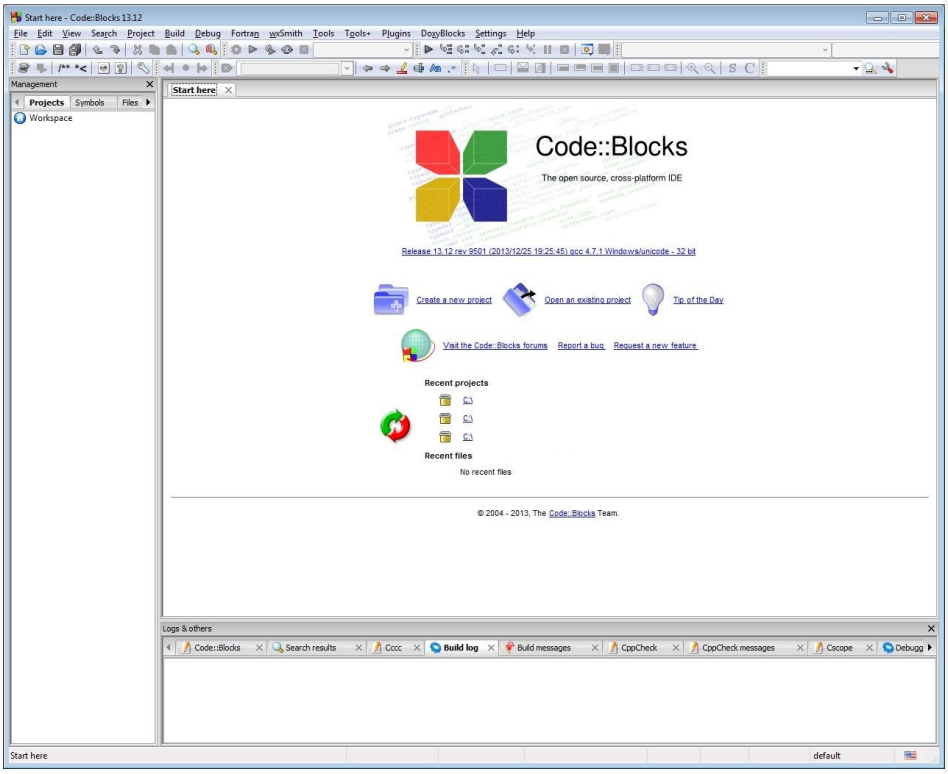
1. hét

Az óra témája

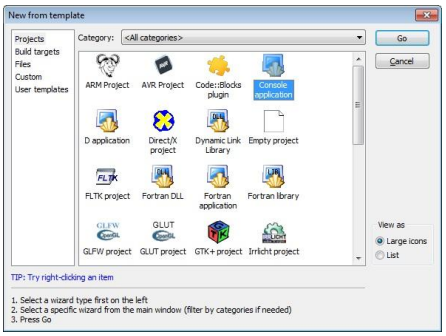
A labor célja, hogy a hallgatók megismerkedjenek a Code::Blocks fejlesztői környezettel, megismerjék hogyan kell új projektet létrehozni és a megírt programjukat futtatni.

A fejlesztői környezet bemutatása, és új projekt létrehozása

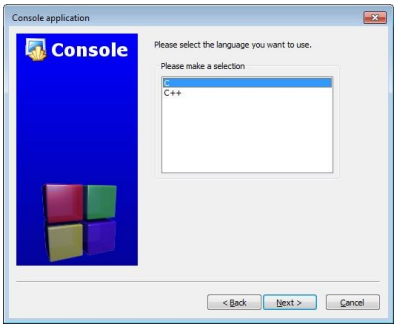
Indításkor a környezet a következőképpen néz ki:



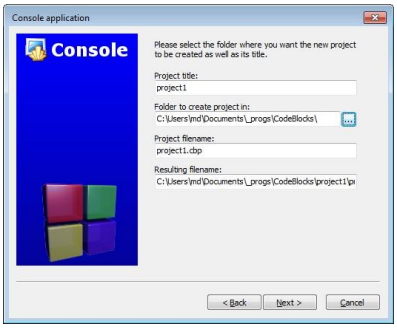
Kattintsunk a középen található „Create a new project” fülre.



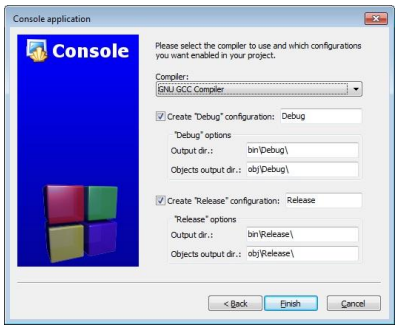
Válasszuk ki a Console application-t majd kattintsunk a Go gombra. Ekkor felugrik egy 'Welcome' ablak, amit a Next gombbal át is ugorhatunk. Ezután kiválaszthatjuk a használni kívánt programozási nyelvet.



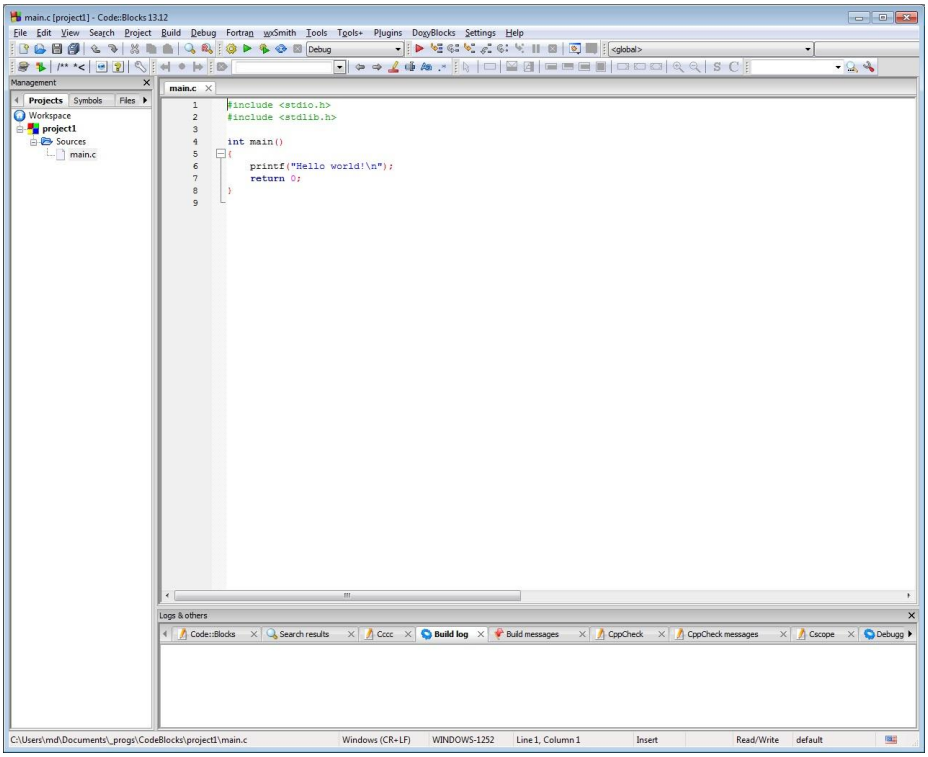
Válasszuk ki a C nyelvet és lépjünk tovább. A következő lépésben megadhatjuk a projektünk nevét és helyét a számítógépen.



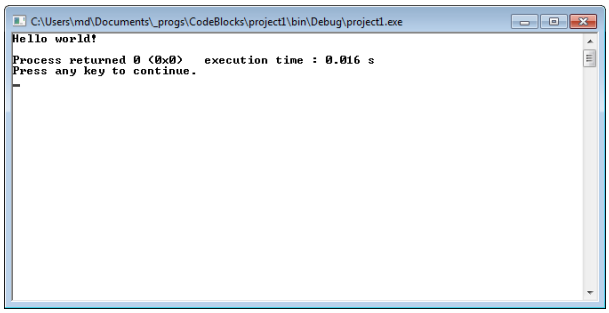
A következő lépésben lehetőségünk van kiválasztani a Compiler-t.



A Finish gombbal létrejön projektünk és látni fogjuk a Workspace-ben. Ha megnyitjuk a 'Sources' mappát megtaláljuk benne a main.c fájlunkat.



Egy programot az F9-es billentyűvel tudunk lefuttatni. A „Hello world" példaprogram kimenete így néz ki:



A felugró terminál ablakban kiírásra került a „Hello world!” felirat. Alatta látjuk, hogy a program nulla visszatérési értékkel tért vissza, ami azt jelenti, hogy a futás során nem történt hiba. Mellette láthatjuk a végrehajtási időt. A következő sorban pedig látjuk, hogy egy gomb lenyomására vár a program. Amíg nem nyomunk le egy billentyűt, addig nem záródik be az ablak. Ha egy workspace-en több projektet hozunk létre, akkor az F9-es gombnyomásra azt fogja futtatni, amelyik projekt aktív. Az aktív projekt neve mindig vastag betűvel látható. Ha egy projektre jobb egérgombbal kattintunk, a legördülő menüben kiválaszthatjuk, hogy a projekt legyen aktív.

Feladat

Írjon példaprogramot, mely a C nyelvben használt változótípusokat mutatja be.

Megoldás

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a, b, c, d;
    unsigned int e;
    float f, g;
    double h;
    long lo = 10;
    long double ld = 0;
    short int ssi;

    a = 1;
    b = 2;
    c = a + b;

    printf( "a: %d b: %d c: %d\n", a, b, c );

    d = 10*9*8*7*6*5*4*3*2*1;
    printf( "10 fakt.: %d\n", d );

    // ----------------------------------------
    printf("\n");
```

```
a = -10;
e = 10;
printf( "%d %u\n", a, e );    // %d - signed integer
                              // %u - unsigned integer

f = 127/15;                   // mivel mind a 127, mind a 15 int,
                              // konstans az osztás eredménye is int lesz
printf( "127/15 = %f\n", f ); // %f - float kiiratása

f = 127/15.0;                 // a 15.0 már float konstans
printf( "127/15.0 = %f\n", f );

f = 127%15;                   // modulo -> maradékképzés
printf( "127/15 maradek: %f\n", f );

a = 10;
b = 3;
f = (float)a/(float)b;        // mindkét operandust float-tá konvertáljuk
                              // a mûvelet elvégzése *elõtt*
g = a/b;                      // ebben a sorban nem konvertálunk
printf( "f = %f\ng = %f\n", f, g );

// ---------------------------------------
printf("\n");

printf( "int: %d, long: %d, float: %d, double %d, long double %d\n\n",
    sizeof( a ), sizeof( lo ), sizeof( f ), sizeof( h ), sizeof( ld ) );
/* Melyik legnagyobb tarhelyet igenylo tipus? */

printf( "Hexadecimalis %d, oktalis %d, decimalis %d\n\n", 0xFF, 077, 99);

printf( "Legnagyobb unsigned int ertek: %u\n", 0xFFFFFFFF );
printf( "Legnagyobb signed int ertek: %d\n", 0x7FFFFFFF );
printf( "Legkisebb signed int ertek: %d\n", 0x80000000 );

printf("\n");
printf("short int merete:\t%d byte\n", sizeof(ssi));
ssi = 0x7FFF;
printf("legkisebb erteke:\t%d\n", ssi );
ssi = 0x8000;
printf("legnagyobb erteke:\t%d \n", ssi );

// ---------------------------------------
printf("\n");

a=2;
b=3;
c=(a++) + (b++);
printf("c=(a++) + (b++) -> %d\n",c);
printf("a:%d b:%d\n",a,b);

a=2;
b=3;
c=(++a) + (++b);
printf("c=(++a) + (++b) -> %d\n",c);
printf("a:%d b:%d\n",a,b);
```

```
    a=2;
    b=3;
    a+=b;   //  a = a + b
    printf("a+=b -> %d\n",a);
    printf("a:%d b:%d\n",a,b);

    a=2;
    b=3;
    b-=a;   // b = b - a
    printf("b-=a -> %d\n",b);
    printf("a:%d b:%d\n",a,b);

    // ---------------------------------------
    printf("\n");

    a = 1; b = 0;
    printf( " a && b = %d\n", a && b );
    a = 1; b = 1;
    printf( " a && b = %d\n", a && b );

    a = 1; b = 0;
    printf( " a || b = %d\n", a || b );
    a = 0; b = 0;
    printf( " a || b = %d\n", a || b );

    return (0);
}
```

Magyarázat

A példaprogram elején látható, hogy tudunk deklarálni változókat, valamint változót közvetlen értékadással deklarálni. Bemutatásra kerültek az `unsigned, long, short` típus módosítók.

A kijelzőre való kiíratáshoz a `printf (const char*, ...)` függvényt használtuk. Ennek ismertetése a jegyzetben megtalálható. A következő táblázatban néhány `printf` ~~specifikátor~~ *specifier* látható, ezekkel adjuk meg, hogy az első sztringparameter után átadott változókat a függvény hogyan értelmezze, hogyan írja ki őket a képernyőre.

formázott: Betűtípus: Dőlt

specifier	**Output**	**Example**
`d` *vagy* `i`	Signed decimal integer	`392`
`u`	Unsigned decimal integer	`7235`
`o`	Unsigned octal	`610`
`x`	Unsigned hexadecimal integer	`7fa`
`X`	Unsigned hexadecimal integer (uppercase)	`7FA`
`f`	Decimal floating point, lowercase	`392.65`
`F`	Decimal floating point, uppercase	`392.65`
`e`	Scientific notation (mantissa/exponent), lowercase	`3.9265e+2`
`E`	Scientific notation (mantissa/exponent), uppercase	`3.9265E+2`
`g`	Use the shortest representation: `%e` or `%f`	`392.65`
`G`	Use the shortest representation: `%E` or `%F`	`392.65`
`a`	Hexadecimal floating point, lowercase	`-0xc.90fep-2`
`A`	Hexadecimal floating point, uppercase	`-0XC.90FEP-2`

c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a `signed int`. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%