

1. LABOR

AZ ÚJ KÖRNYEZET ÉS A REFERENCIA TÍPUS

Általános információk

1. iMSc pontok

Az első laboron még nem szerezhető iMSc pont. A többi laboron szerezhető iMSc-s feladatok megoldásainak **AUT** **portálra** *.zip fájlba csomagoltan való feltöltési határideje az aktuális labortól számított három nap. Ha egy feladatban kérdések szerepelnek, a pontok csak akkor fogadhatók el, ha mellékletben egy **igényes** jegyzőkönyv is szerepel a kérdésekre vonatkozó válaszokkal. iMSc pont szerzésére bármely hallgató jogosult, aki az előtte lévő feladatokkal már végzett (laborvezető ellenőrzi a haladást).

2. Házi feladat

Házi feladat pontozási irányelvekről bővebben a „HF_IRANYELVEK.pdf” szól.

Kötelező feladatok

1. Emlékeztető, ismerkedés az új környezettel

Visual Studioban készíts egy új *Empty project* projektet *GettingStarted* néven, amely valósítsa meg az alábbiakat:

- Olvasson be egy egész számot (N), majd egy szöveget (S)
- N-t és S-t adja át egy *függvénynek* (F)
- F-ben S-t a standard kimenetre *írja ki* N-szer

2. Forráskód dekompozíció

Az előző feladat F függvényét valósítsd meg kétféleképpen:

1. Külön *printing.cpp* fájlban definiáld, majd abban a fájlban, amiben használsz (*main.cpp*), *extern* kulcsszóval jelezd a fordítónak, hogy egy másik *.cpp-fájlban keresse a definíciót
2. Töröld az *extern*-t, helyette hozz létre egy *printing.h*-t
 - a. csak a deklaráció szerepeljen benne, a definíció maradjon a *printing.cpp*-ben
 - b. oldd meg, hogy többszöri include-olás esetén ne legyen többszörös deklaráció

3. Kódrészlet debuggolás

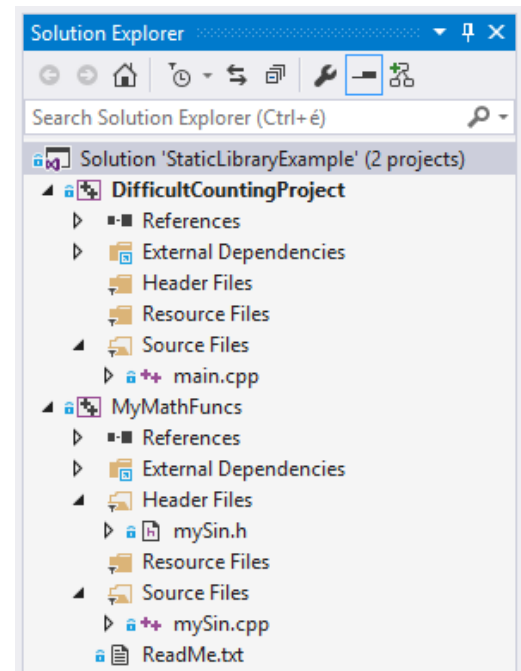
A Visual Studio eszközeit felhasználva futtasd utasításonként a következő kódrészleteket:

C- rossz	C-jó	C++
<pre>#include<stdio.h> void f(int i) { i=i+2; } int main(void) { int i=0; f(i); printf("%d\n",i); } /* A kimenet 0 */</pre>	<pre>#include<stdio.h> void f(int* pi) { (*pi)=(*pi)+2; } int main(void) { int i=0; f(&i); printf("%d\n",i); } /* A kimenet 2 */</pre>	<pre>#include<stdio.h> void f(int& i) { i=i+2; } int main(void) { int i=0; f(i); printf("%d\n",i); } /* A kimenet 2 */</pre>

4. Statikus programkönyvtár készítése

VS Community Edition esetén:

- Készíts egy Empty Projectet MyMathFuncs néven (és válaszd ki, hogy a Solution megegyezik a projekttel, vagy esetleg StaticLibraryExample néven is nevezhető)
 - A projekt legyen static library (projektre jobbklikk, Properties, Configuration type),
- készíts egy *mySin.h* és egy *mySin.cpp* állományt
- implementálj egy *double awesomeSin(double x)* függvényt a *sin(x)* tetszőleges felhasználásával
- projektre jobb klikk -> Build
 - ezzel elkészült a StaticLibraryExample.lib, amit bármely más C++ programnál fel tudsz használni
 - ez nem tartalmazza a *.h fájlokat



VS Code esetén: Kövesd a laborvezető instrukcióit és esetlegesen általa megosztott konfigurációs állományokat.

5. Külső statikus programkönyvtár használata

1. Az előző projekt Solution-jébe (StaticLibraryExample)
készíts még egy *Empty project-et* DifficultCountingProject néven
2. A projekt alatt: *References* -> *Add Reference* -> *pipáld ki a MyMathFuncs-t*
 - a. így a linker már ismeri fogja az elkészített statikus programkönyvtárat
 - b. azonban nincs hozzá deklaráció
3. készíts egy *main.cpp* állományt
 - a. enélkül egyrészt nem látszódná a következő lépés C/C++ menüpontja
4. Projektre jobb klikk -> *Properties* -> *Configuration Properties* -> *C/C++* -> *General* -> *Additional Include Directories* -> Tallózd be a másik projekt mappáját (ami tartalmazza a *mySin.h-t*)
 - a. ez a lépés azért kell, hogy a program megtalálja a *mySin.h-t*
5. írd példakódot a másik projektben felhasznált függvény segítségével
 - a. ne felejtse el include-olni a *mySin.h-t* a *main.cpp*-ben

VS Code esetén: Kövesd a laborvezető instrukcióit és esetlegesen általa megosztott konfigurációs állományokat.

Gyakorló feladatok

- [Referencia helytelen/helyes használata](#) a)-e)