A Code::Blocks fejlesztőkörnyezet

A Code::Blocks egy keretrendszer, amely sokféle platformon (Windows, Mac, Linux), elsősorban **C/C++** programozási nyelvekhez biztosít kényelmes **programfejlesztési környezet**et. Többféle fordítóprogramot képes magába integrálni (javasolt: GNU GCC); lehetővé teszi nyomkövető (debug) rendszer beépülését és kényelmes használatát (javasolt: GNU GDB).
Szolgáltatásai (többek közt): több program egyidejű szerkesztése; rugalmasan módosítható szintaxis kiemelés (syntax highlighting); interaktív kódkiegészítés (code completion).
Használati jog: nyílt forráskódú, szabadon felhasználható

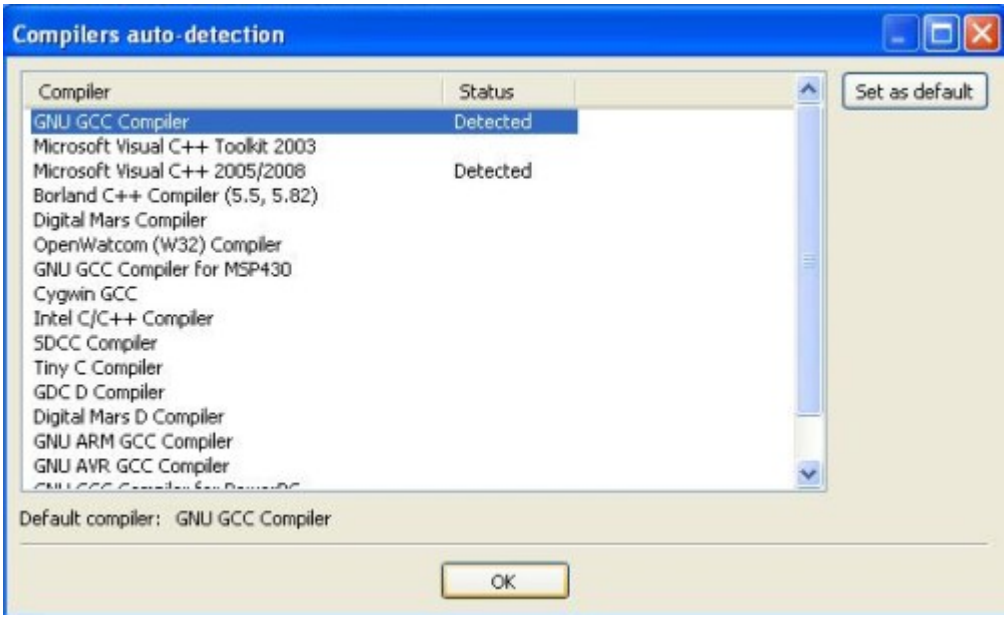
Letöltés, telepítés

A Code::Blocks letöltő helye: http://www.codeblocks.org

Itt a **Downloads** menüpontban a **Download the binary release** lehetőséget kell választani. Ezen belül ki kell választanunk, hogy milyen operációs rendszerre szeretnénk telepíteni. Windows esetén letölthető csak a fejlesztői környezet (a forrásfájl szerkesztő: *codeblocks-13.12-setup.exe*), vagy a fejlesztői környezet és a fordítóprogram együtt (*codeblocks-13.12mingw-setup.exe*). Ez utóbbi javasolt.

Használata

Windows alatt a Code::Blocks első elindításakor történik a fordítóprogram hozzárendelése. Ekkor jelzi ki, hogy mely – számára felismerhető – C vagy C++ fordítóprogramok találhatók a lemezen, és választásra kínálja, hogy melyik legyen az alapértelmezett (default).



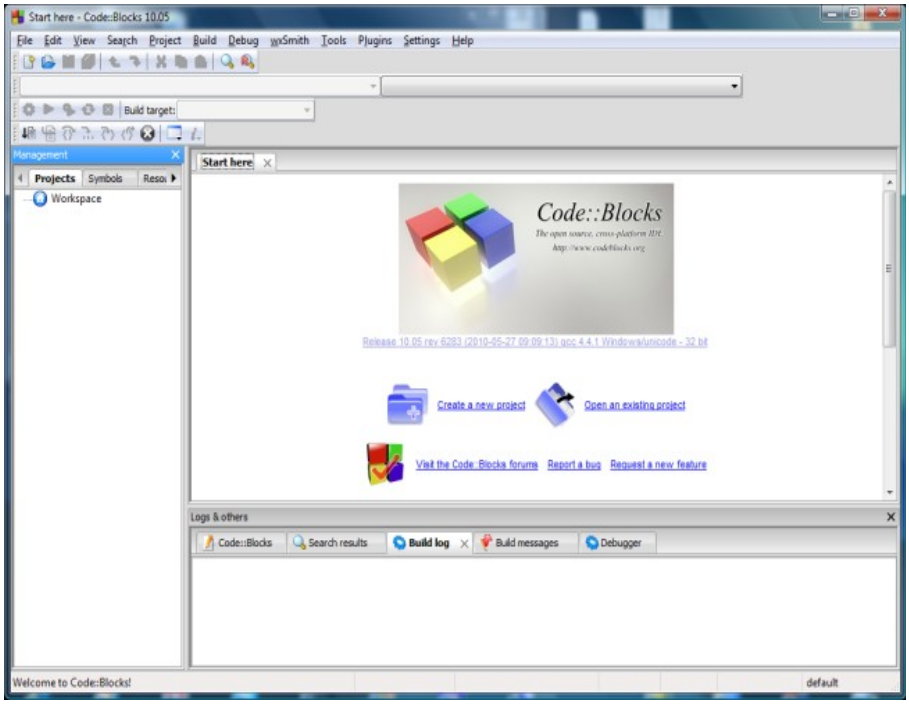
1. ábra: A Code::Blocks első indításakor feltérképezi,hogy milyen fordítóprogramok találhatók a számítógépen.

1. Új projekt létrehozása

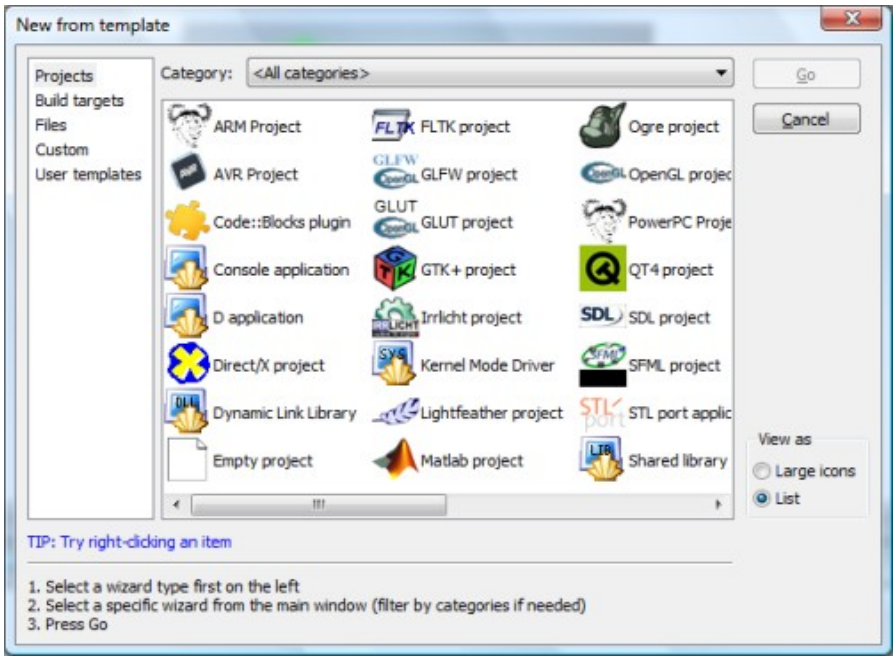
Projekt létrehozása azért lényeges, mert a létrehozandó program több fájlból álló rendszert alkot.

Például lehet több **forrásfájl** is (C nyelvű programot tartalmazó fájl; legalább egy, általában *main.c* nevű van), amelyekből a fordítóprogram készít ún. félig lefordított, *o* kiterjesztésű (ún. object) fájl(okat), majd a **futtatható kódot** (tehát a processzor számára „érthető” gépi utasításokat) tartalmazó, exe kiterjesztésű fájlt. Hogy ez a sok fájl ne keveredjen se egymással, se más programok fájljaival, külön, sajátos könyvtárrendszerbe csoportosítja a Code::Blocks. Többek közt ennek leírását jelenti a projekt, amelyet egy külön fájlban (cbp←Code Blocks Project) tárol.

A nyitó oldalon **File – New – Project** létrehozásakor válasszuk a konzol alkalmazást (**Console application**).



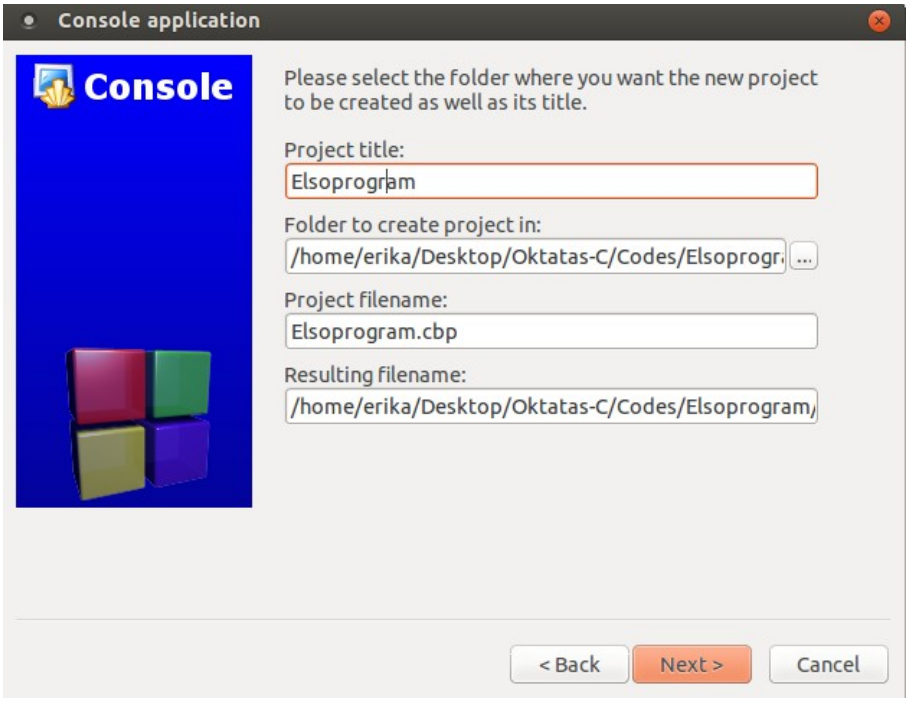
2. ábra: A Code::Blocks nyitó képernyője



3. ábra: Code::Blocks projektek

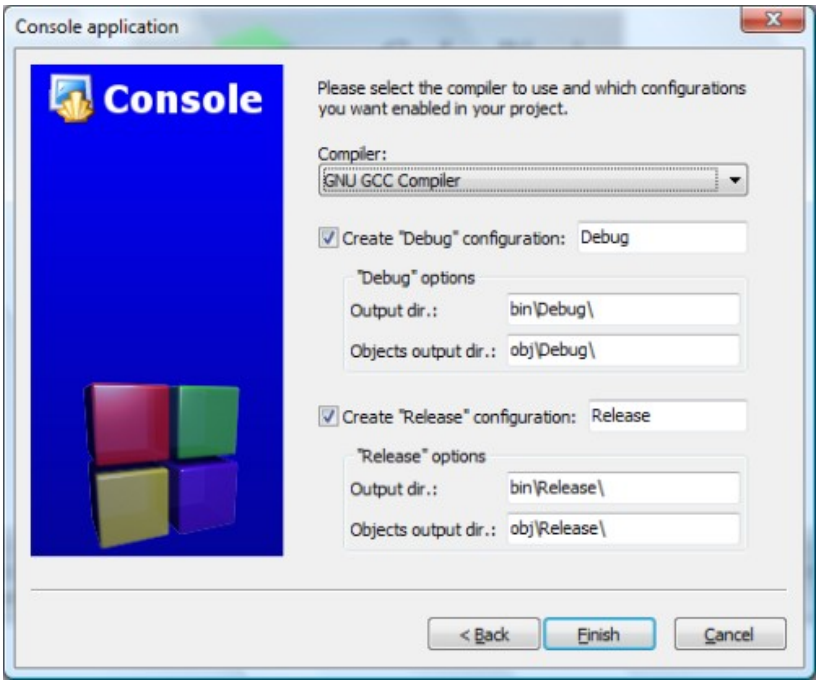
A **programozási nyelv (C) kiválasztása** után következnek a munkakörnyezet (könyvtárszerkezet) felépítését célzó lépések. A **projektnév** kitöltése után annak a **könyvtárnak a kiválasztása**

következik, amelyből (mint „szülőkönyvtárból”) nyílik a projekt nevével megegyező könyvtárunk a készítendő alkalmazásunk számára. Ebből nyílnak majd a speciális fájljainkat tartalmazó további könyvtárak. Az alsó két mező automatikusan kitöltésre kerül.



4. ábra: Konzolalkalmazás első paraméterező ablaka

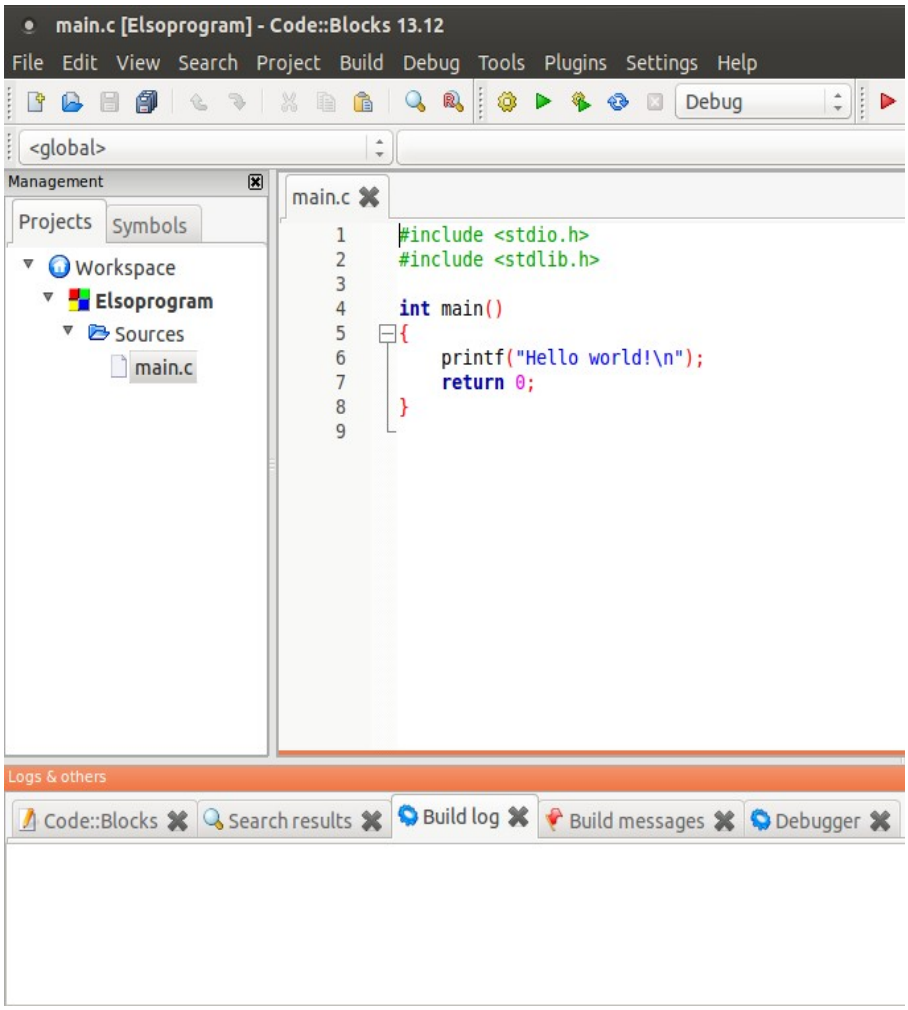
A fenti paraméterezés szerint a projektünket leíró fájl az „elsoprogram.cbp” lesz. Továbblépve döntünk a fordítóprogramról. Válasszuk a **GNU GCC compiler** fordítót. Beállítható még a futtatható, illetve a félig lefordított kódot tartalmazó fájl alkönyvtárának neve. Ezt érdemes a felkínáltnak megfelelően elfogadni.



5. ábra: Konzolalkalmazás második paraméterező ablaka

Az 5. ábrán látható, hogy kétféle célra van mód fejleszteni a programot. Fejlesztés alatt jó, ha a kód tartalmaz olyan információkat is, amely támogatja a nyomkövető programot (debugger). Ennek a kódjai (futtatható és object) kerülnek a **Debug** részben beállítottakba. A véglegesben már csak a minimálisan szükséges kódok találhatók. Ezek a fájlok a **Release** részben beállított alkönyvtár-párba kerülnek.

A Code::Blocks környezet kész a programszerkesztésre. A bal oldali **projekt management** fülön a **Sources**-t kibontva és duplán rákattintva a **main.c**-re, a jobb oldali részben egy kész, mintául szolgáló program forrása jelenik meg.



6. ábra: Az alapértelmezett C program

2. Fordítás

Fordítás előtt a **Build** menü **Select target** menüpontban válasszon az ún. debug (azaz fejlesztés alatt álló) és az ún. release (végleges) változat közül. Ettől függően a *bin/Debug* vagy a *bin/Release* katalógusba kerül az object és a futtatható fájl.

A **Build** menü **Build** menüpontjára vagy közvetlenül az eszközkészlet, fogaskereket mintázó ikonjára kattintsunk rá. A **fordítás** eredményeként a jobb oldali alsó ablaktartományban jelenik meg a fordító üzenete, többek közt, hogy „0 errors, 0 warnings”.
A **Build** menü **Compile current file** menüpontjára kattintva csak az aktuálisan kijelölt fájl kerül lefordításra (nem a teljes projekt).

3. Futtatás

A **Build** menü **Run** menüpontjára (vagy a zöld háromszög ikonra) kattintva lefut a program: kiírja a *Hello world!* üzenetet a képernyőre.
A fordítás és futtatás együttes végrehajtása a **Build** menü **Build and run** menüpontjának kiválasztásával lehetséges. Ennek ikonja a fogaskerék és előtte a zöld háromszög.

A terminál ablak betűméretének megváltoztatásához kattintson jobb egérgombbal a bal felső sarokban látható ablak ikonra. A felugró menüből **Properties** – **Font** fül.

4. Fordítás és futtatás közvetlenül az operációs rendszerben

A gcc fordító segítségével a C forrásfájl fordítása **Terminál ablakban** is történhet. Először be kell állítani, hogy az aktuális katalógus a forrásfájlt tartalmazó legyen. Itt kiadhatók az alábbi utasítások:

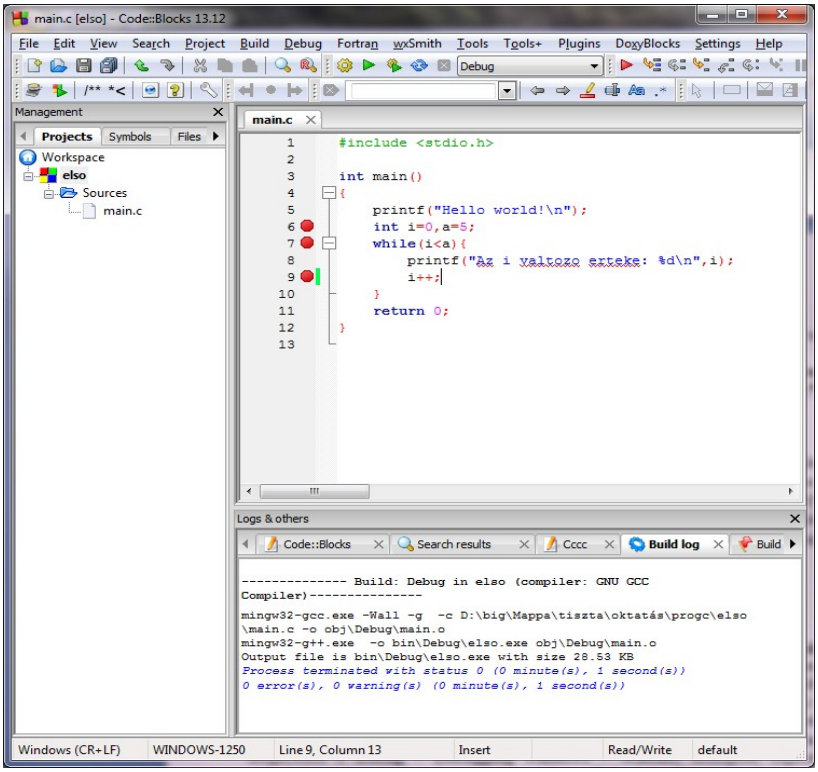
- gcc main.c → ennek hatására elkészül az alapértelmezett object fájl (a.out)
- gcc main.c -c → az elkészült object fájl neve megegyezik a forrásfájl nevével (main.out)
- gcc main.c -o sajat.out → az object fájlnak tetszőleges nevet lehet adni (sajat.out)

Több modulos program esetén a forrásfájlokat egymás után szóközzel elválasztva kell megadni.
Például: *gcc main.c elsomodul.c masodikmodul.c*

A lefordított kód futtatása a ./ paranccsal történik: ***./objectfájl***

5. Hibakeresés, nyomonkövetés

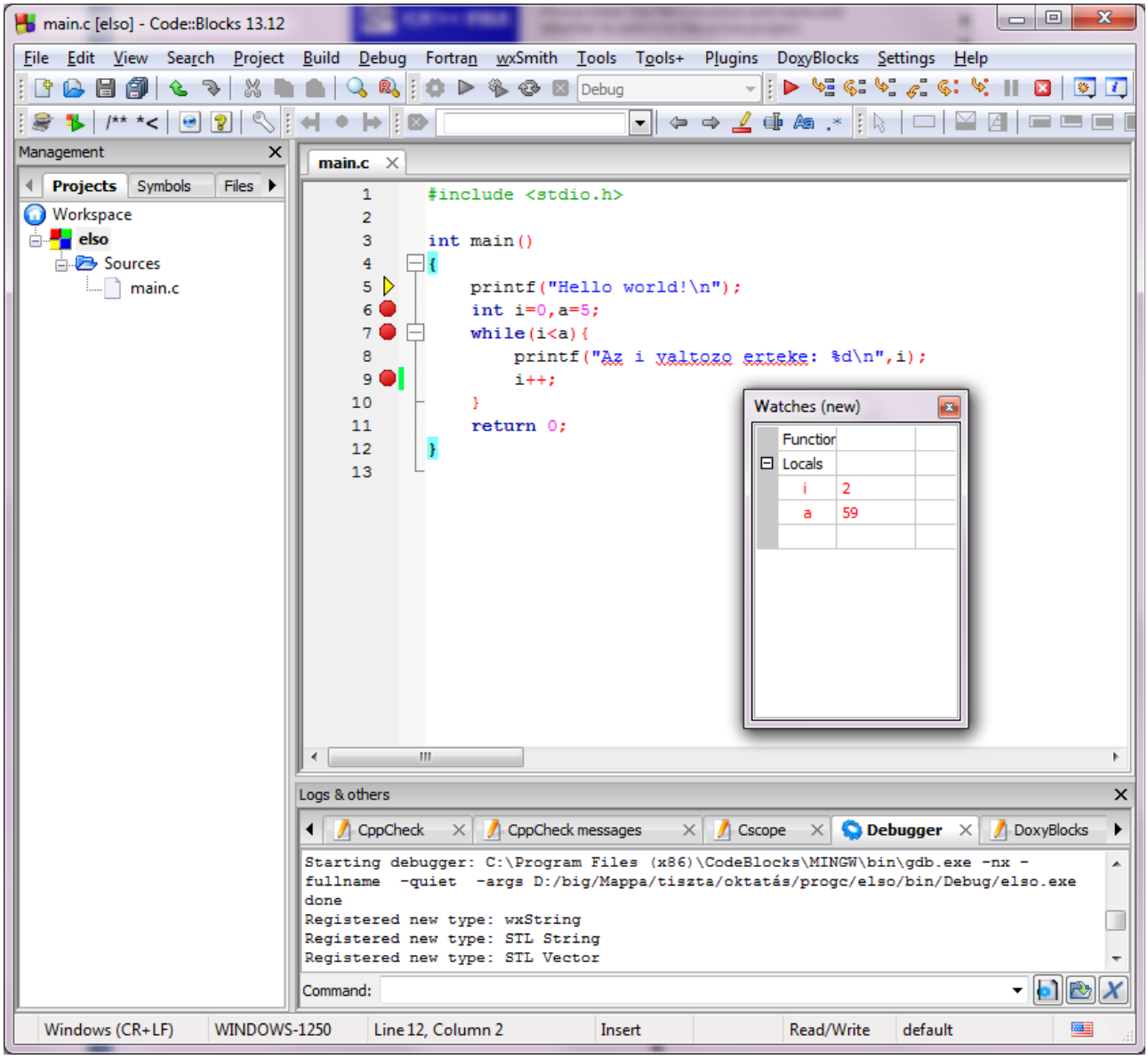
Először be kell állítani a programban a ponto(ka)t (**Debug – Toggle breakpoints**), amelytől kezdődően a program működését/állapotát (azaz a változók, argumentumok értékét) figyelemmel akarjuk kísérni.



7. ábra: Töréspontok kijelölése a programban

A példában az *a* és *i* változók értékének változását figyeljük. A programkód sorszámai után kattintva, megjelenik egy piros pont (breakpoint).

Ezután a **Debug – Start/Continue** menüpontot (vagy a piros nyíl ikont, Step into) kiválasztva a program a kijelölt pontig fut, majd elkezdődik a nyomonkövetés. A figyelni kívánt változókat, argumentumokat egyenként kell hozzáadni a figyelő ablakhoz. Vagyis egyesével ki kell választani ezeket, majd jobb egérgombbal rákattintani és a **Watch** menüpontot kiválasztani. A figyelőablak megnyitása a **Debug – Debugging windows – Watches** menüpont segítségével történik.



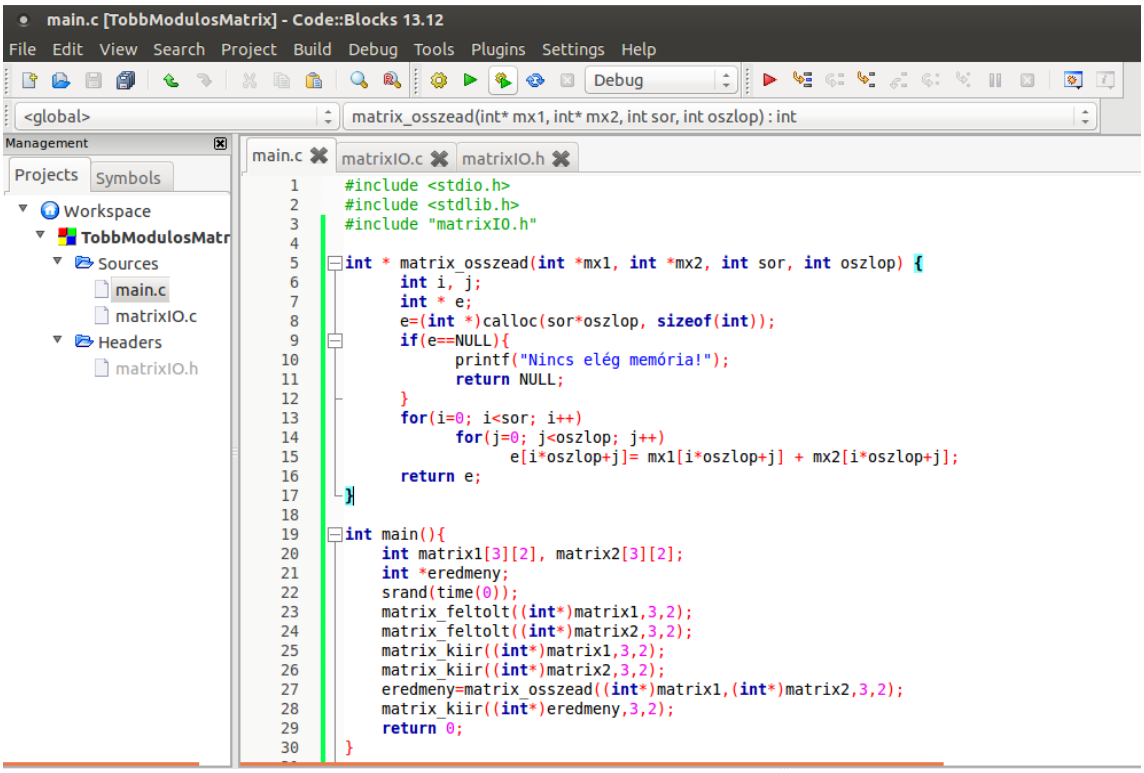
8. ábra: Nyomonkövetés a Watches ablak segítségével

Itt automatikusan megjelennek a lokális változók. Ha nem, egyszerűen a kódból be lehet húzni őket. Ezt követően a felső **Debug toolbar**on a **Next instruction** gombra kattintva lehet a breakpointok között léptetni. A Watches ablakban pedig látható a változók értékének változása.

A nyomonkövetés végeztével a töréspontok a **Debug – Remove all breakpoints** menüpont kiválasztásával törölhetők; vagy egyesével a a piros körökre kattintva.

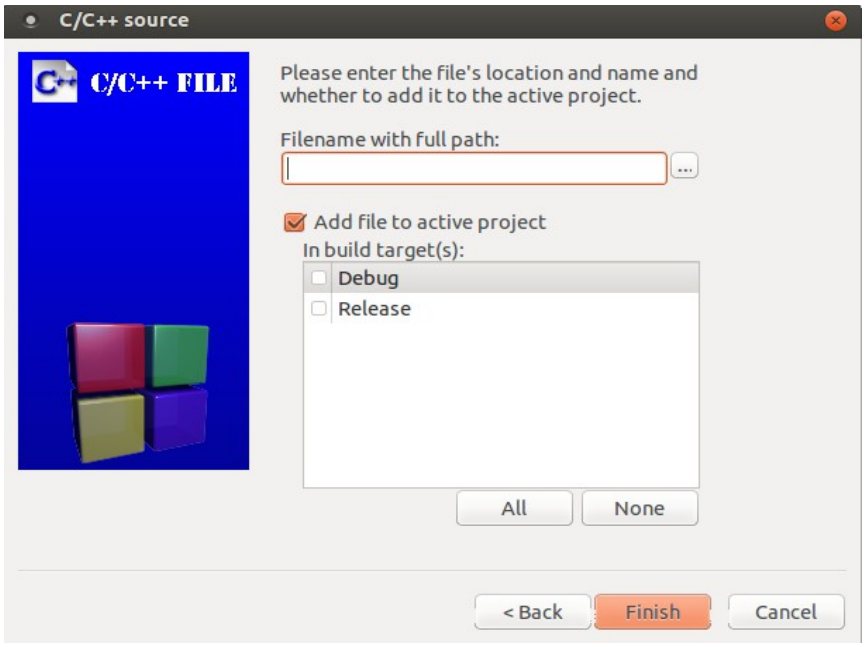
6. Több modulból álló program készítése

A példa programunk két véletlenszámokkal felöltött mátrix összeadását végzi el. Ehhez létrehoztunk egy projektet, amelynek *main.c* modulja a 9. ábrán látható.



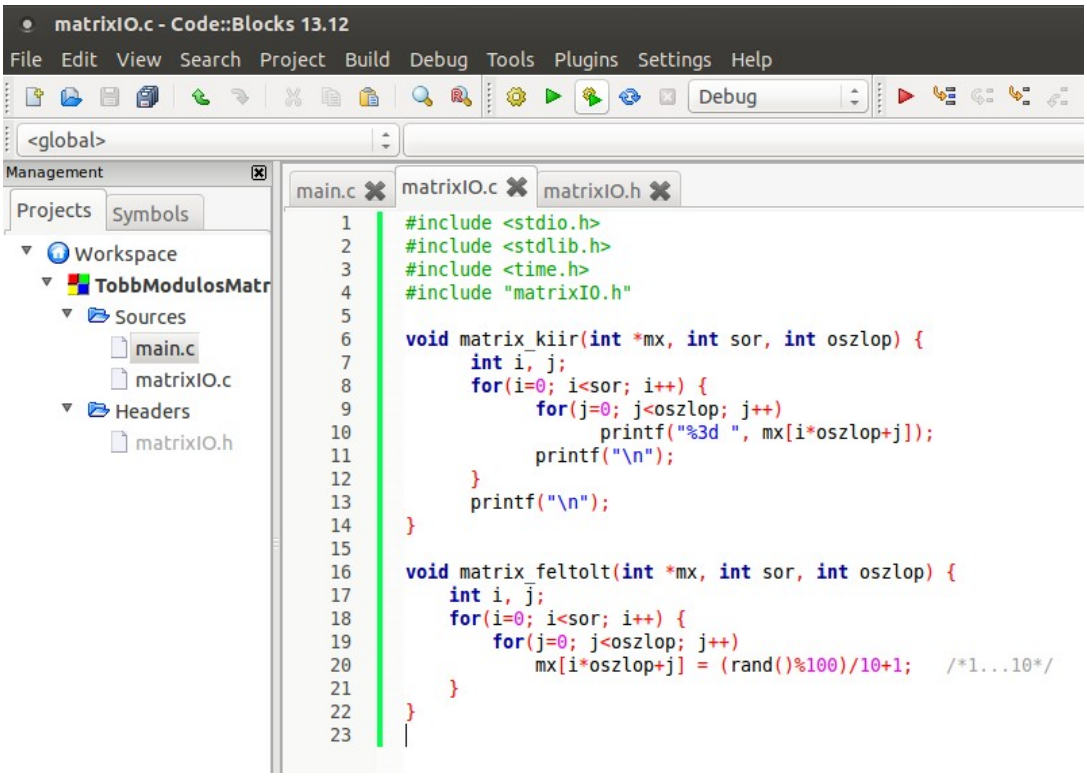
9. ábra: Több modulos program main modulja

Ehhez a projekthez szeretnénk egy újabb forrásfájlt készíteni, ami a mátrixok feltöltését és kiírását hajtja végre. Hozzuk létre a **File – New – File – C/C++ source** menüpontot választva. A fájlnév megadásakor adjuk hozzá a fájlt az aktív projekthez és jelöljük be a *Debug* és *Release* célkatalógusokat.



10. ábra: Új forrásfájl hozzáadása a projekthez

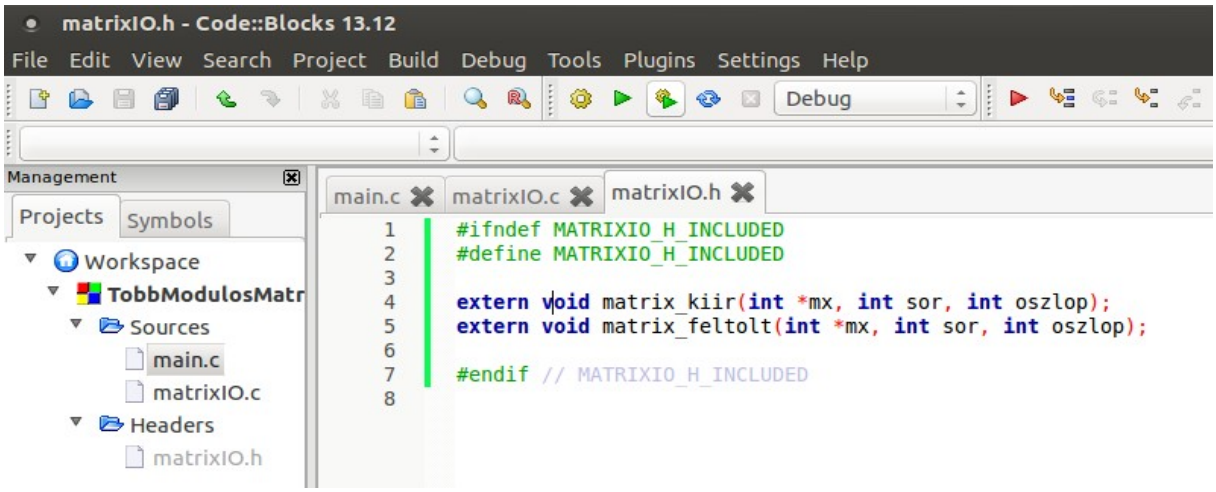
Írjuk meg a forráskódot (függvények definícióját) és építsük be (#include) a forrásfájlhoz tartozó header állományt. A nevük célszerű, ha megegyezik. Saját header állományainkat nem <> zárójelek között, hanem idézőjelek között adjuk meg.



11. ábra: Több modulos program egyik (nem main) modulja

Ezt követően hozzuk létre a header állományt a **File – New – File – C/C++ header** menüpontban (kiterjesztése .h). Adjuk hozzá a projekthez, és jelöljük be a *Debug* és *Release* célkatalógusokat.

A header állomány tartalmazza a hozzá tartozó forrásfájlban szereplő azon változók és függvények deklarációit, amelyekre más modulból szeretnénk hivatkozni. A kezdőérték nélküli változóknál adjuk meg az **extern** (külső) tárolási osztályt; a függvények prototípusánál az extern elhagyható, mert ez az alapértelmezett tárolási osztály.



12. ábra: Több modulos program egyik moduljához tartozó header fájl

A kódszerkesztő automatikusan kiegészíti a header állomány kódját az #ifndef, #define, #endif direktívákkal. Alkalmazásukkal elkerüljük, hogy ugyanazt a modult esetleg többször építsük be a programba, ami hibához vezethet.

Amennyiben létező modult szeretnénk felvenni a projektbe, másoljuk be először a modult a header állománnyal együtt a projekt könyvtárába. Majd adjuk hozzá a fájlokat a projekthez a **Project - Add files** menüpont segítségével. Jelöljük be a *Debug* és *Release* célkatalógusokat. Végezetül építsük be (#include) a forrásfájlhoz tartozó header állományt a *main.c* modulba.

Kódolási javaslatok

1. Programinformációk megjelenítése a forráskódban

A program nyitó sorai megjegyzésként (komment) tartalmazzák a legfontosabb információkat a program készítőjéről, valamint a feladat szöveges leírását és specifikációját (bemenet, kimenet, előfeltétel, utófeltétel, definíció). Ezt célszerű sablonként (template) elmenteni: **File - Save project as template…** Ezt követően minden további programunkhoz felhasználhatjuk ezt a sablont. Sablon betöltése: **File – New – From template...**

2. Programfejlesztés követése

A program fejlesztése során emlékeztető szövegeket helyezhetünk el a forráskódban megjegyzésként: **TODO …** Ide a fejlesztéssel kapcsolatos további teendőinket tudjuk feljegyezni. A **View - To-Do list** menüpontot kiválasztva egy külön ablakban megjelenik a TODO-k felsorolása, amik közül bármelyikre rákattintva a kurzor az adott programpontra kerül.

3. A kód olvashatóságának és karbantarthatóságának biztosítása

- Indentálás:

A forrásfájlok szerkesztése során figyeljük arra, hogy az összetartozó kódrészek vizuálisan is megfoghatók legyenek. A kód olvashatóságát nagymértékben javítja, ha a sorokat indentáljuk (beljebb kezdjük).

- Beszédes változó- és függvénynevek:

Használjuk olyan változó- és függvényneveket, hogy azok szerepe a név elolvasása után azonnal kiderüljön (ne csak a kód értelmezése után).

- Megjegyzések (kommentezés):

Lássuk el a kódot a szükséges megjegyzésekkel, kiegészítő információkkal. Például minden függvény definícióhoz adjuk hozzá megjegyzésként: milyen bemenő adatokat vár, mi a feladata és mi a kimenete.

- Szabványok betartása

A segédletet összeállította: Dr. Baksáné dr. Varga Erika, vargae@iit.uni-miskolc.hu
Miskolci Egyetem Informatikai Intézet, 2014.

Hibakeresés, nyomonkövetés fejezet példája: Maradics Zoltán