Laboration 1, introduktion till Visual Studio och c-programmering

Innehåll

Del 1, Starta Visual Studio 2
Del 2: Skapa ett program 5
Del 3, Felsökning 11
Del 4, Fönsterhantering i VS 13
Del 5, Programmeringsuppgifter 14
Uppgift 1 14
Uppgift 2 14
Uppgift 3 15
Uppgift 4 15
Uppgift 5 15
Uppgift 6 16
Uppgift 7 16

Laboration 1

Syfte:

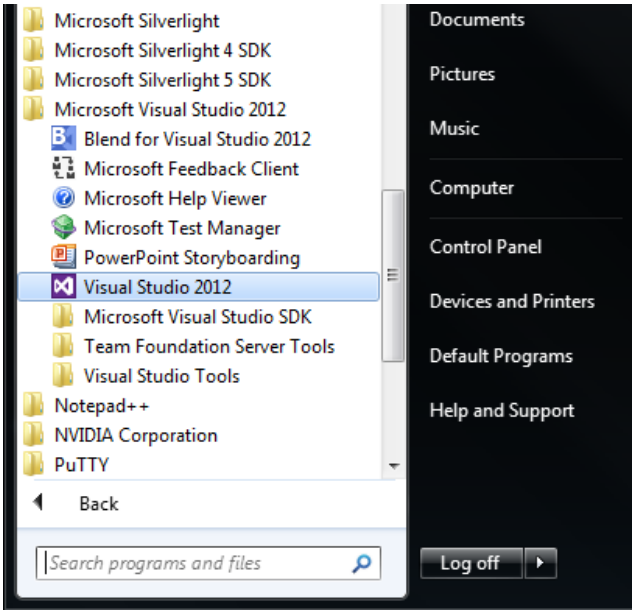
- Lära dig att skapa, kompilera, länka och exekvera program med hjälp av verktyget Visual Studio.
- Träna på att skriva och strukturera C-kod.

Redovisning:

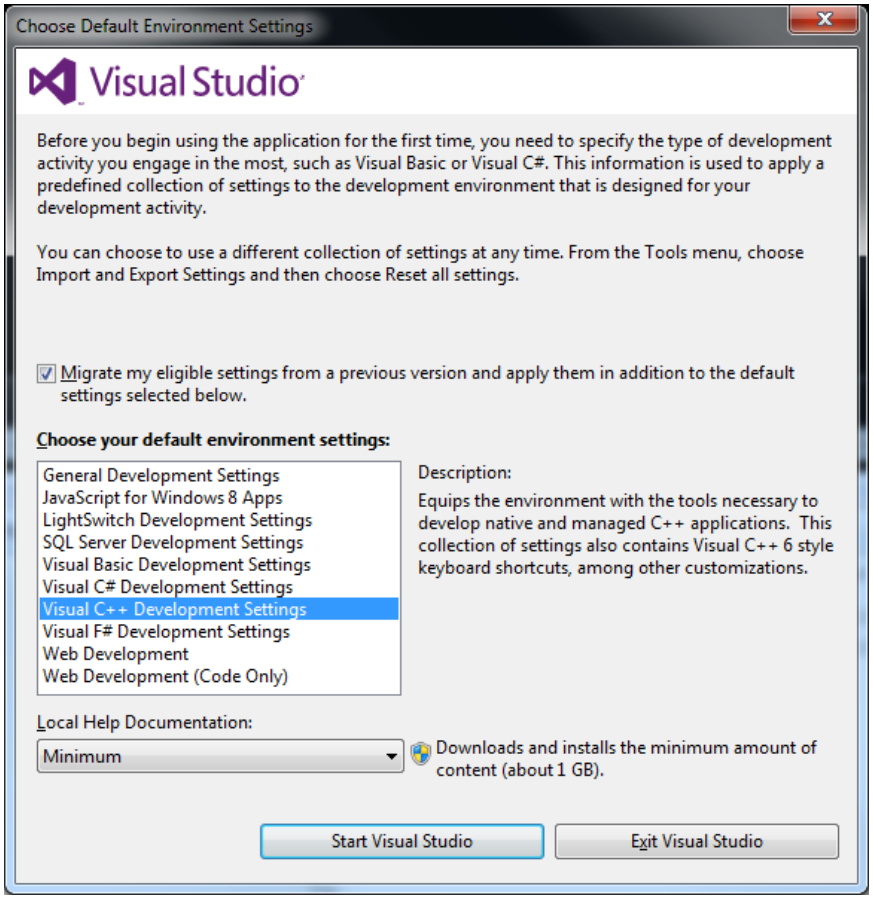
- Skriv en sammanfattning av ditt arbete. Lämna in på IT's i pdf-format. Du kan lämna in individuellt eller tillsammans med (max) en kamrat.
  Sammanfattningen ska innehålla punkterna:
  - ❖ Förutsättningar (hur du trodde att det skulle vara)
  - ❖ Genomförande (vad var lätt, vad var svårt)
  - ❖ Resultat (vad du lärde dig).
  - ❖ **Tips:**
    - ○ Dokumentera ditt arbete under tiden du tar dig igenom anvisningarna.

Del 1, Starta Visual Studio

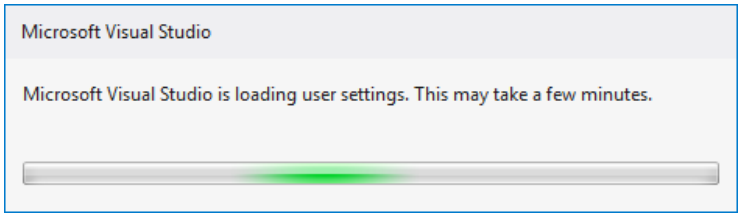
Leta reda på programmet Visual Studio bland de installerade programmen.



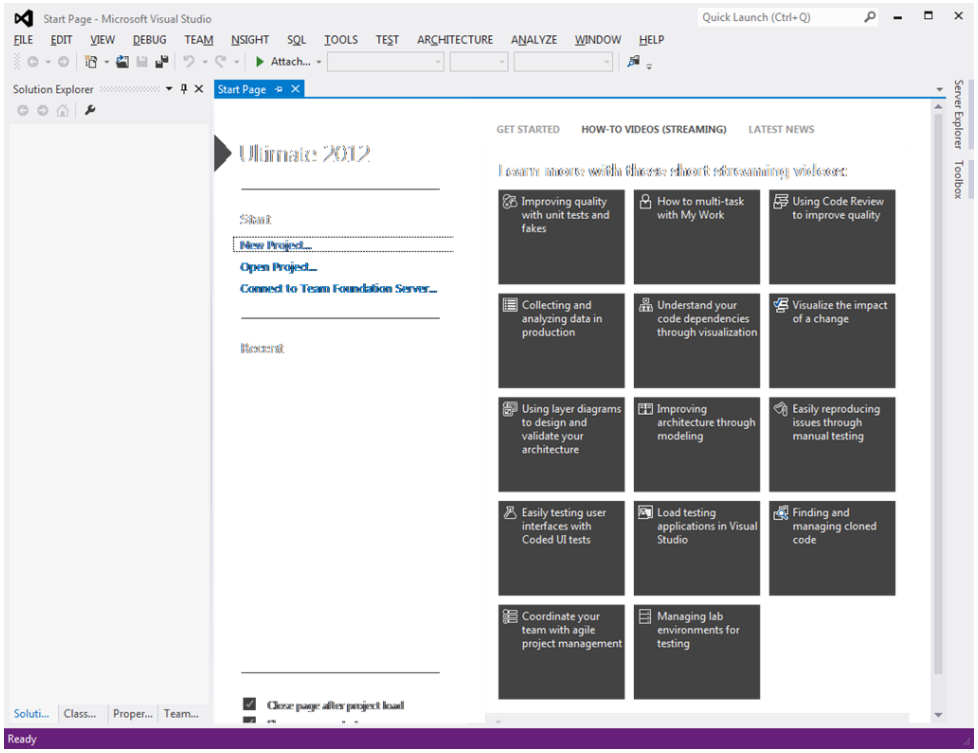
När du startar programmet första gången, får du frågor om diverse inställningar. I nedanstående formulär väljer du Visual C++ som standardmiljö (environment settings). Visual Studio (i fortsättningen bara VS) har ingen speciell miljö för C, vi använder C++.



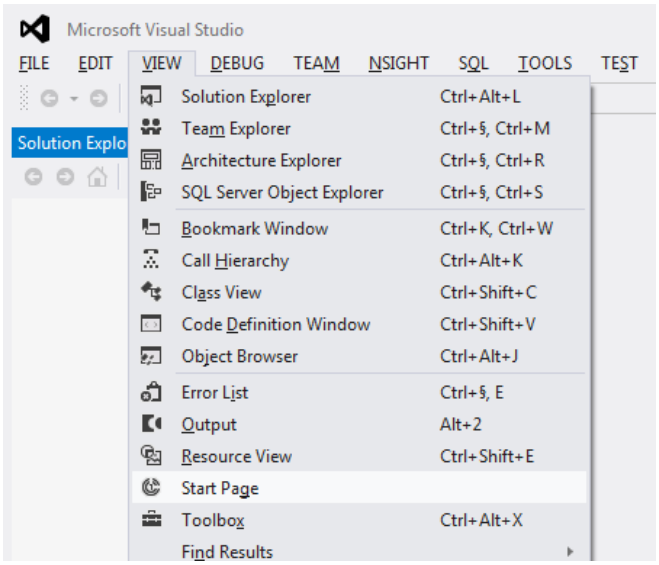
Klicka "Start Visual Studio". Det tar sedan flera minuter innan inställningarna är klara.



När inställningarna är klara, kommer VS:s startbild. "Start Page" tröttnar man snart på och kan stängas.



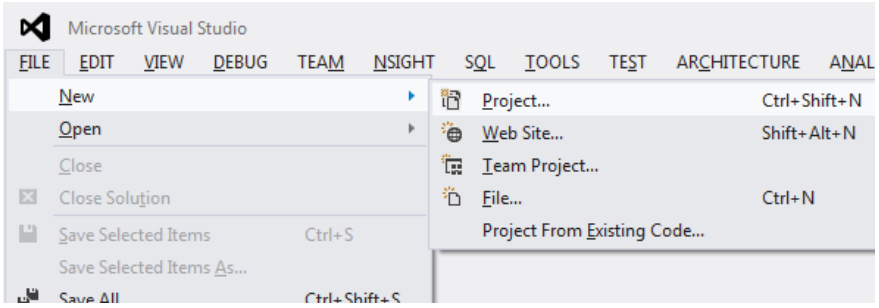
Vill man ha Start Page igen, kan man få upp den via menyvalet View - Start Page.



Del 2: Skapa ett program

När man skapar ett program i VS kallas det ett projekt (Project).

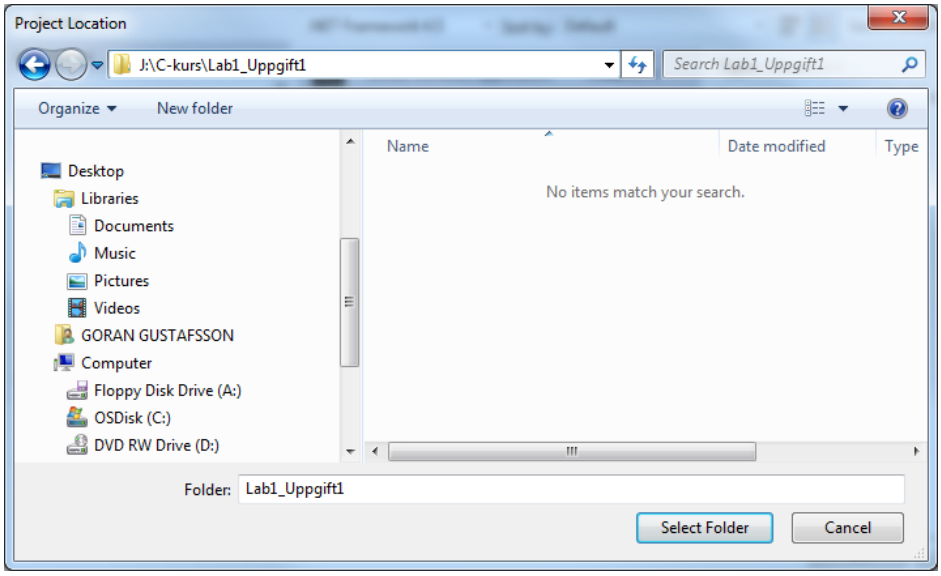
Välj File - New - Project:



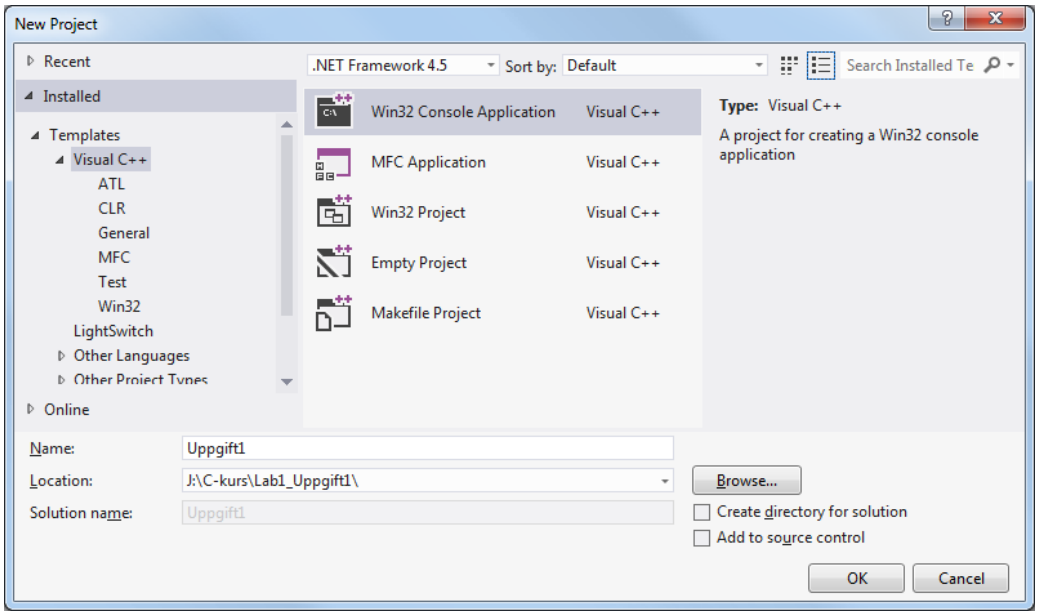
Du får upp ett formulär för att välja var ditt projekt ska sparas på disken.

OBS: du kommer så småningom att ha hundratals program att hålla reda på. Skapa en genomtänkt struktur med mappar så att du kan hitta dina program igen. Skapa förslagsvis en huvudmapp med kursens namn, sedan en undermapp "Labbar" och under den olika mappar för olika labbar och uppgifter.

En mapp skapar du med "New folder". Håll reda på var ditt projekt ligger.



När du skapat och valt en mapp är det dags att välja typ av program. Välj Win32 Console Application och namnge programmet (=projektnamn).



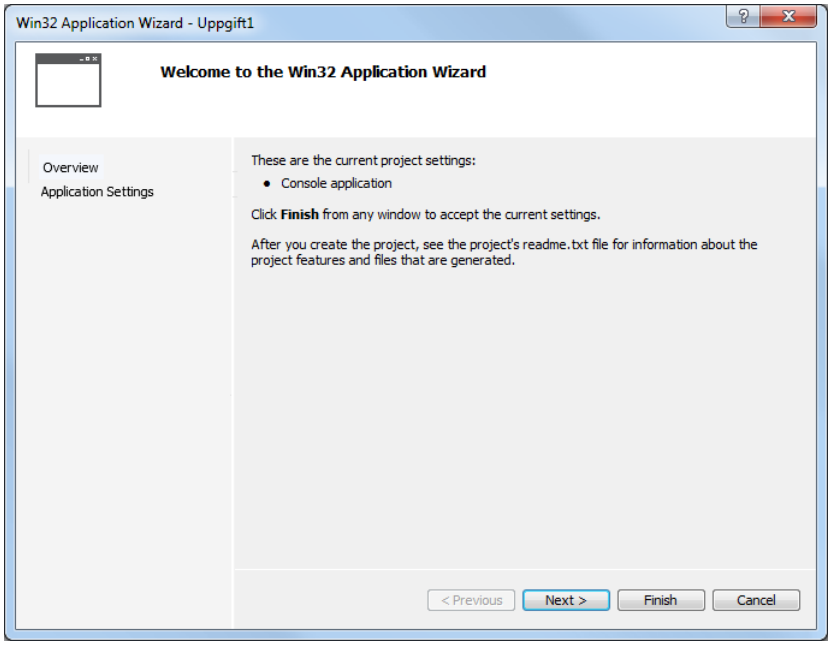
Ge ditt program ett namn som beskriver det.

Kontrollera att "Location" stämmer med vad du tänkt dig, om det inte gör det så tryck Cancel och börja om!

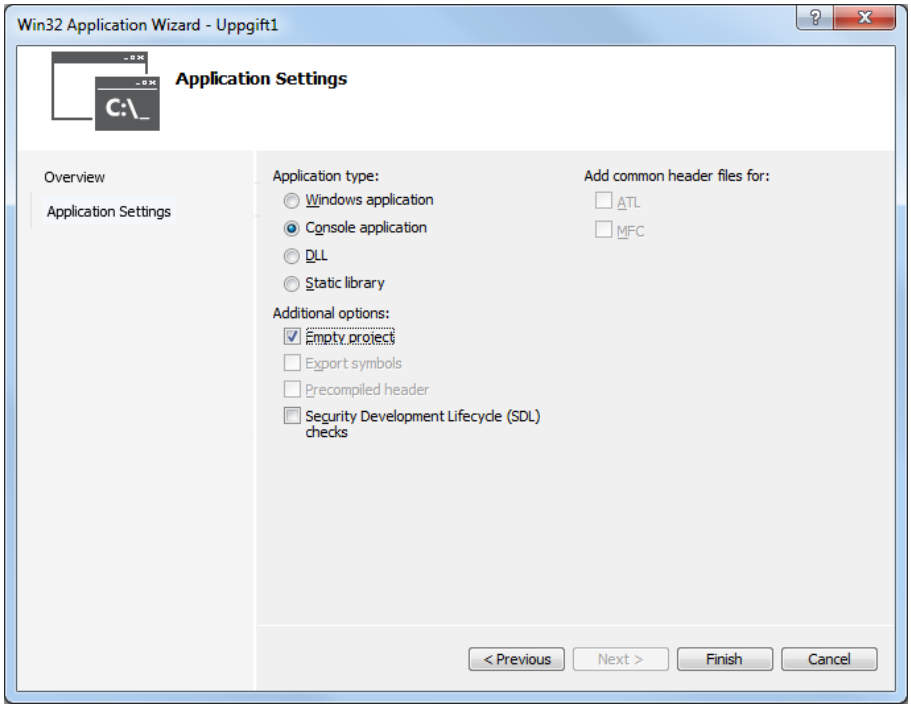
Klicka OK.

Nu ska du beskriva vilken typ av program det ska bli.

Kontrollera att det står Console application, Tryck Next (OBS inte Finnish).



Nästa steg är mera detaljerade val:



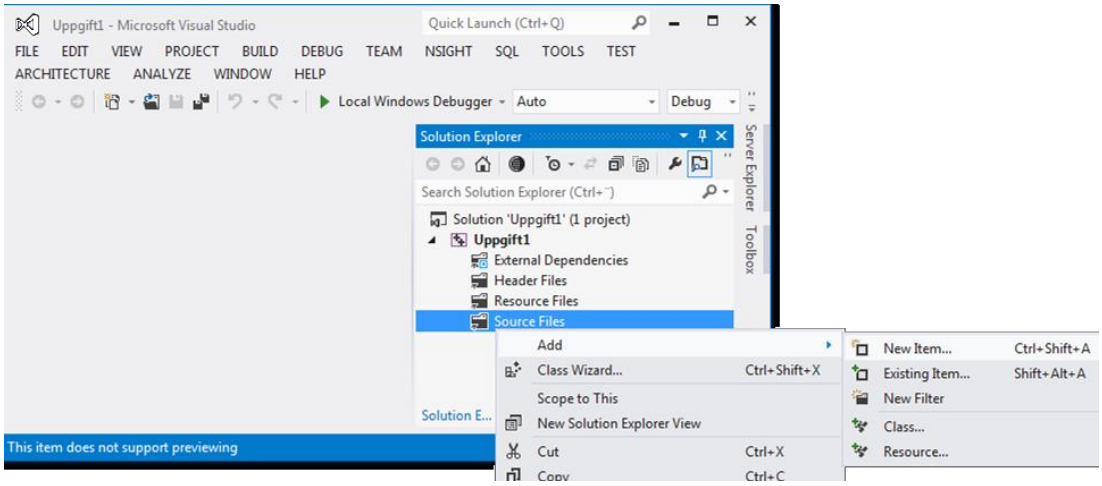
Välj Console application (ska redan vara markerat). I den nedre delen ska det bara vara Empty project som markeras. När du gjort detta klickar du Finnish.

Om allt gått väl, finns det nu ett projekt där du kan lägga till filer.

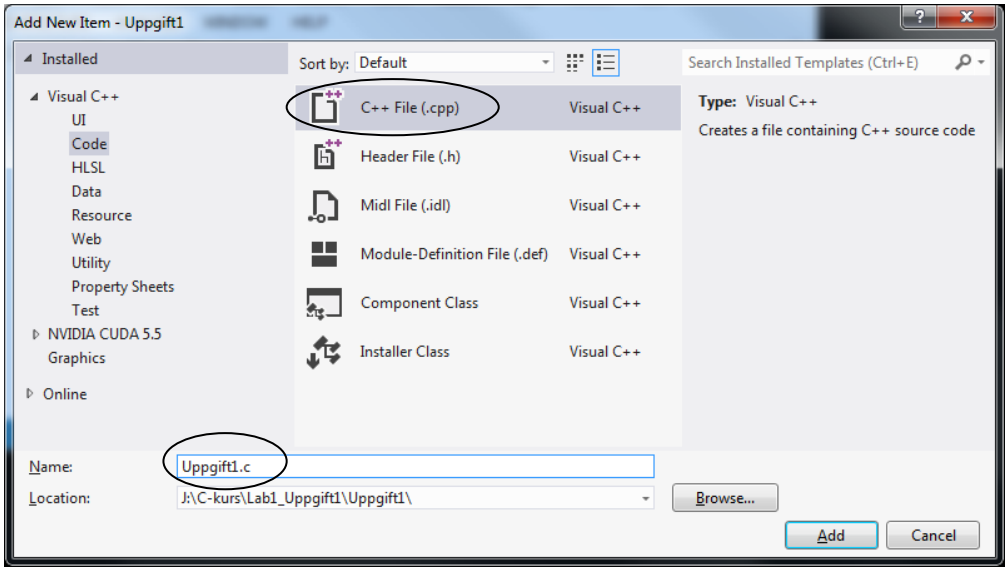
Börja med att lägga till en fil där källkoden kan skrivas.

Många moment i VS kan utföras på olika sätt, det enklaste sättet att lägga till en källkodsfil i VS är nog att högerklicka på Source Files i fönstret Solution Explorer, sedan välja Add - New Item. Om fönstret Solution Explorer inte visas, så välj från menyn View - Solution Explorer.

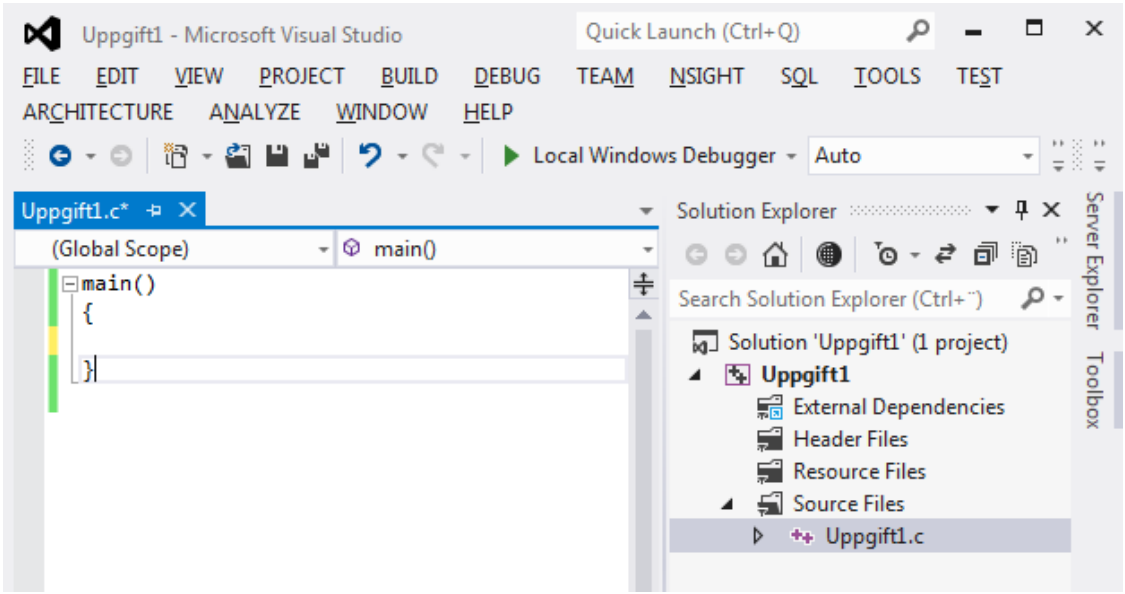
Om det redan finns några filer under Header Files, Resource Files eller Source Files, betyder det att du gjort något fel bland de tidigare valen. Det enklaste är då att börja från början och skapa ett nytt projekt.



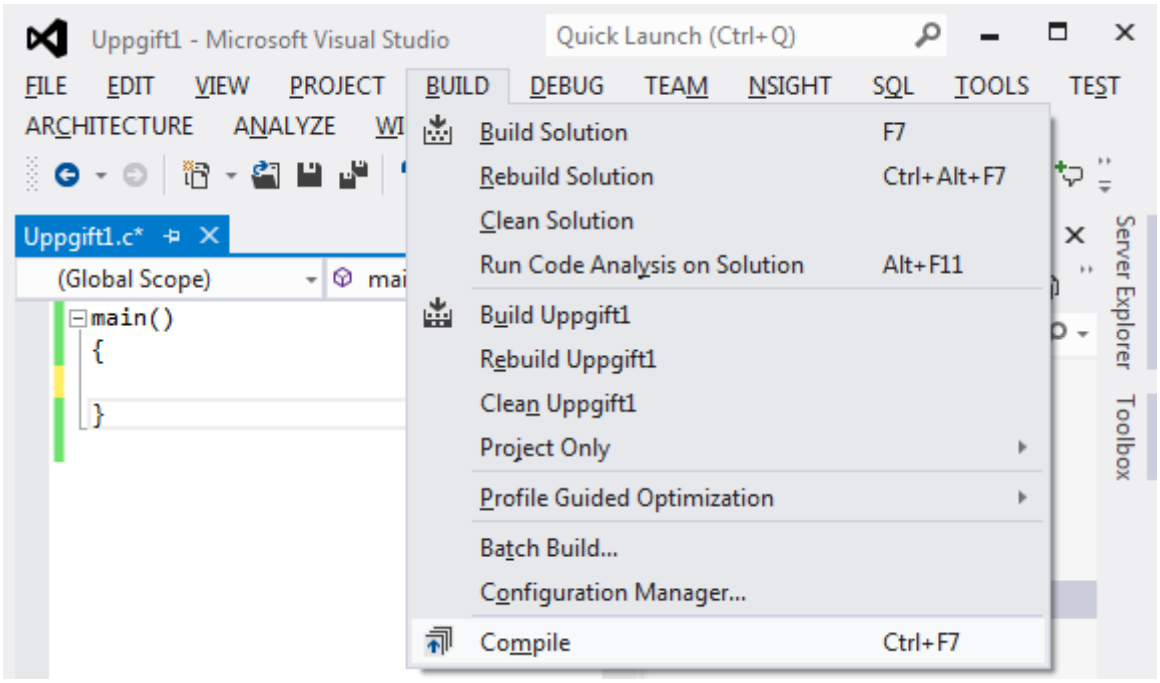
Med valet Add - New Item visas ett formulär där vi kan välja typ av fil och ge den ett namn. Välj typen C++, ge filen ett namn MED EFTERNAMNET (extension) .c. På så sätt talar vi om för VS att det är C vi arbetar med och inte C++.



Skriv ditt första c-program så att det blir så här:

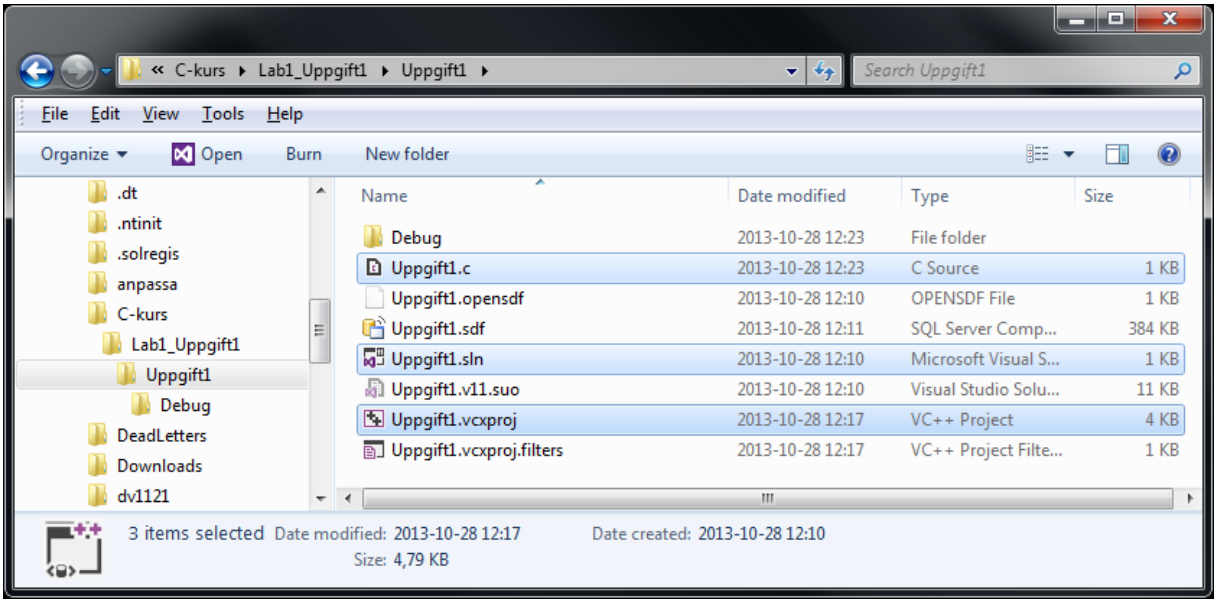


Nu ska din källkodsfil först *kompileras* till en s.k. objektfil, därefter ska den *länkas* ihop med andra s.k. c-bibliotek. För att kompilera väljer du från menyn *Build* alternativet *Compile*.



Om du får något felmeddelande, försök tolka det och korrigera i ditt program samt kompilera igen.

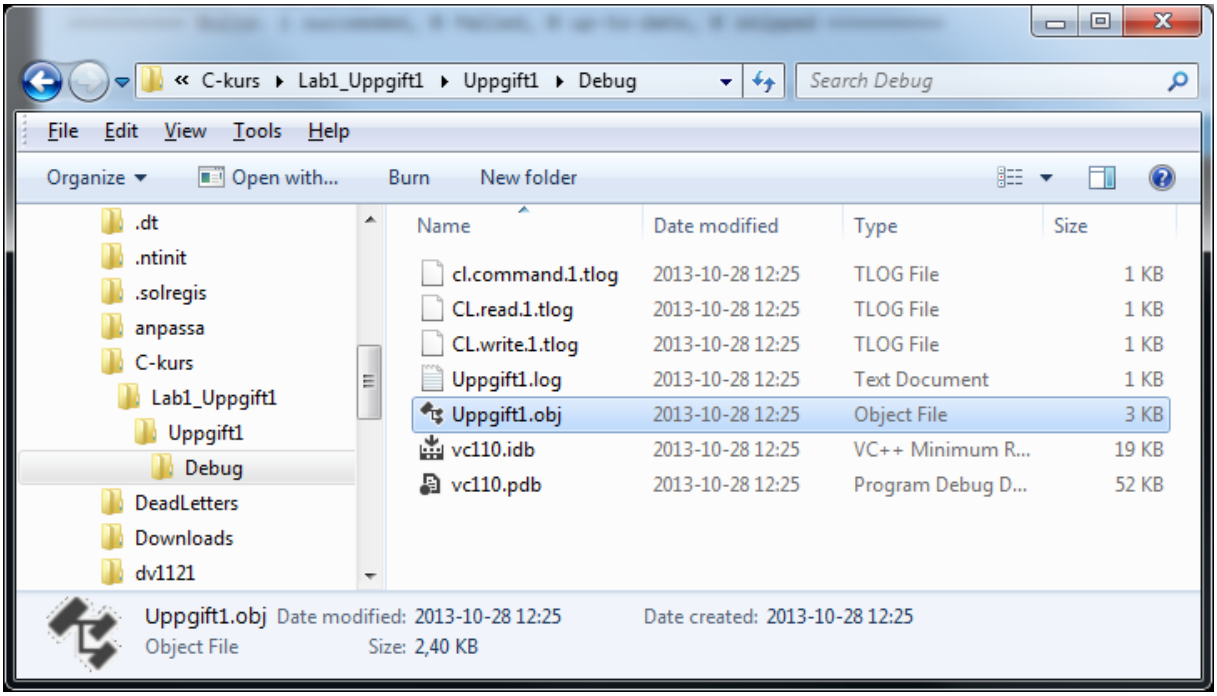
Nu ska du se vilka filer som VS har skapat: öppna en Utforskare (Explorer) i Windows (enklast med windowstangenten + E). Bläddra dig fram till den plats där du valde att placera ditt projekt.



I mappen du valt för ditt program finns din källkodsfil (motsvarar Uppgift1.c ovan) samt ett antal filer som VS skapat. VILKA filer som ingår i ett projekt (program) finns i filen .vcxproj. Ett (eller flera) projekt kan bilda en s.k. Solution. I filen .sln finns en lista på vilka projekt som ingår i aktuell solution. Informationen i dessa två filer är det som VS använder för att visa i Solution Explorer.

De enda filer du behöver hålla reda på och ta säkerhetskopior på är de du skapar och namnger själv, alla andra filer kan genereras igen.

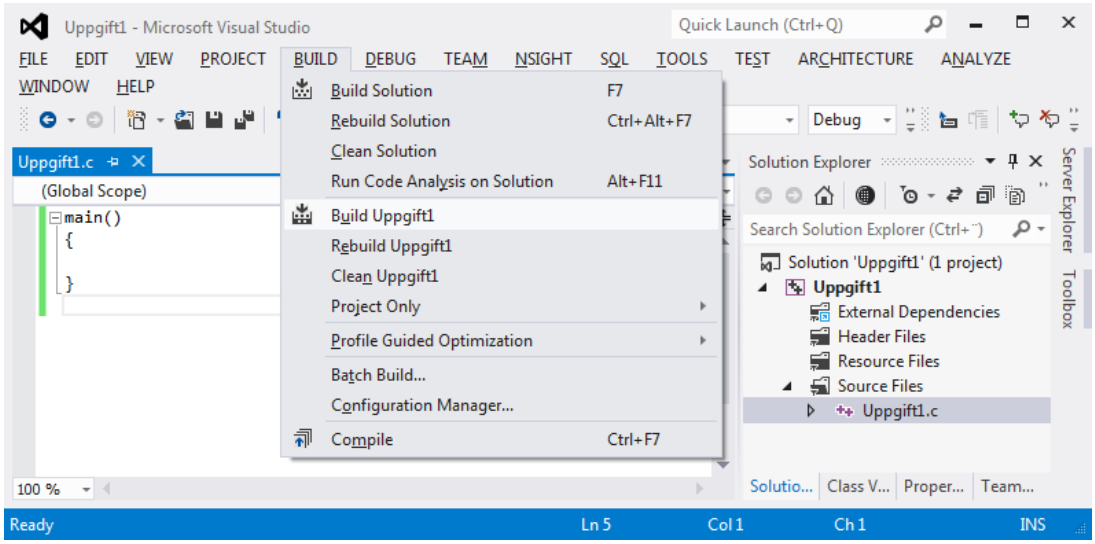
Öppna nu mappen Debug.



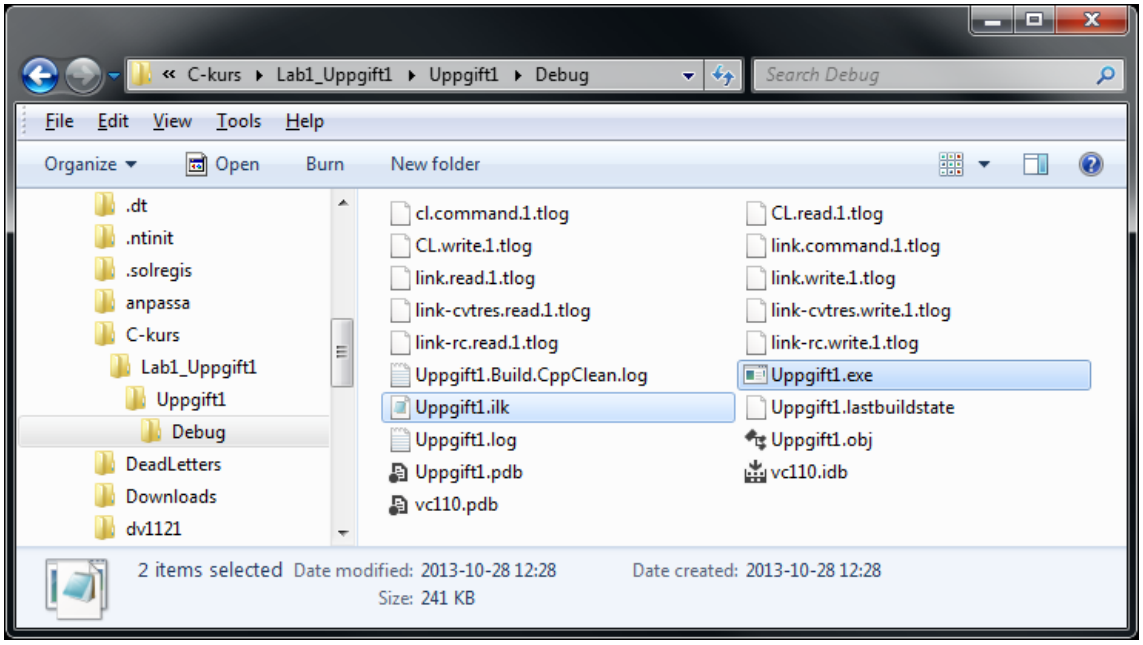
Här finns filer som har med skapandet av den exekverbara filen att göra. Du hittar en fil med efternamnet .obj samt en del andra filer. Det är .obj som så småningom ska ingå i det färdiga programmet.

Nästa steg i processen att skapa ett körbart program, är att länka din .obj-fil med standarddelar i C.

I VS väljer du Build - Build *ditt projektnamn.*



Plocka fram filhanteraren igen och se vilka filer du har nu i mappen Debug.

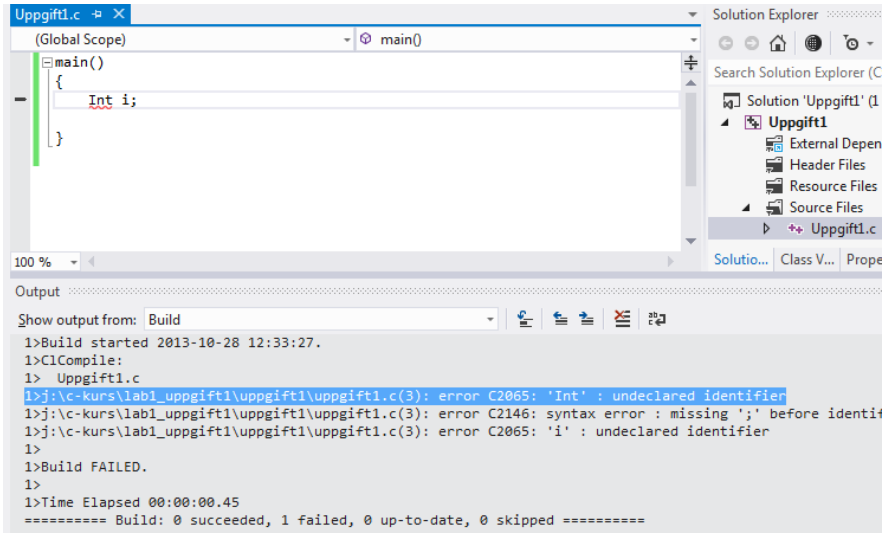


Nu har du en körbar fil med extension .exe. En fil .ilk visar att länkning har skett. Du kan dubbelklicka på din exe-fil för att starta programmet, men du ser inget hända eftersom programmet ännu inte gör något.

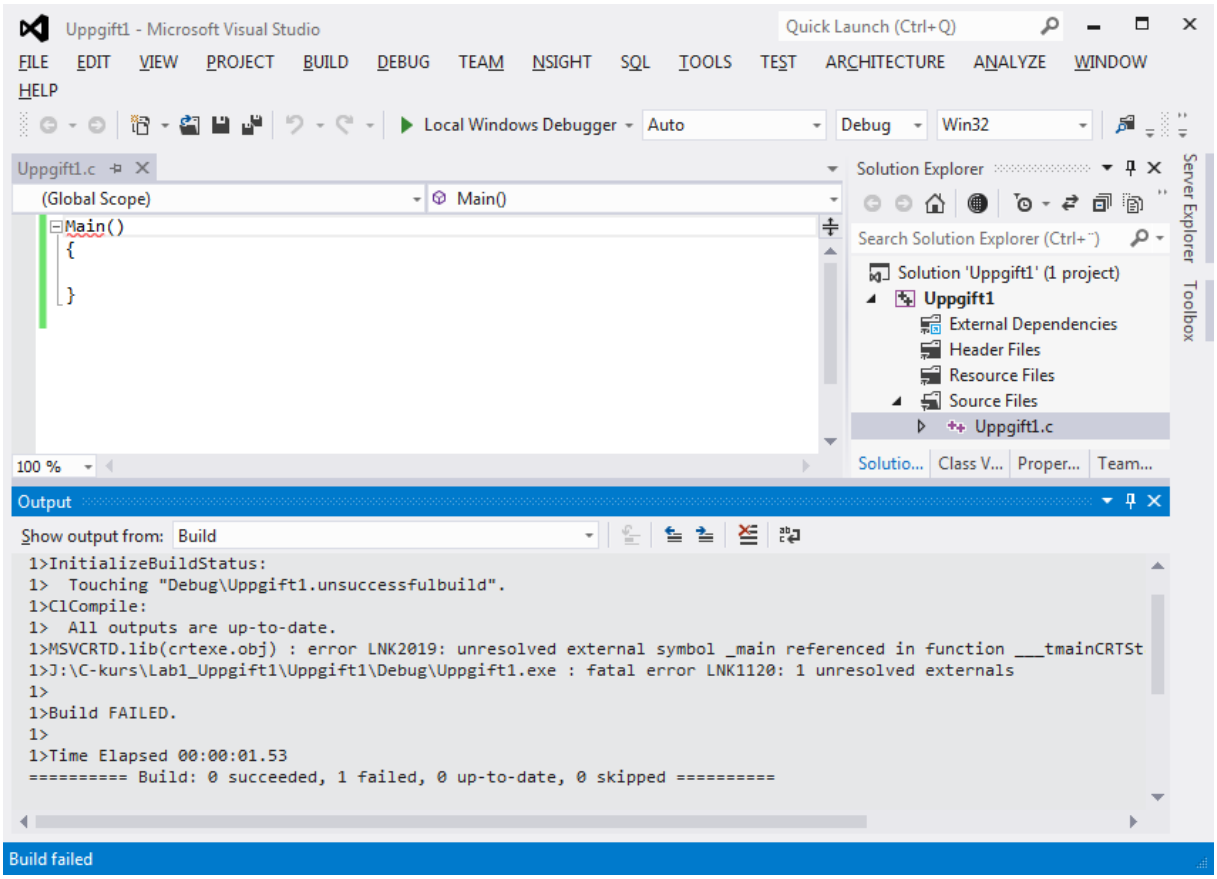
De filer som inte heter .c, .obj eller .exe använder VS för att optimera sitt arbete. Ibland händer det att VS tappar bort sig och man får konstiga fel. Radera då alla filer i dessa mappar utom de du gjort själv. Kompilera och länka därefter.

Del 3, Felsökning

Ibland stavar man fel eller skriver något annat som kompilatorn inte känner igen. Man får ett *kompileringsfel*. I nedanstående exempel står det felaktigt Int i stället för int. Dubbelklicka på felraden i ”Output”-fönstret så markeras den felaktiga raden i programmet. Korrigera och kompilera igen.



Följande fel ger inget kompileringsfel, däremot ett *länkningsfel* i nästa steg (Build):



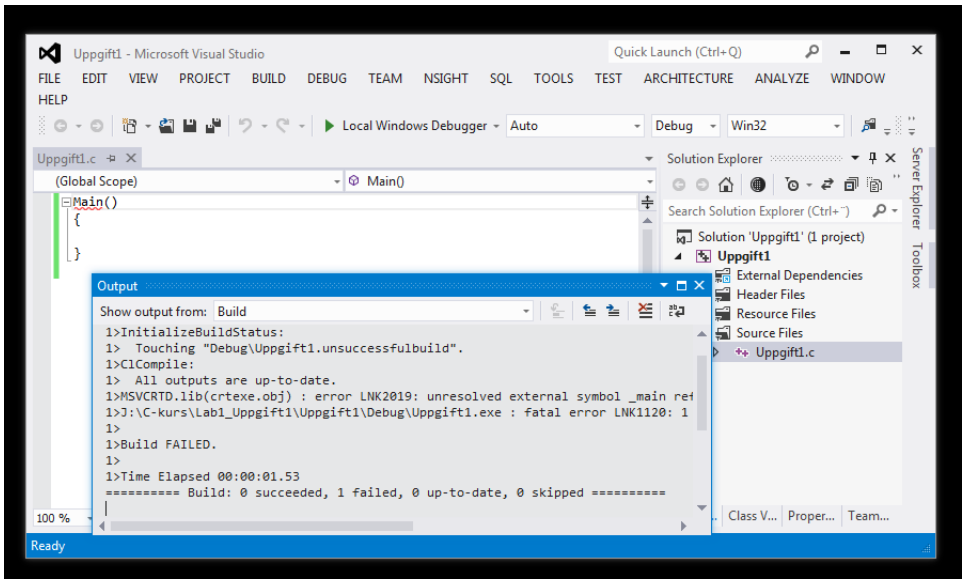
Förklaring: ett program kan bestå av flera filer. Det är tillåtet att ha en s.k. funktion som heter Main i någon av filerna. Vid länkningen söker länkaren efter funktionen main, eftersom den måste finnas i ett färdigt program. I detta fall saknades den.

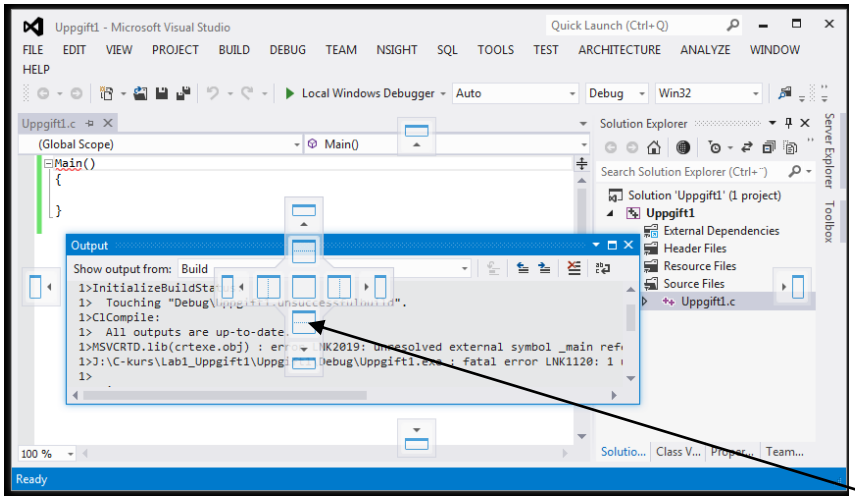
I allmänhet bryr man sig inte om att skilja stegen kompilering och länkning åt. I VS sker automatisk en kompilering om man direkt väljer Build.

Tips: låt VS skriva ut radnummer så blir det lättare att hitta. Från menyn välj Tools - Options -Text Editor - C/C++ och kryssa i Line numbers.

Del 4, Fönsterhantering i VS

I VS finns massor av flikar och delfönster. Dessa kan kopplas loss och flyttas runt, t.o.m. till andra bildskärmar (i fall man har flera). För att kopplas loss ett delfönster (flik) drar man bara i fliken med muspekaren. Nedan har fönstret Output kopplats loss.





Det "flytande" fönstret kan kopplas fast på olika ställen. En speciell figur illustrerar det. Samtidigt som du drar runt fönstret kan du föra muspekaren till något av alternativen i denna placeringsvy och släppa fönstret där.

Del 5, Programmeringsuppgifter

Uppgift 1

Nu är det dags att ditt program gör något vettigt, börja med att låta det skriva ut ditt namn. Ändra så att det blir

```
#include <stdio.h>

main()
{
	printf("Hej jag heter Göran");

	getchar();
}
```

Om ditt namn innehåller å, ä eller ö kommer något annat tecken att skrivas ut. Det beror på att Windows använder kodtabell enligt ANSI-standarden, medan ett konsollprogram använder ASCII-standarden.

Förklaring: kärnan i C innehåller inget för att hantera bildskärm eller tangentbord. Vill vi skriva ut något måste vi tala om att vi vill använda delar utanför kärnan, vi gör en "include". I detta fall inkluderar vi stdio.h (betyder standard input output). Är du nyfiken kan du högerklicka på ordet stdio och välja "Open document".
Extension h talar om att det är en "header"-fil, ligger i filens början. Vinklarna < > indikerar att filen finns i den fördefinierade mappen för standardfiler. När du så småningom skriver egna h-filer blir det i stället #include "minfil.h" (med citattecken).
Raden getchar(); gör så att programmet när det är klart, väntar på att användaren trycker en tangent. Utan denna rad hinner vi inte se vad programmet skriver innan det avslutas.

Uppgift 2

Ändra programmet så att själva utskriftssatsen blir mera generell, den ska kunna skriva ut något från en variabel.

```
main()
{
	char* förnamn = "Göran";
	printf("Hej jag heter %s\n", förnamn);
	getchar();
}
```

Förklaring: Alla uttryck i C har ett värde. Värdet av uttrycket "Göran" är den adress i minnet där första bokstaven (G) ligger. Detta värde (adressen) läggs i en pekarvariabel, *förnamn.* I C beskriver man vad för typ av data som pekaren pekar till med *typ**. *char* förnamn* betyder alltså att det som pekas ut av *förnamn* ska tolkas som ett tecken.

* kallas också innehållsoperator, man kan även skriva *char *förnamn* och säga att **förnamn* betyder innehållet i minnet på den adress som finns i förnamn.

Med vårt tidigare ritsätt för att beskriva variabeln *förnamn* blir det:

namn:	förnamn
plats i minnet:	....
värde:	adressen där 'G' finns
typ:	adress till ett tecken

I printf ligger en platshållare %s som indikerar att där ska skrivas en text (string). Vad som ska skrivas ut kommer efter ett kommatecken. Att fundera på: hur "vet" printf hur lång texten är?

Där man skriver \n kommer en radmatning att göras.

Uppgift 3

Ändra programmet så att det även skriver ut din ålder.

```
main()
{
	char* fornamn = "Göran";
	int alder = 21;
	printf("Hej jag heter %s och är %d år", fornamn, alder);

	getchar();
}
```

Förklaring: %d är en platshållare för ett heltal. Nu har vi två platshållare i det som ska skrivas ut. Det som ska skrivs ut kommer i en lista efter texten (med kommatecken emellan). I Visual Studio är det tillåtet att använda åäö i *identifierare* (namn på något), men en del kompilatorer tillåter bara det engelska alfabetet a-z. Det är därför en bra ide att undvika åäö för att kunna flytta sitt program till någon annan miljö.

I programmering används vanligen engelsk text genomgående på alla identifierare.

Vid programmering gör man *indrag*, texten börjar en bit in på raden. Sådana indrag görs alltid med tabuleringstangenten, ALDRIG MELLANSLAG.

Uppgift 4

Skriv ett program enligt modellen ovan som beräknar och skriver ut area och omkrets för en rektangel. Tips: deklarera heltalsvariabler för side1, side2, area, circumference.

Uppgift 5

a) Före decimalreformen i England 1971 hade man som valuta pund, shilling och pence. Det gick 12 pence på en shilling och 20 shilling på ett pund. (England is going metric, inch by inch).
Skriv ett program som delar upp ett belopp uttryckt i pence i pund, shilling och pence.
Exempel: 1026 pence blir 4 pund, 5 shilling och 6 pence. Använd heltalsdivision (/) och modulooperatorn (%), inte + eller -.

b) Skriv ett program som räknar ut hur många pund, shilling och pence som ska lämnas tillbaka till en kund som handlar för 3 pund, 9 shilling och 3 pence och som betalar med en 5-pundsedel.

Uppgift 6

Skriv ett program som räknar ut hypotenusan i en rätvinklig triangel (Pythogaros sats). I C-kärnan finns inte "roten ur", man måste plocka med ett s.k. mattebibliotek. Det inkluderas på samma sätt som stdio.h:

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

I denna uppgift är det lämpligt att räkna med *flyttal*. Deklarera variabler som float i stället för som int. För att skriva ut flyttalet x med printf blir det

```
printf("Här kommer ett flyttal: %f", x);
```

(Vid kompilering fås en varning att sqrt egentligen jobbar med typen double, vi struntar i den varningen just nu.)

Uppgift 7

Om tredjegradsekvationen $ax^3 + bx^2 + cx + d = 0$ har en reell rot och därmed två komplexa rötter så kan den reella roten bestämmas med Cardanos metod:

Låt

$$r = \frac{b}{a} \qquad p = \frac{c}{a} - \frac{r^2}{3} \qquad q = \frac{d}{a} - \frac{cr}{3a} + \frac{2r^3}{27}$$

$$t = \frac{4p^3+27q^2}{27} \qquad s = \sqrt[3]{\frac{\sqrt{t}-q}{2}}$$

då blir den reella roten $x = s - \frac{p}{3s} - \frac{r}{3}$

x^y beräknas med pow(x,y)

$\sqrt{x}$ beräknas med sqrt(x)

Skriv ett program som kan lösa ut en reell rot ur en tredjegradsekvation. Prova med ekvationen $4x^3 - 12x^2 + x - 3 = 0$

(svar: x = 3.00)

Om ekvationen inte har en reell rot kommer någon av beräkningarna att spåra ut och programmet kraschar. Vi återkommer till hur man hanterar det.

==========