Programozás alapjai
1. gyakorlat

Labor használati szabályzat ismertetése → Hallgatói nyilatkozatok aláírása
Felhasználói accountok kiosztása
Bejelentkezés Linux-ra → Szerveren jelszó megváltoztatása → Ellenőrzés: tananyag elérése
Windows regisztráció (winadmin.iit.uni-miskolc.hu)

Ismerkedés a CodeBlocks fejlesztőkörnyezettel
Első C programok

A programkészítés menete CodeBlocks-ban:
1. Indítsa el a CodeBlocks integrált fejlesztőeszközt.
2. Hozzon létre egy új projektet (File – New – Project – ConsoleApplication - C).
3. A Project Manager ablakban (bal oldalon) keresse meg a projekthez tartozó forrásfájlok között (source mappában) a main.c állományt.
4. Szerkesztés és mentés után a Build menüpontban (F9) vagy az ikonsoron: Build and run.
5. Ha sikertelen a fordítási kísérlet (szintaktikai hiba van), piros négyzet jelenik meg a hibás sor elején. Ha nem látszik a Logs ablak (alul), akkor a View menüpontból nyissuk meg. A hibaüzenet (error) szintaktikai hibát (fordítási idejű hibát) jelez, míg a figyelmeztetés (warning) valószínűsíthető futási idejű hibára (szemantikai hibára) hívja fel a figyelmet (de ettől a program lefordul). A hiba javítása után próbáljuk meg újra: Build and run.
6. Sikeres fordítás esetén az eredmény terminál ablakban jelenik meg. Amíg ez az ablak nyitva van, nem lehet új programfutást indítani.
7. Ha a program nem működik helyesen: futás közben “elszáll” vagy hibás eredményt szolgáltat, szemantikai (futási idejű) hiba van. Ennek megkeresése nehezebb feladat. Az integrált fejlesztő eszközökben van hibakeresés funkció (Debug). Ebben az üzemmódban úgy futtatjuk a programot, hogy az minden utasításnál megáll és Watch ablakban (Debug - Windows menüpontból indítható) mutatja a változók értékeit. Így segít a hiba pontos helyének megtalálásában. Ha ez a funkció nem elérhető, akkor a program sarkalatos pontjain a printf függvénnyel kiírathatjuk a kérdéses változók értékét.

Megjegyzés:
A C program több forrásfájlból állhat. Ezeket egyenként hozzá kell adni a projekthez (Project – Add file). A C forrásfájlok kiterjesztése .c. A fájl nevében ne használjunk ékezetes karaktert, space-t és .-ot.

A C program szerkezete
- A C nyelven megírt program egy vagy több forrásfájlban (fordítási egységben, modulban) található, melyek kiterjesztése .c.
- A programhoz csatlakozó header fájlokat az #include utasítással építjük be, melyek kiterjesztése .h.
- A header fájlok az adott forrásfájlon kívül definiált függvények deklarációit tartalmazzák, és konstans makrókat.
- A C program mindig a main függvény első végrehajtható utasításával indul (ez a függvény a program belépési pontja és egy programban csak egy példányban létezhet).

Egy C program felépítése:

```
#-al kezdődő, előfordítónak szóló direktívák (pl. #include)

globális deklarációk és definíciók

saját függvények deklarációja

int main ( ) {
        lokális deklarációk
        utasítások
        return 0;
}

saját függvények definíciója
```

Formázott adatkiírás (output):

printf("formátum", argumentumlista);

Standard függvény, deklarációja az stdio.h standard header állományban található, amit a forrásprogramba az alábbi utasítással illesztünk be: **#include <stdio.h>**

A formátum sztring meghatározza az argumentum értelmezésének a módját, valamint a megjelenítés formáját.

%d – int, %c – char,
%f – float, %lf – double
%s – sztring

1. Indítsunk új projektet. Irassunk ki a képernyőre üdvözlő szöveget.

```
#include <stdio.h>
int main()
{
        printf("\nHelló Világ!\n");     /* sztring konstans */
        return 0;                       //sikeres visszatérés: 0; sikertelen: tetsz.poz.egész szám
}
```

2. Adjuk meg egy téglalap 2 oldalát és számítsuk ki a téglalap kerületét és területét.

```
#include <stdio.h>
int main()
{
        int a = 5, b = 10;
        printf("K = %d \n", 2*a + 2*b);
        printf("T= %d \n", a*b);
        return 0;
}
```

3. Idézzünk elő fordítási hibát. Töröljük az egyik utasítás végéről a pontosvesszőt.
4. Idézzünk elő futási idejű hibát.
   a) Az egyik változónak ne adjunk kezdőértéket.
   b) A %d-t cseréljük %s-re.

Formázott adatbevitel (input):

scanf("formátum", argumentumlista);

Standard függvény, deklarációja az stdio.h standard header állományban található.
A szabványos bemenetről olvas karaktereket, majd a formátumsztring specifikációi szerint értelmezi és konvertálja azokat. A konverzió eredményét a soron következő argumentum által kijelölt memóriaterületre (címre) tölti. Lehetővé teszi, h. a változóknak a program futása során adjunk értéket.

A scanf() függvény működése:

- Space/újsor karakterig olvas.
- Az input adatokat az operációs rendszer az <Enter> billentyű lenyomásakor egy ideiglenes tároló területre (pufferba) tölti. Innen a scanf() fv csak annyi karaktert dolgoz fel, amennyit értelmezni képes a megadott formátum szerint, a többi bennmarad a pufferban.
- A C szabványban nincs definiálva eljárás az input buffer kiürítésére, de egyes Windows verziókban implementált: *fflush(stdin)*.
- Karakter beolvasása: a speciális karakterek is beolvashatók %c formátummal, ezért amikor egy karaktert akarunk beolvasni, át kell ugrani az előző beolvasás után az input bufferben maradt újsor karaktert, azaz *" %c"* (space %c) fog helyesen működni.

Két szám összege:

```
#include <stdio.h>
int main() {
        /*a változókat nem inicializáljuk, futásidőben kapnak értéket*/
        int a;   double b, c;
        printf("\nKérek két számot: ");                  //csak ír!
        /*a program egy int és egy double típusú értéket vár vesszővel elválasztva, amiket a változók
        által kijelölt memória címre tesz */
        scanf("%d, %lf", &a, &b);                       //csak olvas!
        c = a + b;
        printf("A két szám összege: %d + %lf = %.2lf \n", a,b,c);
        return 0;
}
```