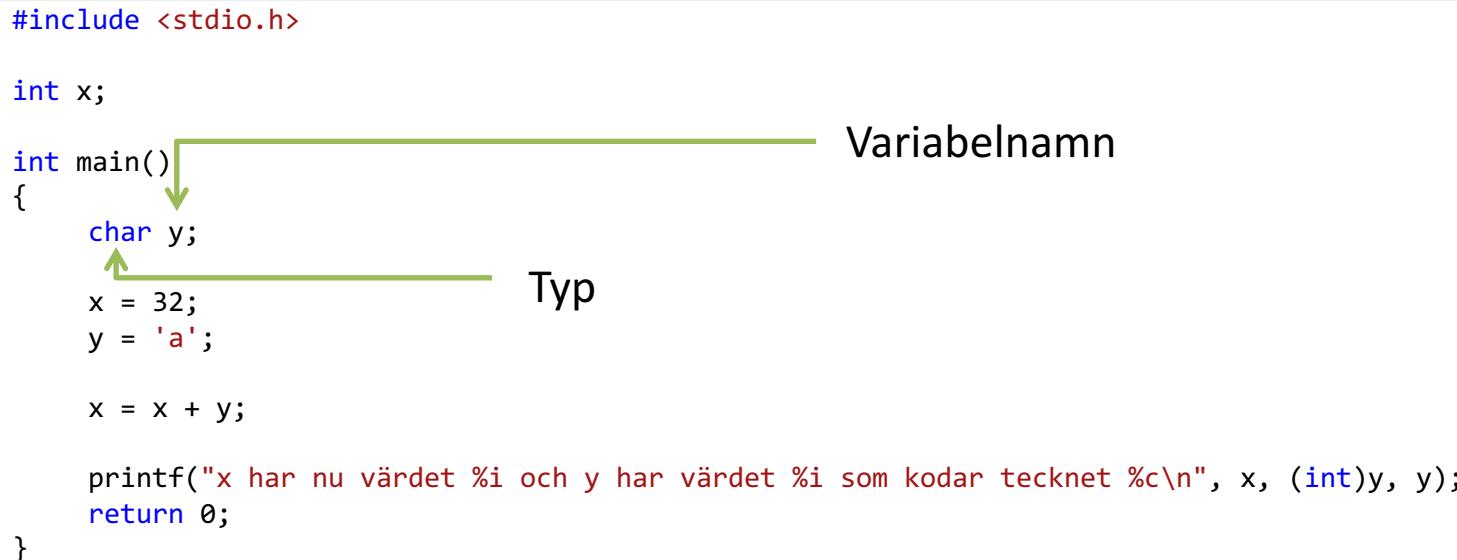
```
#include <stdio.h>

int x;

int main()
{
    char y;
    x = 32;
    y = 'a';

    x = x + y;

    printf("x har nu värdet %i och y har värdet %i som kodar tecknet %c\n", x, (int)y, y);
    return 0;
}
```



The diagram illustrates the C code with annotations:

- A green arrow points from the label "Variabelnamn" to the declaration `char y;`.
- A green arrow points from the label "Typ" to the declaration `char y;`.
- A green arrow points from the label "Variabelnamn" to the declaration `int x;`.
- A green arrow points from the label "Typ" to the declaration `int x;`.

Utskrift:

x har nu värdet 129 och y har värdet 97 som kodar tecknet a

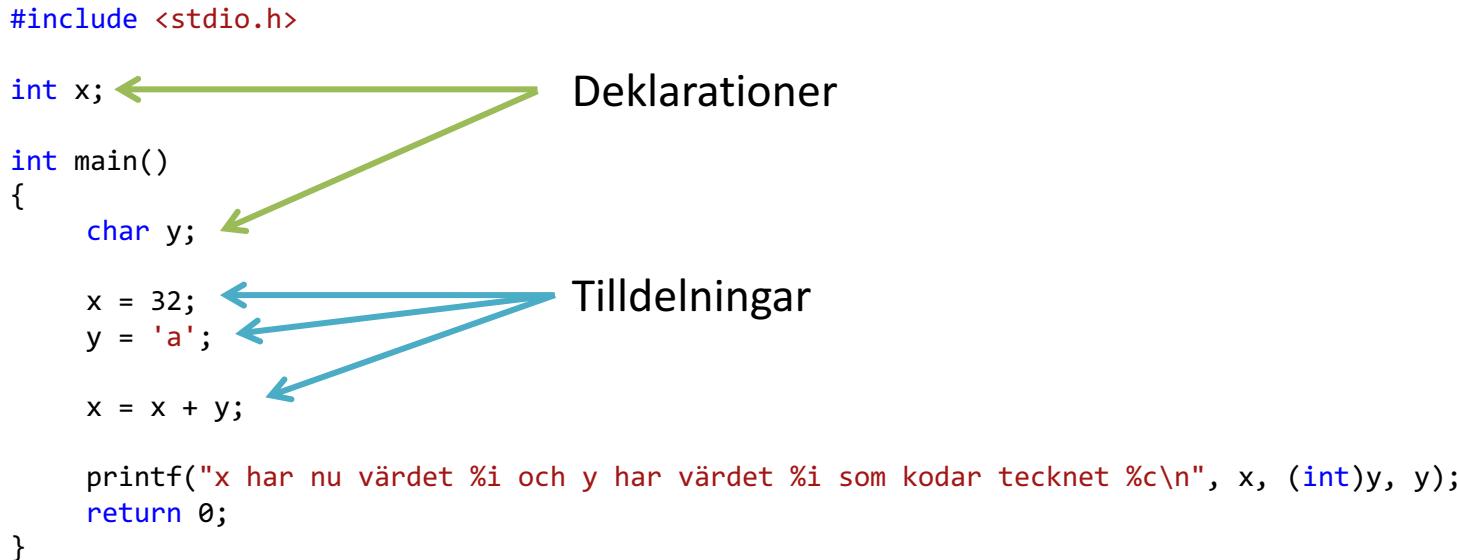
För `printf()`-parametrar, se t ex [C Reference Guide](#)

Deklarationer och tilldelningar

```
#include <stdio.h>

int x;
int main()
{
    char y;
    x = 32;
    y = 'a';
    x = x + y;

    printf("x har nu värdet %i och y har värdet %i som kodar tecknet %c\n", x, (int)y, y);
    return 0;
}
```



Deklarationer

Tilldelningar

En deklarerad variabel som ännu inte tilldelats ett värde är oinitierad

C89 – deklarationer först

```
#include <stdio.h>

int x;

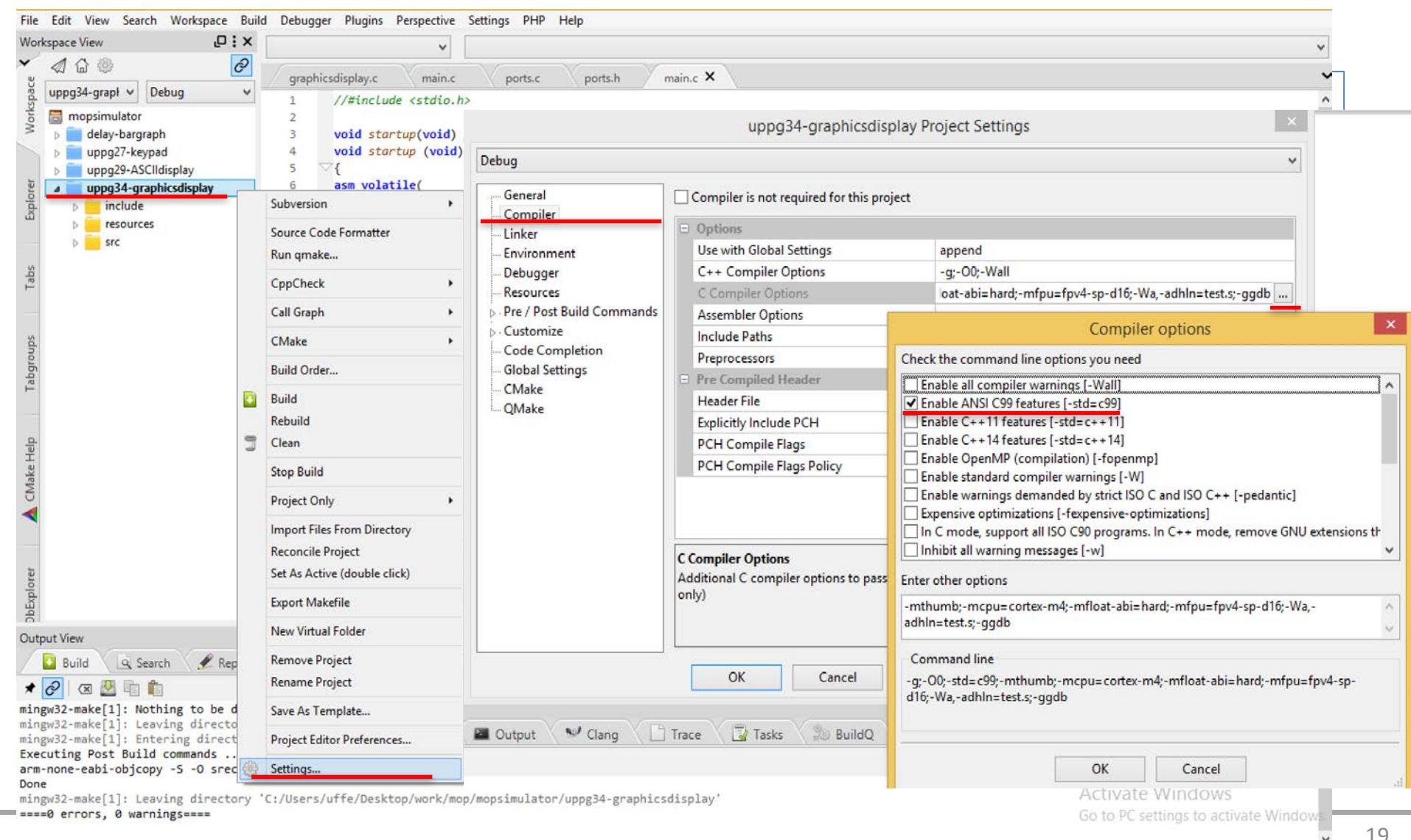
int main()
{
    x = 32;           ← I C89 måste alla deklarationer
    char y = 'a';    ← komma allra först i en funktion.

    x = x + y;

    printf("x har nu värdet %i och y har värdet %i som kodar tecknet %c\n", x, (int)y, y);
    return 0;
}
```

Fungerar ibland ändå (t ex i gcc), men för vår ARM-kompilator måste C99 explicit anges som kompilatorflagga.

Enable C99



The screenshot shows the Eclipse CDT IDE interface with the following details:

- File Menu:** File, Edit, View, Search, Workspace, Build, Debugger, Plugins, Perspective, Settings, PHP, Help.
- Toolbar:** Workspace View, Debug.
- Left Sidebar (Explorer):** Shows the project tree with the current active project: **uppg34-graphicsdisplay**. Other projects listed include mopsimulator, delay-bargraph, uppg27-keypad, uppg29-ASCIIdisplay, and uppg34-graphicsdisplay.
- Code Editor:** Displays source code files: graphicsdisplay.c, main.c, ports.c, ports.h, and main.c.
- Project Settings Dialog:** Opened for the **uppg34-graphicsdisplay** project. The **Debug** tab is selected. Under the **Compiler** section, the option **Compiler is not required for this project** is unchecked. In the **Options** section, the **C++ Compiler Options** field contains `-g;-O0;-Wall`, and the **C Compiler Options** field contains `oat-abi=hard;-mfpu=fpv4-sp-d16;-Wa,-adhln=test.s;-ggdb`.
- Compiler Options Dialog:** A modal dialog titled "Compiler options" is open, overlaid on the project settings. It lists various compiler options:
 - Check the command line options you need
 - Enable all compiler warnings [-Wall]
 - Enable ANSI C99 features [-std=c99] **(This option is highlighted with a red box)**
 - 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`
- Output View:** Shows build logs for mingw32-make, indicating no errors or warnings.

Typkonverteringar

```
#include <stdio.h>

int x;

int main()
{
    char y;

    x = 32;
    y = 'a';

    x = x + y;           Implicit typkonvertering     Explicit typkonvertering

    printf("x har nu värdet %i och y har värdet %i som kodar tecknet %c\n", x, (int)y, y);
    return 0;
}
```

Typkonvertering kallas också cast, och man säger att man castar.



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	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Array (Fält)

```
#include <stdio.h>

char namn1[] = {'E', 'm', 'i', 'l', '\0'};
char namn2[] = "Emilia";
char namn3[10];

int main()
{
    printf("namn1: %s \n", namn1);
    printf("namn2: %s \n", namn2);
    printf("sizeof (namn2): %i \n", sizeof(namn2));

    return 0;
}
```

Utskrift:

```
namn1: Emil
namn2: Emilia
sizeof(namn2): 7
```

```
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;
}
```

Funktioner

```
#include <stdlib.h>
int foo(int x, char y)
{
    int sum = 0;

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

    return sum; ← Returvärde av returtyp
}
```

Argumenten är ”pass-by value”.

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

↑ var2 har fortfarande värdet 7
eftersom funktionen inte ändrat dess värde

Synlighet/Visibility/Scope

- Global synlighet (global scope)
- Filsynlighet (file scope)
- Lokal synlighet (e.g. function scope)

Synlighet

```
#include <stdlib.h>

char x;

int foo()
{
    // x är synlig
    // y är inte synlig
}

char y;
```

Synlighet på funktionsnivå

```
#include <stdlib.h>

char x;

int foo(float x)
{
    // argumentet x (float) är synligt
}
```

Synlighet på funktionsnivå

```
#include <stdlib.h>

char x;

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

Vilken synlighet har högst prioritet?

```
#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 %i", x);    Vad är x?
    return 0;
}
```

Funktionsprototyper

```
#include <stdio.h>

// funktionsprototyp
int foo(int x);

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

int foo(int x)
{
    // funktionskropp
}
```

Programstruktur

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

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

c-fil

Inkluderar header-fil

```
// foo.h
int foo(int x);
```

header-fil

Innehåller

funktionsprototyper

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

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

    return x;
}
```

c-fil

header-filen måste inte ha samma namn som c-filen, men det är enklare så.

Om man har längre programkod lägger man inte allt i main.c utan delar upp funktionaliteten mellan .c-filer. .h-filerna innehåller deklarationerna och inkluderas av andra .c-filer.



Workspace View

uppg27-keyp: Debug

Explorer

Tabs

Tabgroups

CMake Help

DbExplorer

main.c x 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 */
29         " bl main\n"                /* call main */
30         " bl Exit\n"                /* call exit */
31         " . . . "
32     );
33 }
```

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



Workspace View

uppg27-keyp: Debug

mopsimulator

- delay-bargraph
- uppg27-keypad
 - resources
 - md407-ram.x
 - src
 - main.c
 - types.h
- uppg29-ASCIIdisplay
- uppg34-graphicsdisplay

Explorer

Tabs

Tabgroups

CMake Help

DbExplorer

main.c types.h

```
8  #ifndef TYPES_H
9  #define TYPES_H
10
11  typedef unsigned char          byte;
12  typedef unsigned short         word;
13  typedef unsigned int           dword;
14  typedef int                   bool;
15  typedef signed char           int8_t;
16  typedef unsigned char          uint8_t;
17  typedef signed short int      int16_t;
18  typedef unsigned short int    uint16_t;
19  typedef signed int            int32_t;
20  typedef unsigned int           uint32_t;
21  typedef unsigned long long   uint64_t;
22  typedef long long             int64_t;
23  typedef int8_t                int8;
24  typedef uint8_t               uint8;
25  typedef int16_t               int16;
26  typedef int16_t               uint16;
27  typedef int32_t               int32;
28  typedef uint32_t              uint32;
29  typedef int64_t               int64;
30  typedef uint64_t              uint64;
31  typedef unsigned char          uchar_t;
32  typedef uint32_t              wchar_t;
33  typedef uint32_t              size_t;
34  typedef uint32_t              addr_t;
35  typedef int32_t               pid_t;
36
37  #endif
```

Definieras ofta i "stdint.h"
Men för arm-kompilatorn
har vi inga färdiga .h-filer.
Kan finnas online...

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

Från källkod till exekverbar

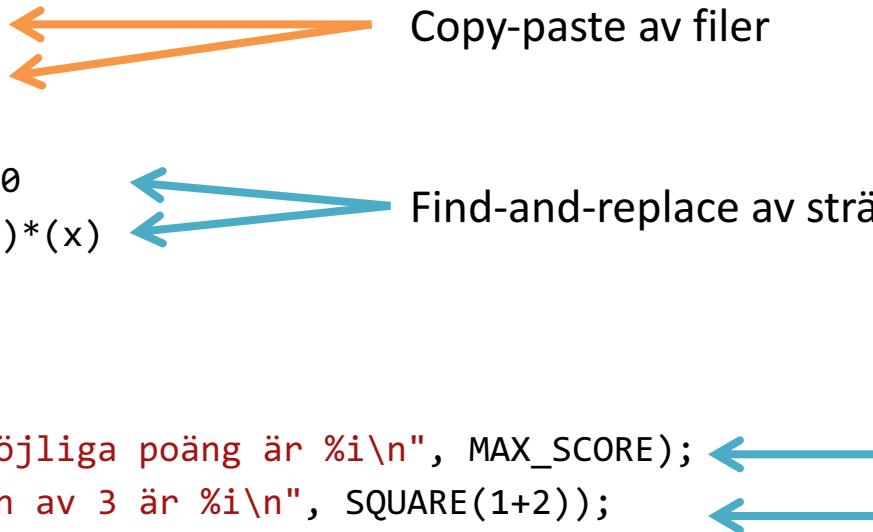
1. Preprocessing
2. Kompilering
3. Länkning

Preprocessorn

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

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

int main()
{
    printf("Högsta möjliga poäng är %i\n", MAX_SCORE);
    printf("Kvadraten av 3 är %i\n", SQUARE(1+2));
    printf("x is %i", foo(0));
    return 0;
}
```



Copy-paste av filer

Find-and-replace av strängar

Preprocessorn arbetar på källkoden på ”textnivå”.

Kompilering

- Processar en .c-fil i taget:
 - Skapar en objektfil (dvs .o-fil), per .c-fil, som innehåller:
 - Maskinkod för instruktioner
 - Symboler för addresser
 - För funktioner/variabler i objektfilen.
 - För funktioner/variabler i andra objektfiler/bibliotek.



File Edit View Search Workspace Build Debugger Plugins Perspective Settings PHP Help

Workspace View

uppg27-keypad Debug

Workspace Explorer Tabs Tabgroups CMake Help DbExplorer Output View

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     /* ... */
12 }
13
14 static void main( void )
15 {
16     /* ... */
17 }
18
19 #endif
20
21 void main( void )
22 {
23     /* ... */
24 }
25
26 }
```

Open in CodeLite Open with Default Application Open Shell Open Containing Folder Compile Preprocess Exclude from Build Rename... Remove Git Svn

File Home Share View Copy Paste Cut Copy path Move to Copy to Delete Rename New folder New item Easy access Properties Open Select all Select none Invert selection Select

work work mop mopsimulator uppg27-keypad Debug

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 (Disk free space: 8,78 GB) 48,1 KB Computer

```
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



File Edit View Search Workspace Build Debugger Plugins Perspective Settings PHP Help

Workspace View

Explorer

Tabs

Tabgroups

CMake Help

order

Build

main.c X 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 */
29         " . . . "
30 );
31 }
```

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++

Länkning

- Sätter samman (flera) objektfiler till en exekverbar fil (.exe / .s19 el dyl).
- Översätter symbolerna till (relativa) adresser.

```
// main.c  
#include <stdio.h>  
#include "foo.h"  
  
int main()  
{  
    printf("x is %i", foo(0));  
    return 0;  
}
```

.c-fil

Kompileras till

.o-fil

```
// foo.c  
#include <stdlib.h>  
  
int foo(int x)  
{  
    if( x == 0 ){  
        int x = 4;  
        return x;  
    }  
  
    return x;  
}
```

.c-fil

Kompileras till

.o-fil

Länkas till:

.exe/.s19-fil



File Edit View Search Workspace Build Debugger Plugins Perspective Settings PHP Help

Workspace View

Explorer

Tabs

Tabgroups

CMake Help

Output View

Build Search Replace References Output Clean

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

Build

main.c x types.h

main.c // #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)
{
 /*
 * Initialize keypad pins
 */
}

void main(void)
{
 /*
 * Call initialization function
 */
 /*
 * Main program loop
 */
}

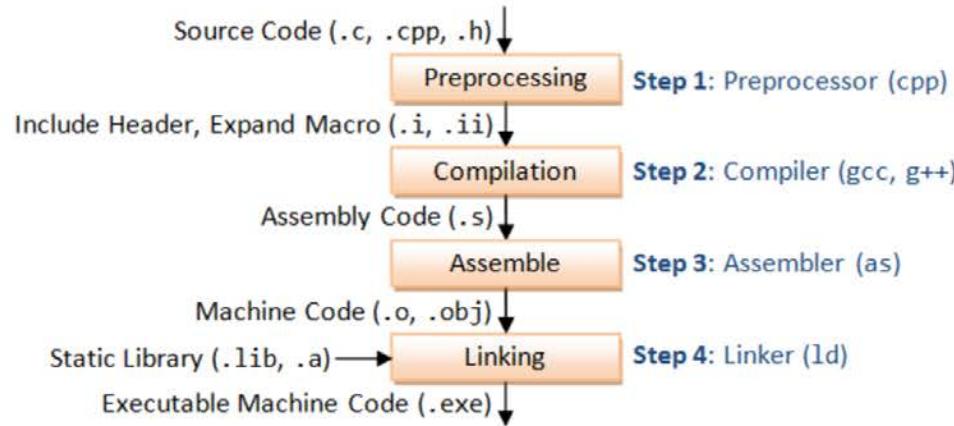
Clipboard Organize New Open Properties History Select all Select none Invert selection Select

work mop mopsimulator uppg27-keypad Debug

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 (Disk free space: 8,78 GB) 48,1 KB Computer

Ln 14, Col 0, Pos 196 SPACES C++



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
```

Aritmetiska operatorer

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

Jämförelseoperatorer

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

Logiska operatorer

Logical negation (NOT)	<code>!a</code>
Logical AND	<code>a && b</code>
Logical OR	<code>a b</code>

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

Bit operationer

Bitwise NOT	$\sim a$
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

Sammansatta tilldelsningsoperatorer

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 satser

```
int x = -4;

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

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

Utvärderar till falskt, kör ej.

Utvärderas till sant, kör, ty ($x \neq 0$)

- Noll betraktas som falskt.
- Allt som är skilt från noll betraktas som sant.

Loopar

```
int x = 5;  
  
while( x!=0 )  
    x--;
```

```
int x = 5;  
  
while( x )  
    x--;
```

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

Tre ekvivalenta loopar. Om inga
måsvingar används så är loop-kroppen
ett enda uttryck.

```
for(int i=0; i<5; i++ ) {  
    if(...)  
        continue; // hoppar till nästa iteration  
    if(...)  
        break; // avbryter loopen.  
    ...  
}
```

break och **continue**



Öva på bitoperationer

Bitwise OR används typiskt för att ett-ställa bitar.

Bitwise AND används typiskt för att nollställa bitar.

Bitwise XOR används typiskt för att invertera vissa bitar

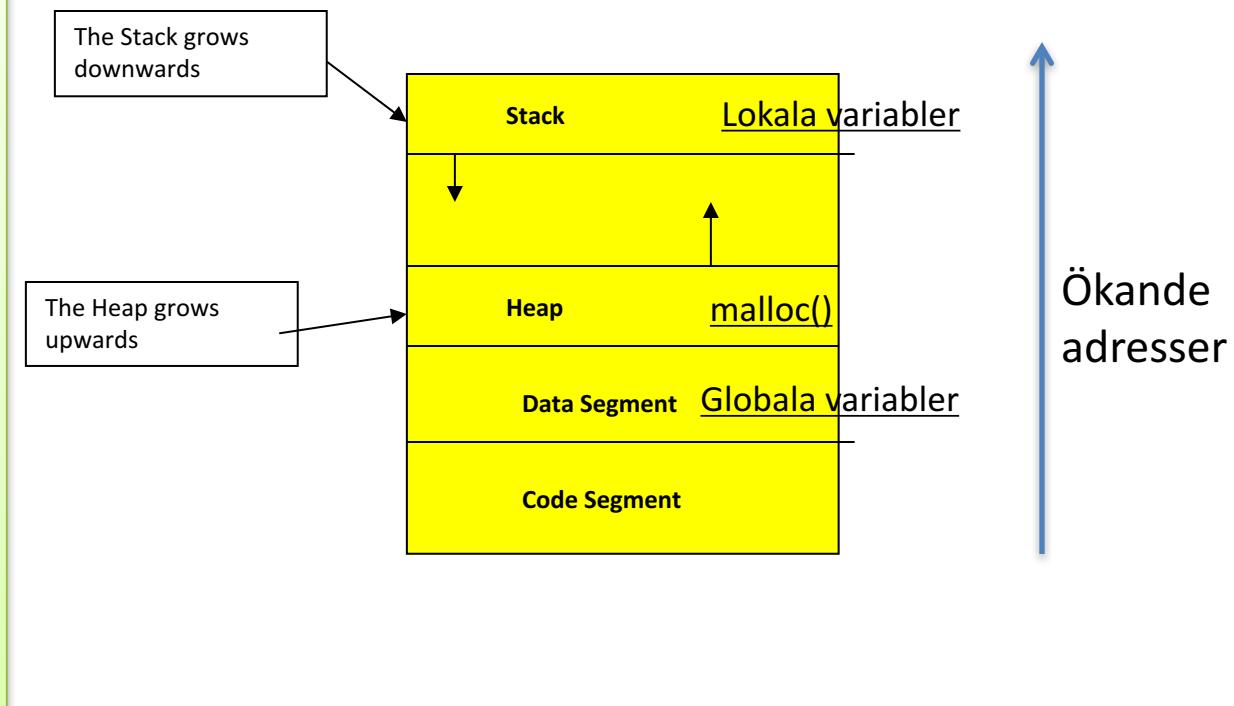
Bitwise NOT används för att invertera alla bitar

Ett Programs Adressrymd

- Alla program som körs, har något associerat minne, vilket typiskt är indelat i:
 - Code Segment
 - Data Segment (Holds Global Data)
 - Stack (where a function's local variables and other temporary data is stored)
 - Heap

```
#include <stdio.h>

int x; // Global variabel
int foo(int y) // som lokal variabel
{
    if( x == y ){
        int z = 4; // Lokal variabel
        z = z + x + y;
        return z;
    }
    int *a = malloc(1000); // heap
    ...
    free(a);
    return x;
}
```





Nästa föreläsning

Pekare, Arrayer, Portadressering

Pekare

- Har ett värde och en typ
 - Värdet är en minnesadress.
 - Typen talar om vad som finns där.

```
int tal[] = {1, 5, 3, 100, 7, 6, 15};  
int* störstTal = &tal[3];  
*störstTal = 200;
```

- Kunna referera till ett objekt eller variabel,
dvs utan att behöva skapa en kopia

Operatorer för pekare

Operator name	Syntax
Array subscript	$a[b]$
Indirection ("object pointed to by a ")	$*a$
Reference ("address of a ")	$\&a$
Structure dereference ("member b of object pointed to by a ")	$a->b$
Structure reference ("member b of object a ")	$a.b$

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

Sista operatorn är för att referera medlemmar av en struktur (**struct**), så ej en pekaroperator.