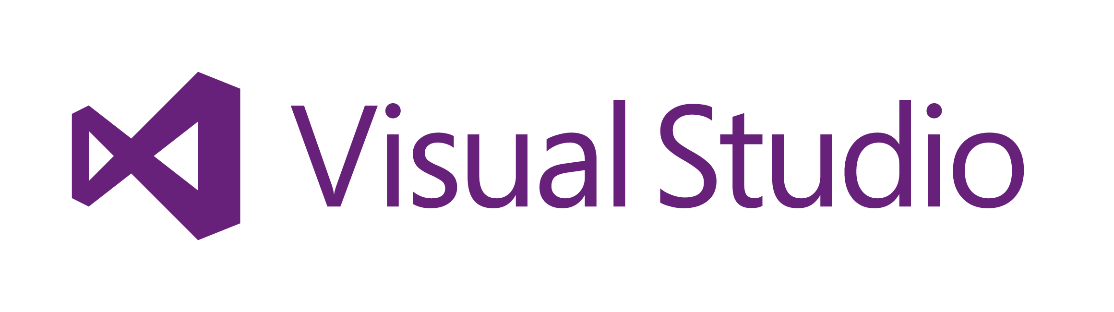
****

## Mi az a Visual Studio?

A Microsoft Visual Studio egy integrált fejlesztői környezet (IDE). Segítségével programozhatunk (többek között) Windows-ban futtatható programokat. A fejlesztési folyamatot több eszközzel is segíti

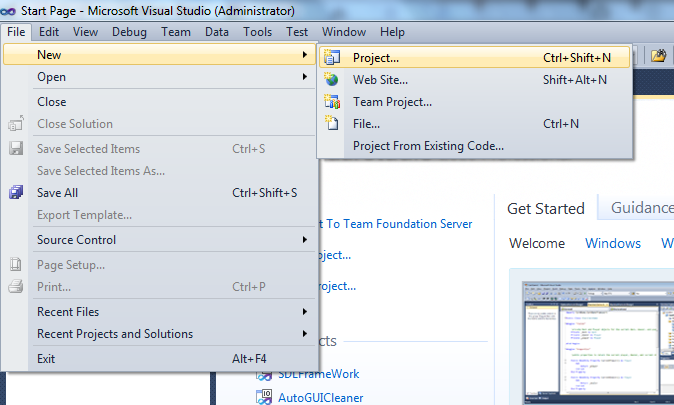
(IntelliSense, kódrefaktorálás, Debugging stb). Mi C++ nyelvű programozásra fogjuk használni, de több nyelvet is támogat. Apropó nyelvek - porold le az angoltudásod, vagy tegyél készenlétbe egy szakszótárat, Google Translate-et!

*Személyes felhasználásra letölthető a Visual Studio Express verziója a* [*Microsoft honlapjáról*](http://www.visualstudio.com/downloads/download-visual-studio-vs)*.(Express verzióból a C++-os/****Desktop*** *kell) (2015 előtt).* 2015-től a Visual Studio Community Edition vette át az Express szerepét, ami letölthető a <https://www.visualstudio.com/downloads/> oldalról - ez a régi Professional verziónak felel meg nagyjából.

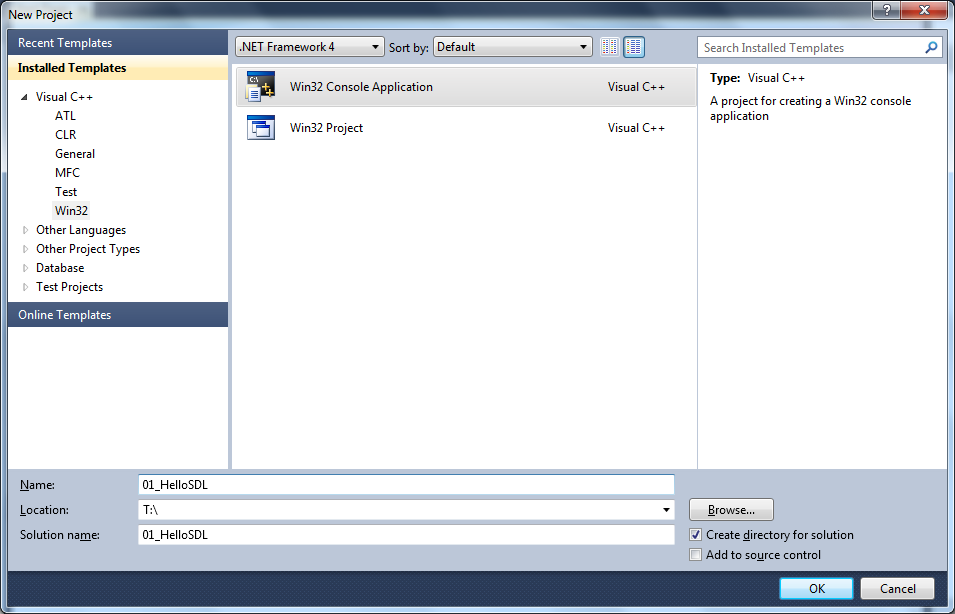
Ezen túl az egyetemi polgárok számára a [Dreamspark-on keresztül](http://msdnaa.elte.hu) a „nagyobb” verziók is elérhetők. A gyakorlaton Visual Studio 2010/2012-vel készített programokat nézünk, de a kódok Visual Studio 2013-ban is működnek.

## Új project létrehozása

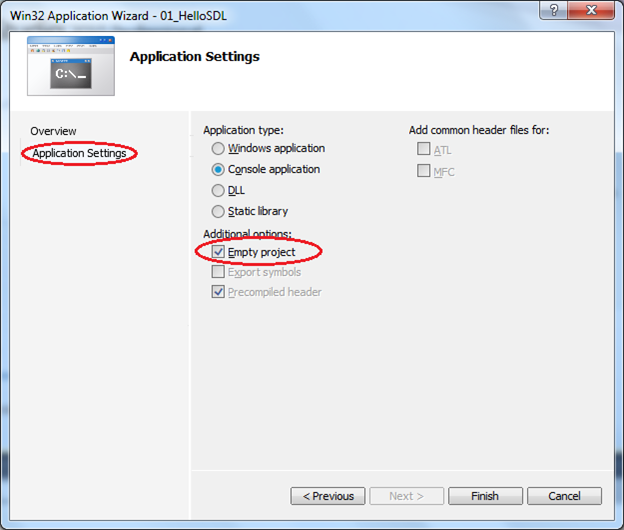
A Visual Studio megnyitása után kattintsunk a *File/New/Project* mennüpontra (vagy használjuk a Ctrl+Shift+N billentyűkombinációt):

****

A megjelenő menüben válasszunk a *Win32 Console Application*-t, nevezzük el, és mentsük el a kívánt könyvtárba. Célszerű a grafika laborban a T meghajtót használni, lényegesen meg fogja könnyíteni a dolgunkat a későbbiekben ismertetett külső könyvtárak elérése miatt. Otthon is érdemes a grafika projecteket tartalmazó könyvtár elérését T-re állítani.

****

A továbbiakban arra figyeljünk, hogy az Application Settings-ben jelöljük meg az *Empty project*-et, és kész is :)



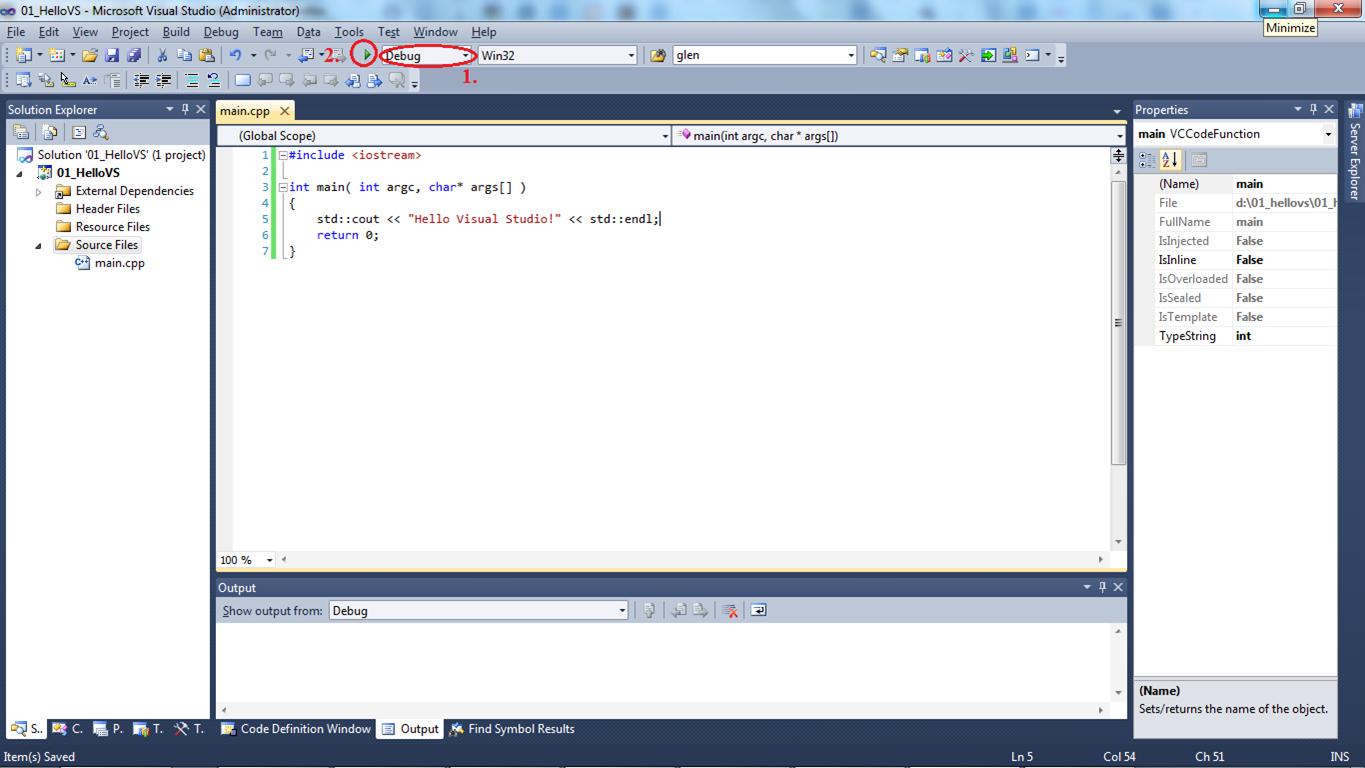
## A project fordítása és futtatása

A fordításhoz válasszuk ki a kívánt **fordítási profilt**. Visual Studio-ban többféle fordítási profilt használhatunk, a *Release* és *Debug* alapból megtalálható, de készíthetünk sajátokat is (pl. 32 és 64 bites stb.)

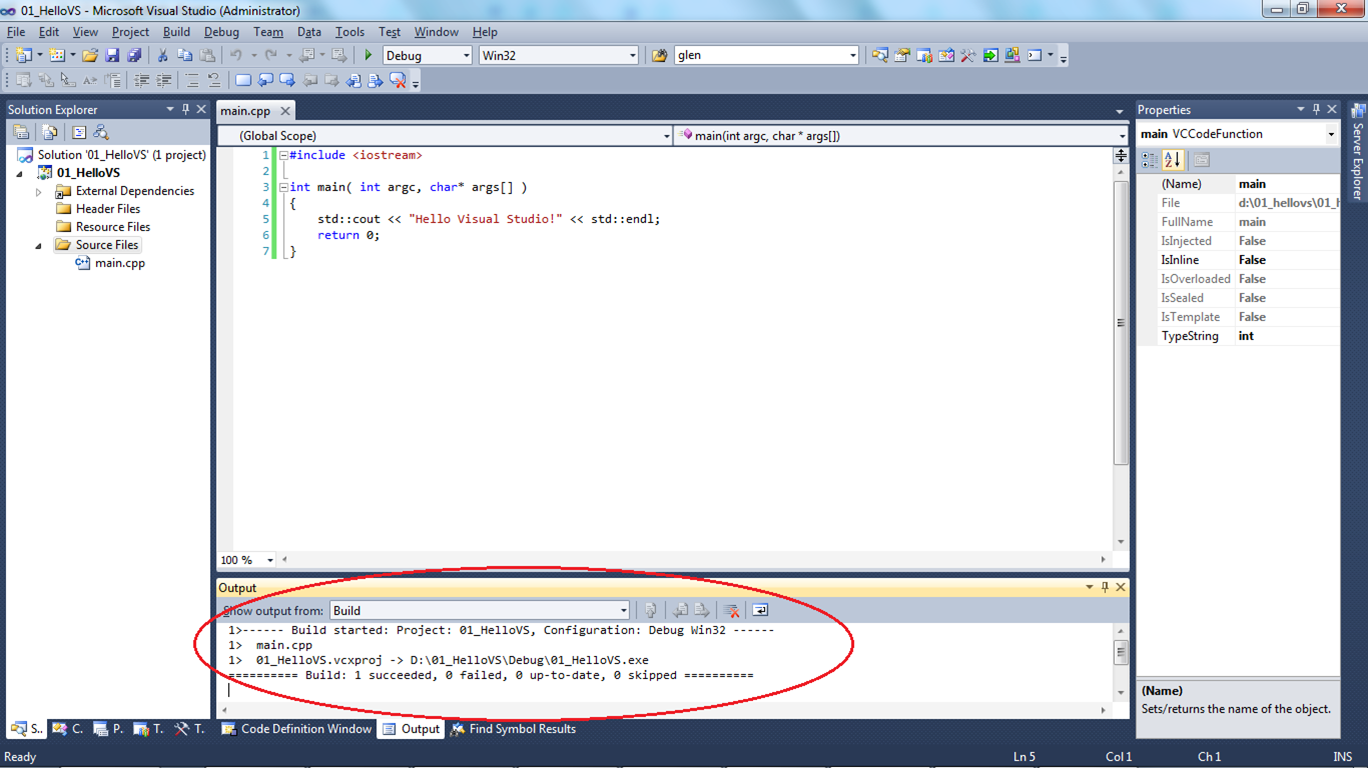
A Debug módban fordított kódokba extra utasításokat is beletesz a rendszer, amik a hibakeresést segítik (és a hatékonyságot csökkentik), illetve plusz utasítások kerülnek be, amik Release módban nincsenek (pl. inicializálatlan változók azonos kezdőértéket kapnak Debug módban, míg Release-ben memóriaszeméttel indulnak).

Ezután fordítsuk és futtassuk a kódunkat:

* Fordítani a Ctrl + Shift + B vagy F7 billentyűkkel tudjuk (vagy válasszuk ki a *Build/Build solution* menüpontot)
* Fordítani és futtatni az F5-tel lehet (vagy a „Play” gombra rányomva)



A fordítás eredménye (jobb esetben) a következőképpen néz majd ki:



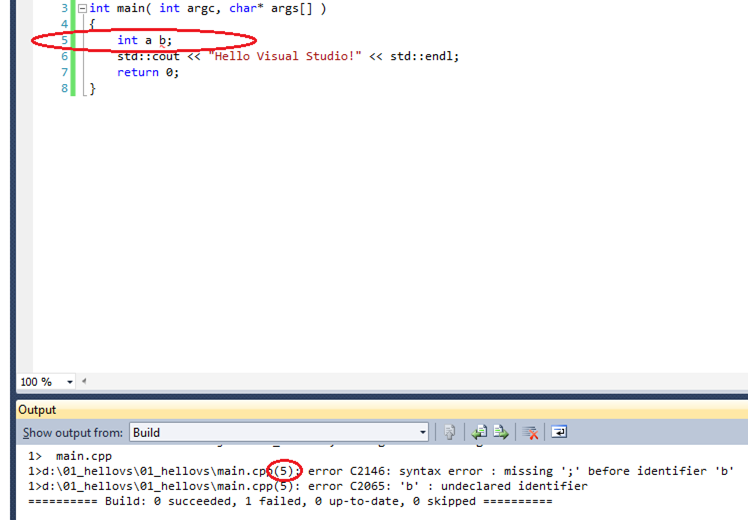
## Segítség, valami hiba van!!!

A fordítás során nem a “Build: 1 Succeeded, 0 Failed, …” üzenet jelent meg, hanem az alábbi ablak ugrott fel, mi a teendő?

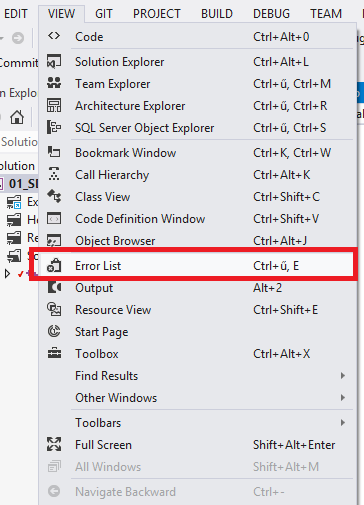


Ilyenkor ne ess pánikba! Ösztönösen jönne az ötlet, hogy a “Yes”-re kattintva a korábbi sikereiden lovagolj. Esetleg annyira takarnád a takarni valót, hogy a “Do not show this dialog again”-t is beixelnéd? **NE TEDD!** Ekkor ugyanis nem fogod tudni megkeresni és kijavítani a hibá(i)d, cserébe a régi eredmény fog futni újra és újra. Senki sem szeret hibázni, de ne próbáld leplezni, hogy te is halandó vagy :) E helyett kattints a “**No**”-ra, és vedd fel a harcot!

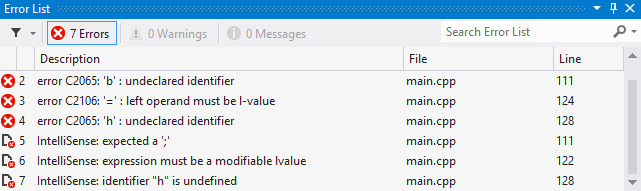
A Visual Studio ki fogja írni, hogy hol és mi a hiba. A hibaüzenetre duplán kattintva oda is ugrik a kódban. Kezd a **legelső hibá**val, mindig **olvasd el** és **értelmezd** a hibaüzenetet, és próbáld kijavítani a legjobb tudásod szerint.



A hibák (***error*** - ezeket mindenképpen ki kell javítani, hogy forduljon a kód) és figyelmeztetések (***warning*** - ezek csak olyan kódrészletekre hívják fel a figyelmed, ahol előfordulhat, hogy nem az történik, amit szeretnél, de javításuk nélkül is lefordul a kód, de lehet, hogy nem működik jól) egy rendszerezettebb nézete a View menü Error List opciójával kapcsolható be:



Kétszer rákattintva egy üzenetre itt is odavisz a Visual Studio a hibás sorra:



## Külső könyvtárak beállításai

Az új projekt létrehozása után be kell állítani a programjaink által használt külső könyvtárak elérési útvonalait:

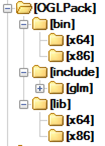
A félév során a következő rendszereket használjuk:

* **SDL**: cross platform multimédiás réteg, ablakozásra, grafikára, hanghardverhez és egyéb bemeneti eszközökhöz való hozzáféréshez
* **OpenGL**: ld. később
* **GLM**: nyílt forráskódú matematikai könyvtár

Az órai programokban használt rendszereket

* Visual Studio esetén [**ebben**](http://cg.elte.hu/~bsc_cg/resources/OGLPack.zip)

az OGLPack-ban lehet megtalálni, ami a többi grafikához tartozó package-dzsel együtt az alábbi linken található: [http://cg.elte.hu/~bsc\_cg/resources](http://cg.elte.hu/~bsc_cg/resources/)



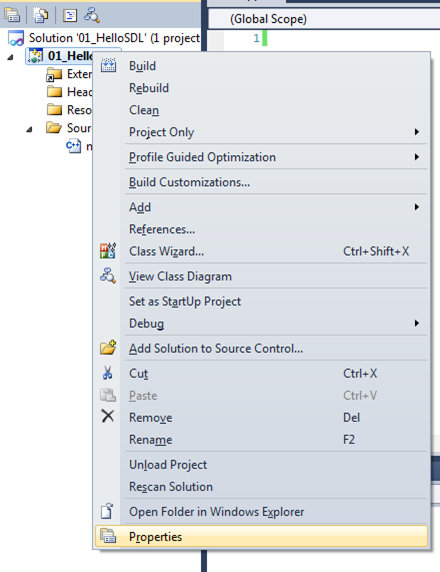
A fenti rendszereket használó programjainknak be kell állítani, hogy az egyes könyvtáraknak

* hol találhatóak a *header* fájljai
* hol találhatóak a fordításkor szükséges *lib* fájljai
* opcionálisan megadhatjuk, hogy Visual Studio-ból történő futtatás esetén hol keresse a programokhoz szükséges *dinamikusan szerkesztett könyvtár*akat (DLL-ek) a rendszer

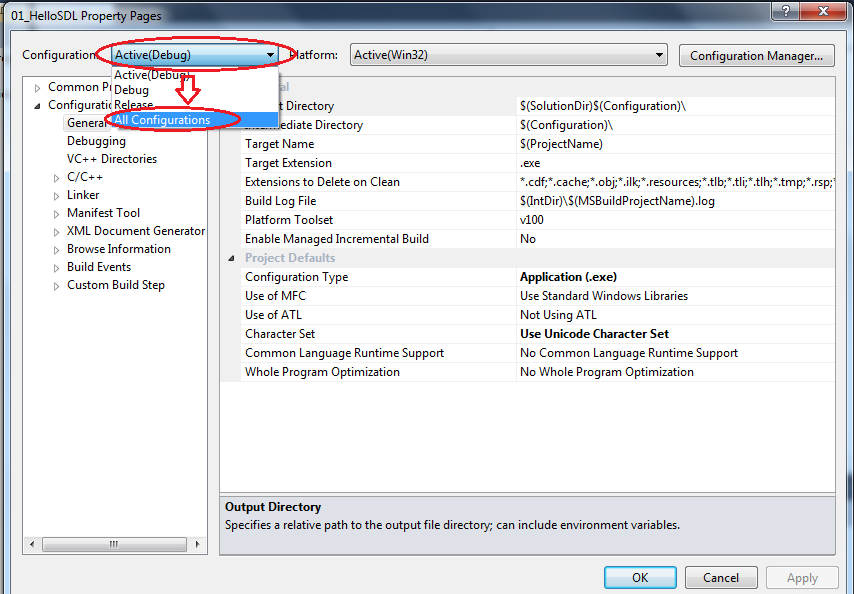
Más könyvtárakhoz egyéb dolgokat is be kell állítani (pl. fordításhoz szükséges külső futtatható fájlok elérési útvonalait.)

Egyes külső könyvtárak más-más lib-eket vagy DLL-eket igényelnek a különböző fordítási profilokhoz (Debug/Release). Az órákon 32 bites kódokat futtatunk, de a 64 bites fordításhoz szükséges lib és dll fájlok is megtalálhatóak az összes általunk használt rendszert összefoglaló OGLPack-ben. Csak annyi a dolgunk, hogy beállítsuk, hogy Debug és Release módban is ugyanazokat az include és lib könyvtárakat keresse a VS fordításkor és ugyanazt a bin könyvtárat futtatáskor.

A külső könyvtárak elérését a *Project/Properties* ablakon belül a *Configuration Properties* részen állíthatjuk be. Ehhez kattintsunk a Project nevén jobb egérgombbal (nem a Solution-én! Általában félkövérrel szedi az aktív project nevét a bal oldali *Solution Explorer*-ben), és a lenyíló menüben válasszuk ki a *Properties* alpontot. A megjelenő ablak bal oldalán pedig jelöljük ki a *Configuration Properties*-t.



Legegyszerűbb, ha *All Configurations*-t használva állítjuk be az eléréseket:



Ezután csak a megfelelő elérési útvonalat kell megadni. Ez a Grafika laborban az alább feltüntetett helyen szereplő **OGLPack** segítségével gyerekjáték. Ezért célszerű otthon is az OGLPack-ot a nekünk kényelmes helyen létrehozott és T-meghajtónak átnevezett munkakönyvtárban tárolni.

Ez utóbbihoz legyen <útvonal> a kitömörített OGLPack-ot tartalmazó mappa elérési útvonala, a lezáró \-vel együtt (pl.: *C:\mappa\akarmi\01\* ). Ekkor a Parancssorban kiadott

subst t: <útvonal>

utasítással létrejön egy virtuális T: meghajtó, ami újraindításig megmarad. Az aktuális könyvtárat a

subst t: .

paraméterekkel lehet T: meghajtóvá tenni, a D: meghajtót pedig a következőképpen:

subst t: d:\

Virtuális meghajtó törlésére a

subst /D t:

utasítás szolgál.

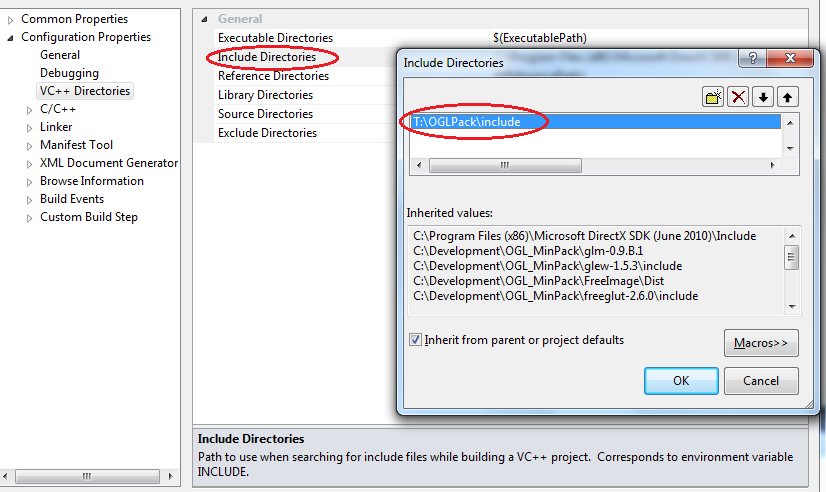
Az OGLPack használatát segítő két .bat fájl a következő:

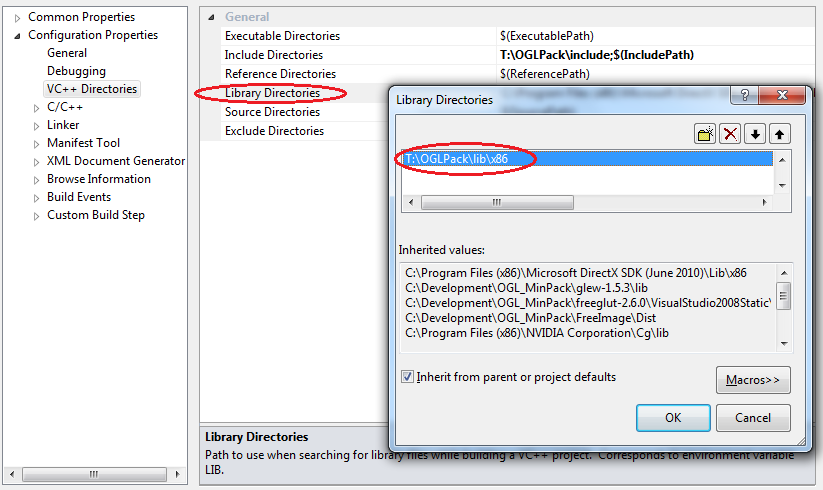
* [mount\_t.bat](http://cg.elte.hu/~bsc_cg/resources/util/mount_t.bat) az aktuális könyvtárat behelyettesíti egy új virtuális T:\ meghajtó gyökerébe
* [delete\_script.bat](http://cg.elte.hu/~bsc_cg/resources/util/delete_script.bat) a Visual Studio által generált összes átmeneti fájlt törli, a futtatási könyvtárából indulva, annak összes alkönyvtárát bejárva

A bal menüben a *Configuration Properties/VC++ directories*-nál:

* **Include** fájlok: T:\OGLPack\include
* **Lib** fájlok: T:\OGLPack\lib\x86

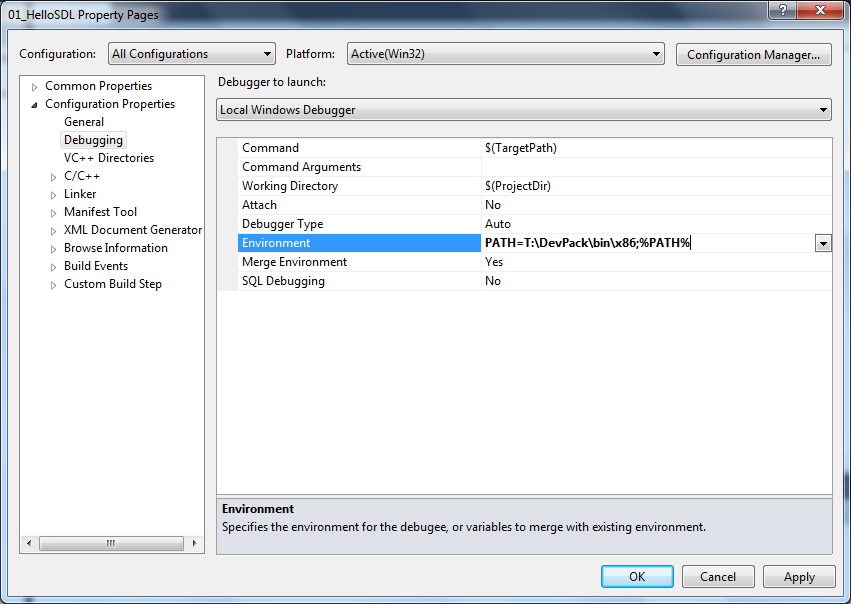
(32 bites libek)



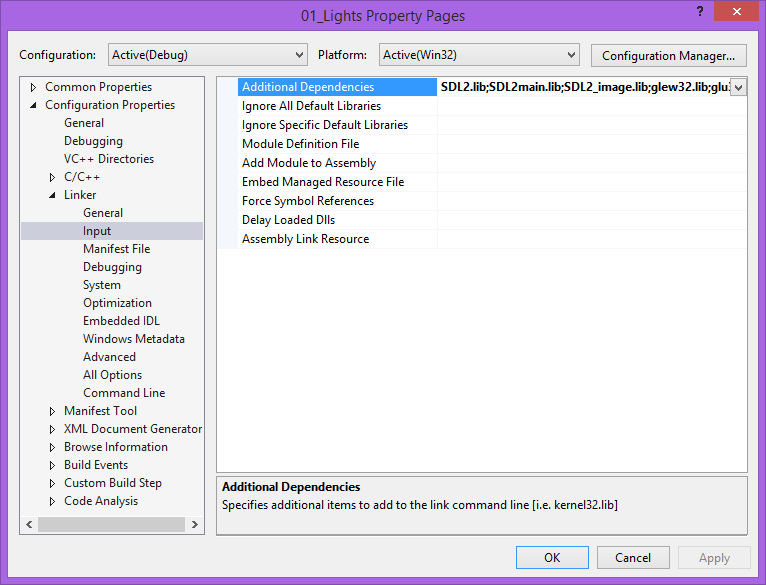


A bal menüben a *Configuration Propertie.-s/Debugging*-nál:

* **Environment** részhez: PATH=T:\OGLPack\bin\x86;%PATH%



Végül már csak hozzá kell adni a projekthez azokat a .lib fájlokat, amelyek segítségével az általunk használt API-k belefordíthatóak a kódba. Ezt szintén az All configurions-t bekapcsolva a *Linker/Input/Additional dependencies* résznél tehetjük meg:



A következő lib fájlokra lesz szükségünk a félév OpenGL-es része során:

SDL2.lib

SDL2main.lib

SDL2\_image.lib

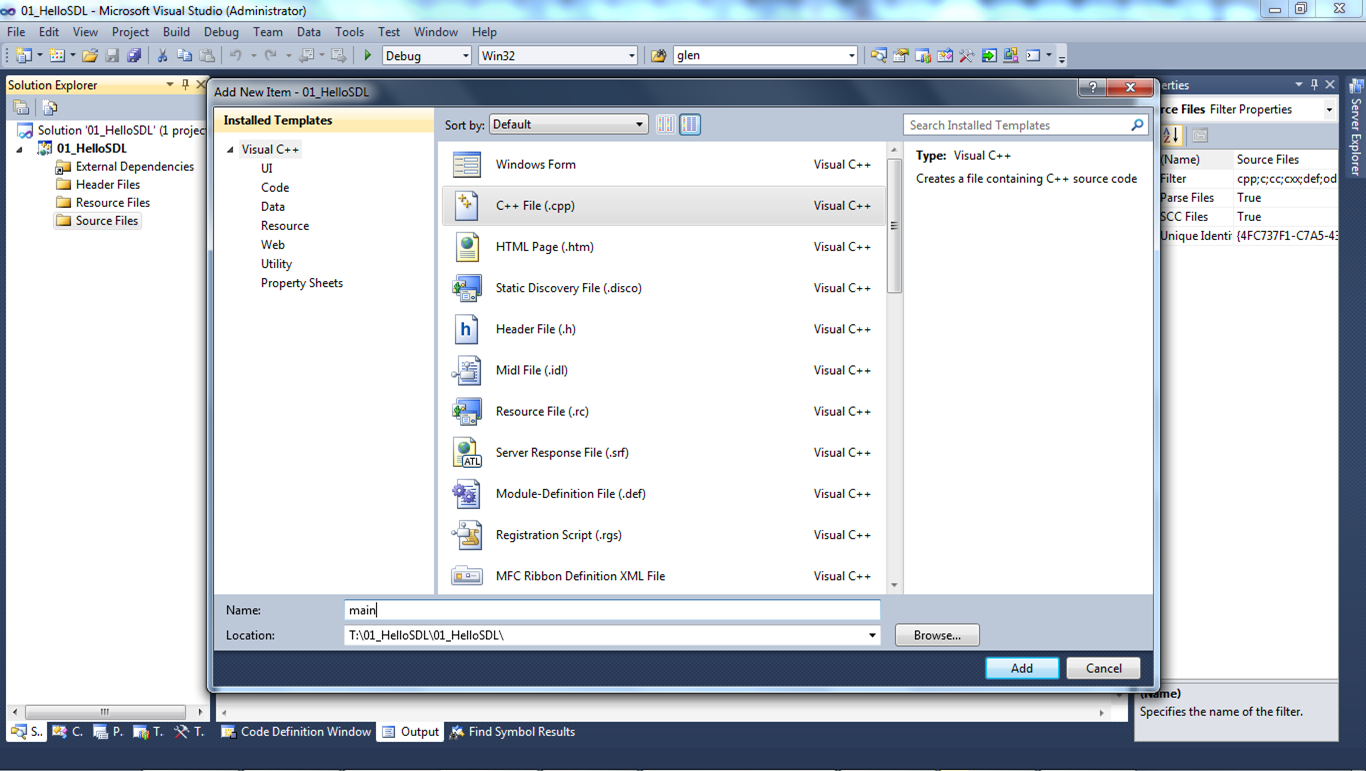
glew32.lib

glu32.lib

opengl32.lib

## Forrásfájl hozzáadása a project-hez

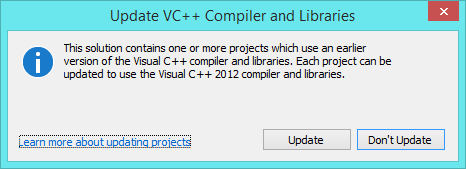
Jobb egérgombbal kattintva a *Solution Exploler*-ben a megfelelő könyvtárra fel fog a Visual Studio dobni egy párbeszédablakot, ahol kiválaszthatjuk a forrásfájl típusát (mi a .cpp forráskódot és .h header-t tartalmazóakat használjuk főként). Elnevezhetjük, hozzáadhatjuk, és már szerkeszthetjük is!



## Órai kódok fordítása Visual Studio 2010 fölött

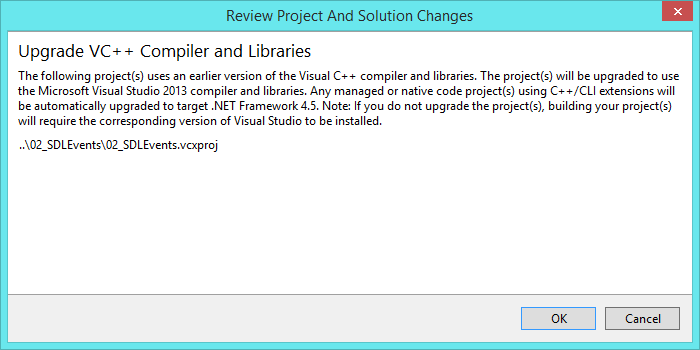
A gyakorlatokon legtöbbször Visual Studio 2015-mal fogunk dolgozni. Amikor egy korábbi Visual Studio-val készített solution fájlt (.sln) töltünk be, akkor induláskor a rendszer figyelmeztet arra, hogy a solution bizonyos beállításait frissíteni kell. Engedjük meg, hogy ezt automatikusan megtegye a program.

Visual Studio 2012 esetében az Update gombra kell kattintani:



(Visual Studio 2012)

Visual Studio 2013-nál pedig egyszerűen az OK-ra:



(Visual Studio 2013)

Az eredményeket egyszerűen csak jóvá kell hagyni ezután. Figyeljünk arra, hogy az Express Desktop verziókkal dolgozzunk!

# Megjegyzések linuxos megszállottaknak

Az SDL platformfüggetlen környezet, ezért természetesen semmi akadálya nincsen, hogy Linux alatt fejlesszünk. Itt röviden összefoglaljuk a telepítéshez szükséges tudnivalókat:

## SDL2 telepítése:

1. Töltsük le a <http://www.libsdl.org/download-2.0.php> címről a legfrisebb forrást
2. Az SDL könyvtárába belépve a szokásos módon configuráljunk és instalálljuk az alábbi parancsokkal:
   * ./configure
   * make
   * sudo make install
   * sudo ldconfig
3. Alternatív megoldás magunkfajta lusta Ubuntu (és valószínűleg minden Debian-alapú) Linuxot használó usereknek:
   * sudo apt-get install libsdl2-dev

Ezek után az SDL-es programok fordulni fognak a gépünkön. Feltéve, hogy az include-hoz és a link-eléshez jól adjuk meg a könyvtárakat.

Konkrét példaként nézzük meg, hogy a <http://cg.elte.hu/~bsc_cg/gyak/01/01_HelloSDL.zip> példakönyvtárban a main.cpp által leírt programot szeretnénk futtatni. Ha egy külön könyvtárba tesszük a main.cpp fájlt, lefordítani így tudjuk:

g++ -I/usr/local/include/SDL2 -c main.cpp

(Mindez feltételezi, hogy telepítéskor az include fájlok -- esetünkben pl. az SDL.h -- a /usr/local/include könyvtárba kerülnek.

Fordítás során pedig a libSDL2.so fájlt kell megtalálni, ezt a -l opcióval és a helyét tartalmazó könyvtár (/usr/local/lib) megadásával adhatjuk meg:

g++ main.o -L/usr/local/lib/ -lSDL2 -o test

Ez a partancs a futtatható állományt test-nek nevezi, a

./test

paranccsal a példaprogramot el is tudjuk indítani.

## SDL\_image telepítése

A későbbi feladatokhoz szükség lesz az SDL\_image könyvtár használtatára is, melyet innen lehet letölteni: <https://www.libsdl.org/projects/SDL_image/>

A forrás kitömörítése után, az SDL\_image könyvtárba belépve a szokásos parancsokkal lehet telepíteni

* + ./configure
  + make
  + sudo make install
  + sudo ldconfig

Csomagkezelőből való telepítés Ubuntun:

* sudo apt-get install libsdl2-image-dev

Ezek után linkeléshez a -lSDL2\_image kapcsolót is hozzá kell fűzni a parancshoz. Például az előző linkelési példa kiegészítve:

g++ main.o -L/usr/local/lib/ -lSDL2 -lSDL2\_image -o test

(A fordítás ugyanúgy mehet, mivel az include fájlok szintén a /usr/local/include/SDL2 könyvtárban vannak.)

Ez a rövid leírás a linuxos parancssori (fapados) fordítási lehetőséget mutatta be. Természetesen IDE segítségével (pl. NetBeans) lényegesen barátságosabb felületen tudunk programozni.

## OpenGL telepítése

A további parancsok Ubuntura vonatkoznak, más esetben természetesen a megfelelő csomagkezelőt kell használni.

sudo apt install glew-utils

sudo apt install libglm-dev

Ezután a `glewinfo` programot futatva megkapjuk a gépünk által támogatott OpenGL verziót. Ha ez:

* 3.0+, akkor elvileg minden jó lesz, egy-két dolog esetleg nem
* 1.x, 2.x, akkor baj van.

Ha emelni próbáljuk a verziót, akkor a videókártya-driverek frissítése szükséges. Az Nvidia driverek jellemzően az *nvidia-???* csomagban vannak. AMD esetén a *radeon*, és az *amdgpu* csomagokkal lehet próbálkozni.

Ezután telepítés:

sudo apt install libglew-dev

Fordítási parancs:

g++ main.cpp MyApp.cpp -I/usr/include/SDL2 -L/usr/local/lib/ -lSDL2 -lSDL2\_image -lGL -lGLEW -o test

## Néhány technikai megjegyzés

A textúrákat is használó projektek a GLU könyvtárat is használják. Előfordulhat, hogy a MyApp.cpp-ben az `#include <GL/GLU.h>` sor hibát jelez, ekkor kikommentelhető, és a -lGLU kapcsoló is hozzátevendő a fordítási parancshoz.

Az újabb rendszerek panaszkodnak, hogy olyan kiterjesztést használunk, ami esetleg később nem fut. A hiba a “---GLM: GLM\_GTX\_transform is an experimental extension…” üzenettel jön. A megoldás, hogy a kódunk elejére (pl. MyApp.h-ba) beillesztjük az alábbi sort:

#define GLM\_ENABLE\_EXPERIMENTAL