**Fejlesztői Dokumentáció**

**Alapinformációk**

**Projekt neve:** Amőba 1.0  
**Fejlesztési nyelv:** C# (.NET 6 vagy újabb)  
**Típus:** Konzolos alkalmazás  
**Fejlesztő:** Szilágyi Zsombor, Szászfai László  
**Fő fájl:** Program.cs  
**Namespace:** Amoba

## A program célja

A program egy **10x10-es amőba játék** konzolos megvalósítása.  
A játék célja, hogy a két játékos (X és O) **egymás után öt azonos jelet** helyezzen el vízszintesen, függőlegesen vagy átlósan a pályán.  
A program támogatja:

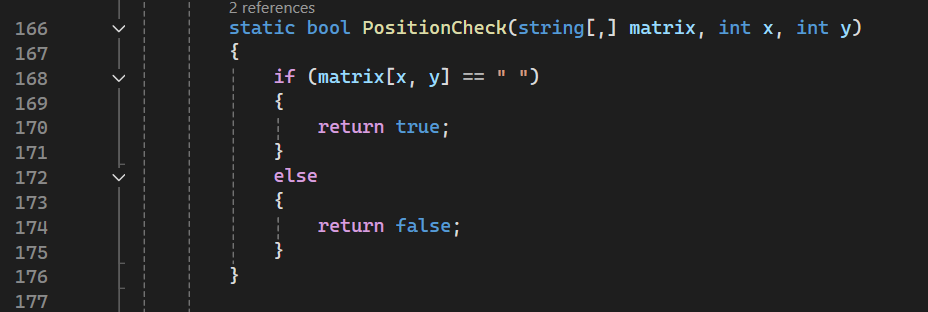
* a billentyűzettel történő vezérlést (nyilak + Enter),
* a kézi pozíciómegadást,
* a győzelem automatikus felismerését,
* egy egyszerű főmenüt.

**Programfelépítés**

**Szilágyi Zsombor program részletei:**

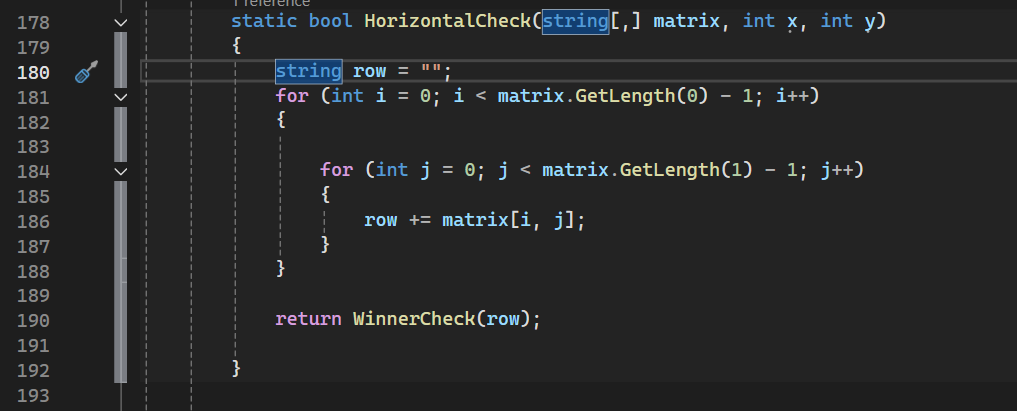
* **PositionCheck()**
* **HorizontalCheck()**
* **VerticalCheck()**
* **DiagonalRightCheck()**
* **DiagonalLeftCheck()**
* **WinnerCheck()**
* **ShowMenu()**
* **Menu()**
* **IsFull()**

**PositonCheck()**

****

Ez egy egyszerű alprogram, ami visszaadja, hogy arra a helyre raktak-e már X-et vagy O-t.

**HorizontalCheck()**



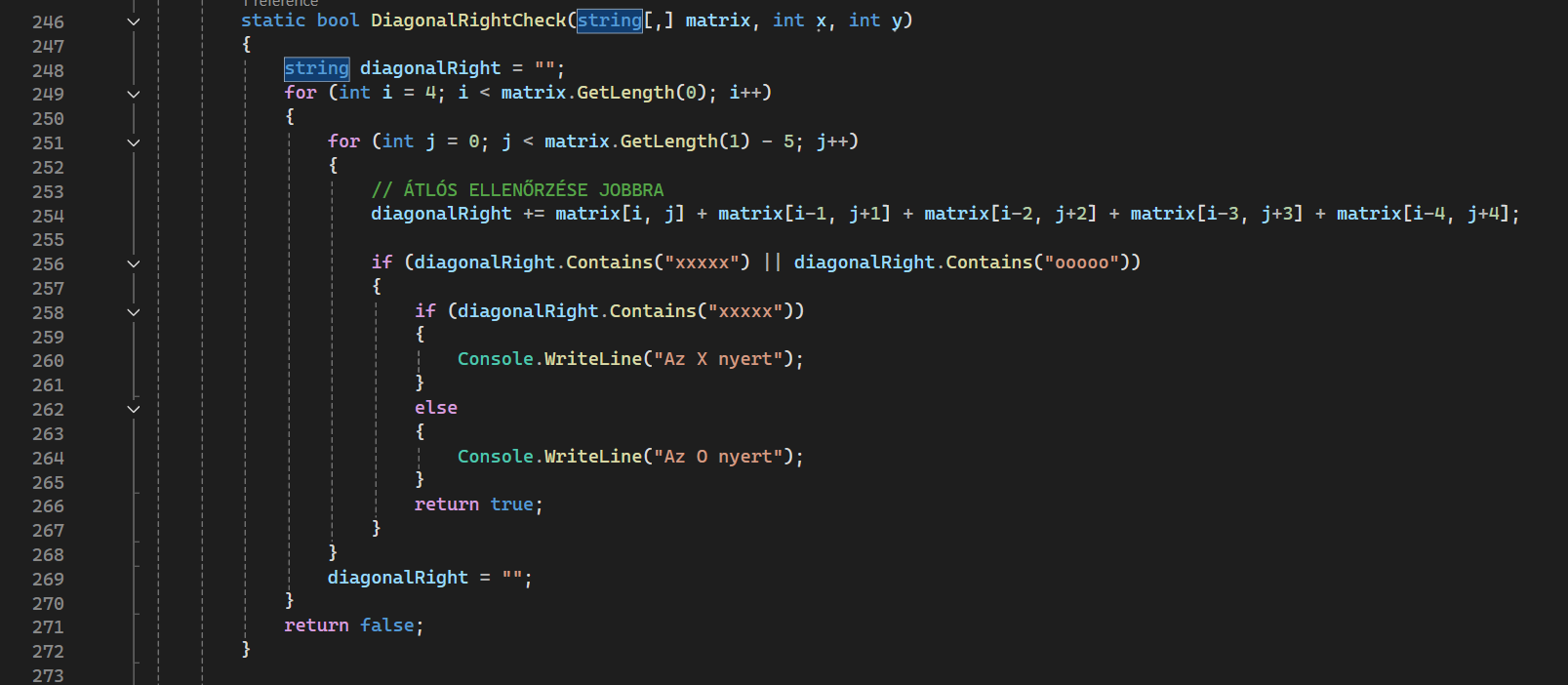
Ez az alprogram két **for** ciklussal végig megy a mátrixon és a felhasznált karaktereket egy string változóban tárolja. Ez az alprogram felhasználja a **WinnerCheck()** alprogramot ami a **return** értéket fogja megadni.

**VerticalCheck()**

****

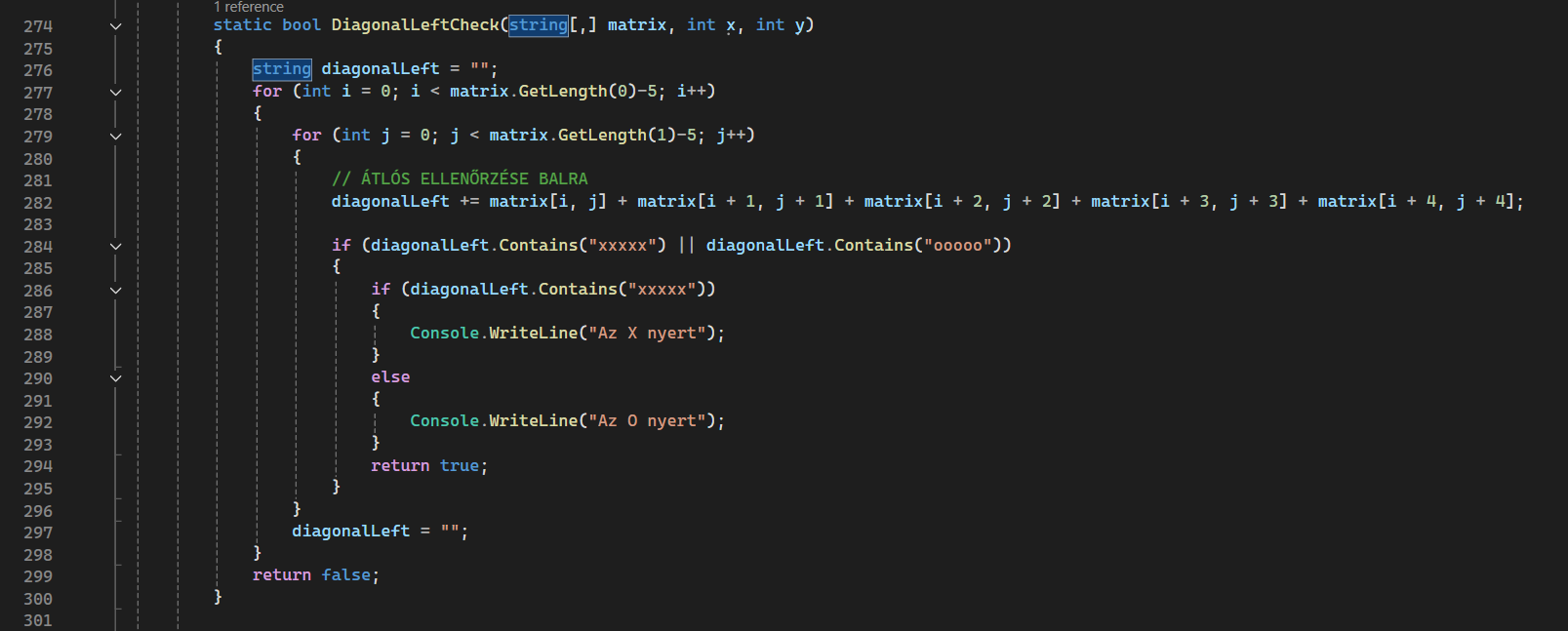
Ez az alprogram egy **string** változóba tárolja el az oszlopokat két **for** ciklus segítségével. Ebben egy másik **WinnerCheck()** alprogram működik (Elrontottuk a refaktorálást ezért két WinnerCheck alprogramot csináltunk *ami működik ahhoz ne nyúlj hozzá* módon).

**DiagonalRightCheck()**

****

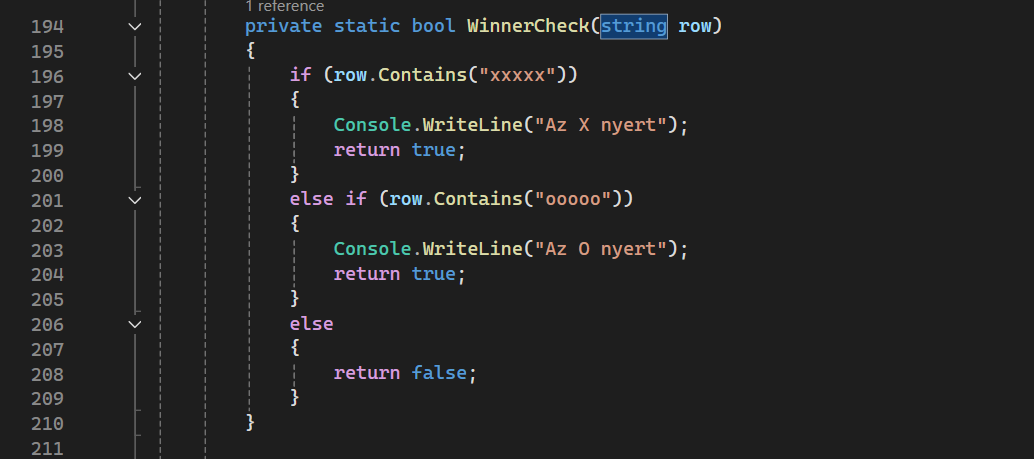
Ez az alprogram felel az alulról felfelé, kereszt irányba történő vizsgálatért. Itt is két **for** ciklussal megyünk végig a mátrixon, viszont itt a **string** változóba úgy adtuk hozzá az elemeket hogy egyszer jobbra egyszer felfelé az összeset. A **.Contains()** függvénnyel pedig megnéztük hogy megtalálható-e benne a „xxxxx” vagy a „ooooo” minta. Ha igen akkor vissza ad egy Mondatot hogy kinyert és egy True értéket ha nyer. Ha nem akkor meg egy False értéket.

**DiagonalLeftCheck()**



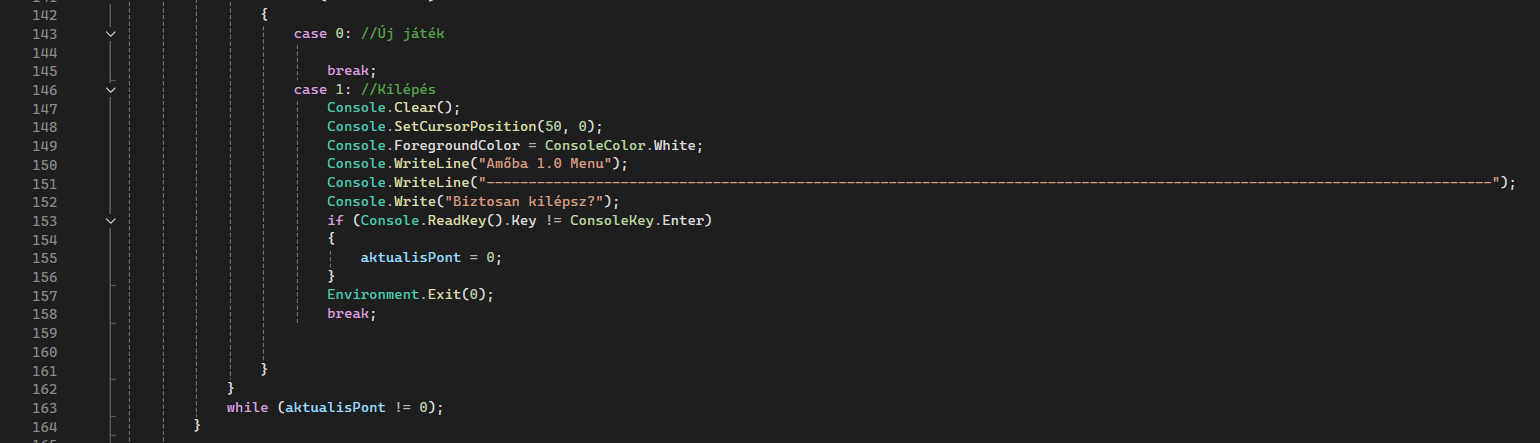
Ez az alprogram rész ugyan azt csinálja mint az előbb említett **DiagonalRightCheck()** függvény, de itt egyet jobbra, egyet le minta alapján adja hozzá a **string** változóhoz az elmeket.

**WinnerCheck()**



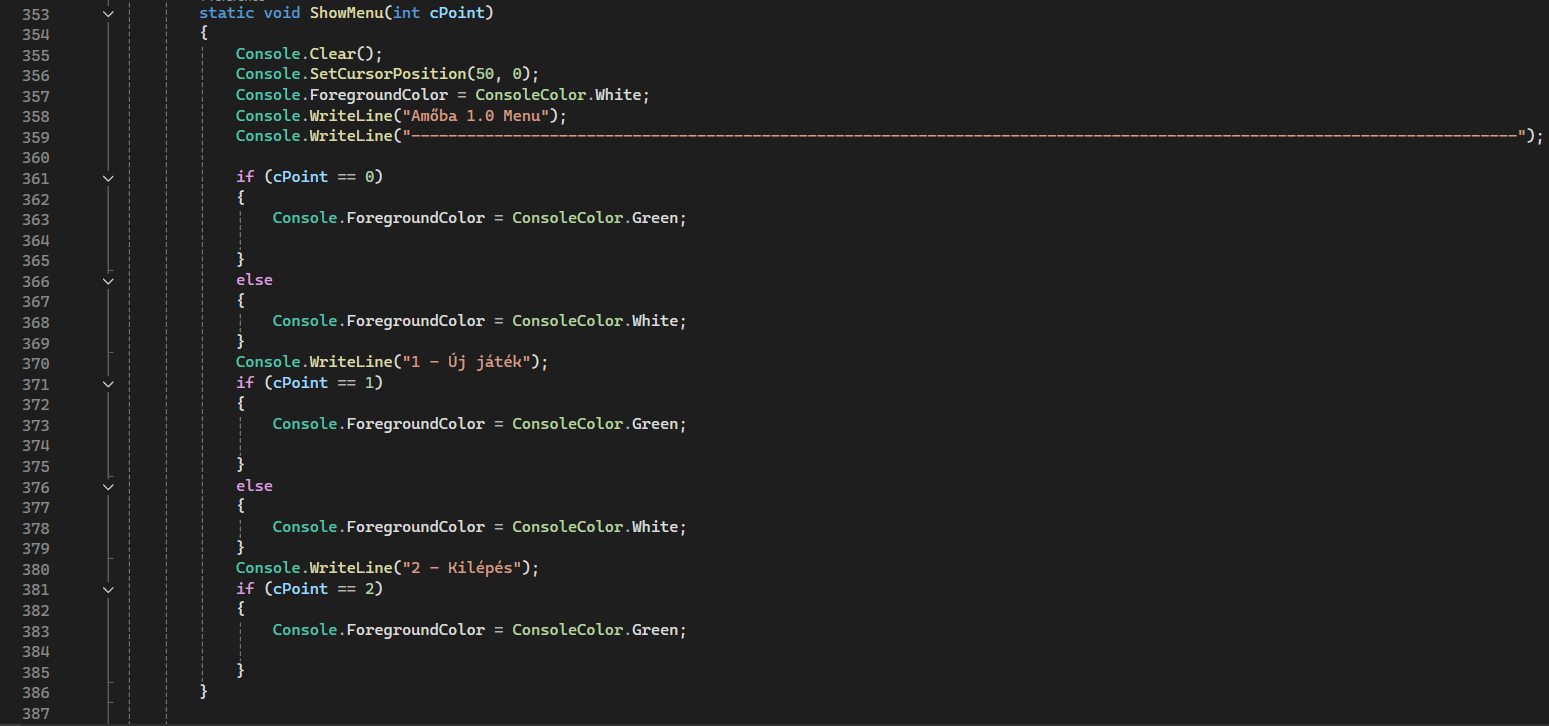
Ez csak egy egyszerű függvény ami vissza ad egy mondatot és egy **True bool** értéket amikor a sorba megtalálja a „xxxxx” vagy a „ooooo” mintát. Alapvetően **False** értéket ad vissza.





Ez az alprogram ami először lefut a programban. ha az „Új Játékra” nyomunk akkor egyszerűen kilép az alprogramból és a **main** következő függvény fut le. Ha viszont a **Kilépés**re nyomunk akkor egy **Environment.Exit(0);** paranccsal kilép az egész programból.

**ShowMenu()**



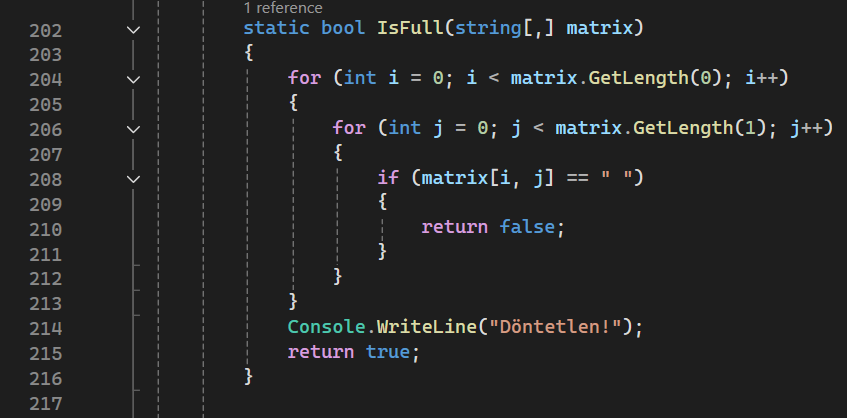
Ez az alprogram felel azért hogy a nyilak hatására változzon a kiválasztott menü pont színe.

Main **while** ciklus:



Ez foglalja össze az ellenőrzéseket és hívja meg őket.

**IsFull()**

****

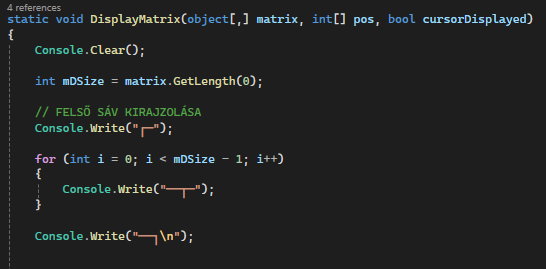
Ez az alprogram ellenőrzi le hogy a tábla betelt-e.

**Szászfai László program részletei:**

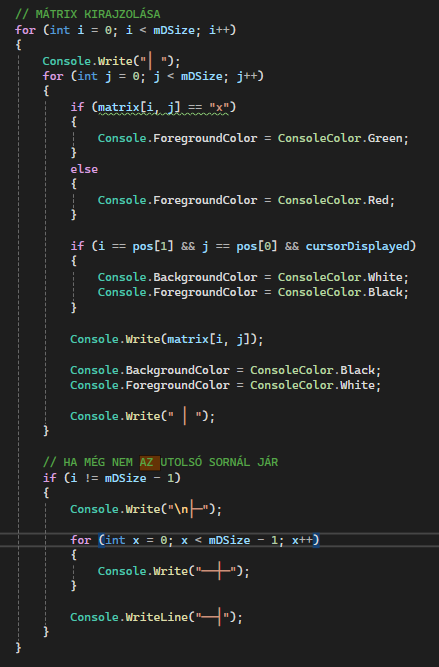
* DisplayMatrix()
* MatrixGenerate()
* MatrixAppend()
* ChangePos()
* ChangeSymbol()
* Main és fő loop

**DisplayMatrix()**

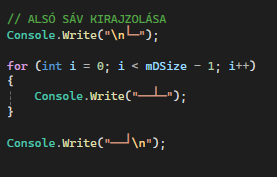
Ez az alprogram felel a játéktér megfelelő kiírásáért



Első feladata a játéktér felső sávjának kirajzolása

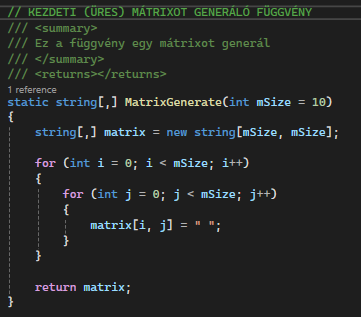


Második a Mátrix “belsejének” ki írása és az elválasztó elemek és a kurzor megjelenítése amíg nem ér az utolsó sorhoz



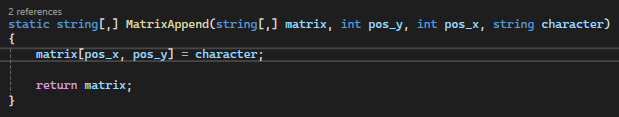
Harmadik az alsó sáv kirajzolása melyet az eddigiektől eltérő karakterekkel valósít meg

**MatrixGenerate()**



A kezdeti mátrixot generáló függvény mely egy space-ekből álló mátrixot ad vissza a megadott méretben

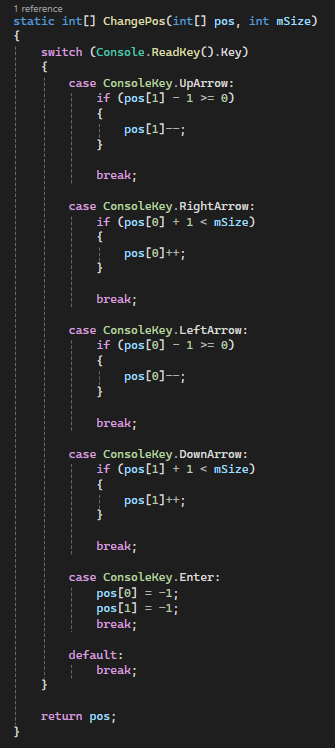
**MatrixAppend()**

****

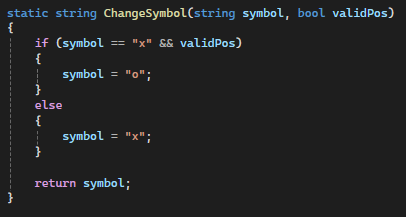
Ez a függvény a megadott helyre a mátrixban elhelyezi a függvénynek paraméterként megadott karaktert majd visszaadja a mátrixot

**ChangePos()**

Ez a függvény a kurzor pozícióját változtatja a billentyűlenyomásnak megfelelően, majd visszaadja az új pozíciót

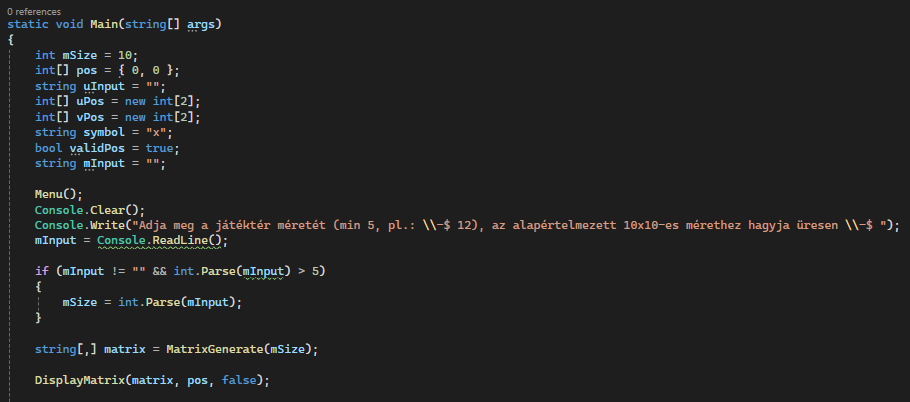
****

**ChangeSymbol()**

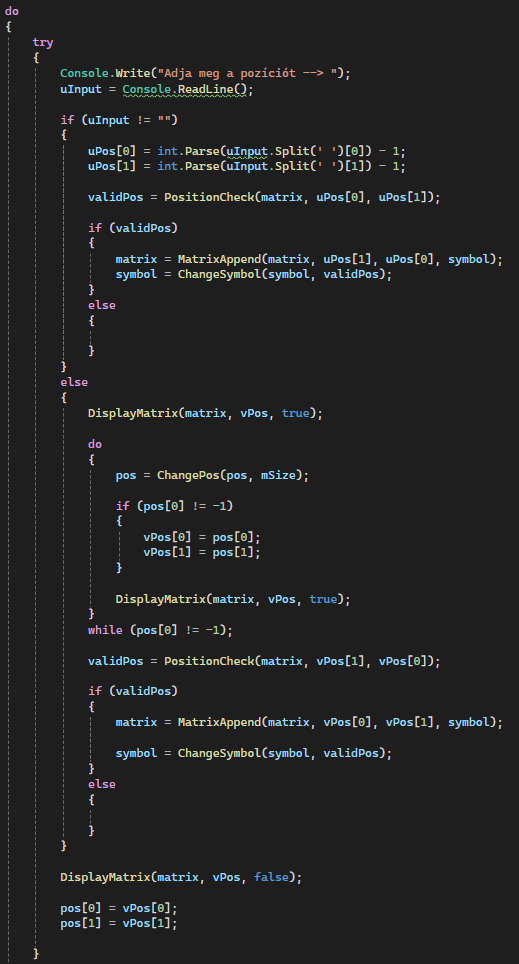


Ez a függvény megváltoztatja a jelenlegi szimbólumot (ha O x-et ad vissza ha X akkor pedig O-t)

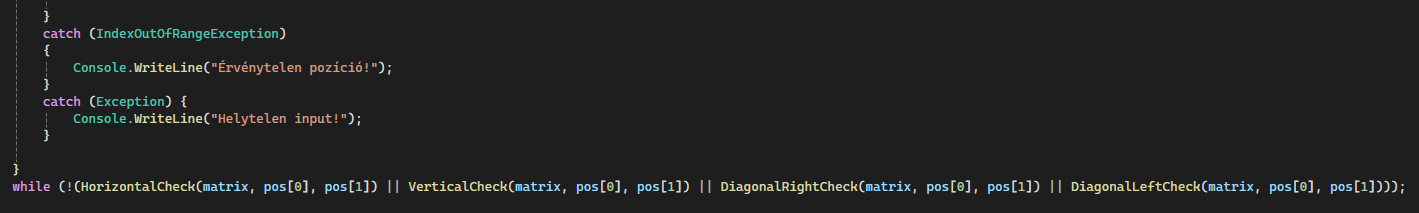
**Main() - és a fő loop**



A fő programrész ezen részének feladata a változók deklarálása majd a menü megjelenítése és a mátrix méretének bekérése majd a mátrixot generáló függvény meghívása



A fő ciklus feladata hogy folyamatosan bekérje a felhasználóktól a pozíciókat és a pozíció megadásának módját (kurzoros vagy koordinátás üzemmód) majd annak megfelelően bővíti a mátrix tartalmát és megjeleníti.



A hibakezelés és a dowhile ciklus while része

**Adattípusok, amit használtunk:**

* Mátrix
* String
* Int
* Bool
* Tömb

Mátrix:

Szimbólumok tárolására 2 dimenziós tömböt/mátrixot használunk a mátrix **string**ként tárolja az adatokat, így a szimbólumokhoz pozícióját megadva könnyedén és gyorsan tudunk hozzáférni.

String:

A felhasználóhoz az adatbekéréshez használtuk és a szimbólumok tárolására. Ezen kívül az ellenőrzéshez is stringet használtunk.

Int:

Mátrix méretnének tárolása és egyéb számlálási folyamatok pl: for ciklus

Bool:

Az ellenőrzésekhez használt alprogramok bool értéket adtak vissza.

Tömb:

Használtunk inteket és stringeket használó tömböket egyaránt.

Stringeket tároló tömb pl:

* felhasználói input felbontása

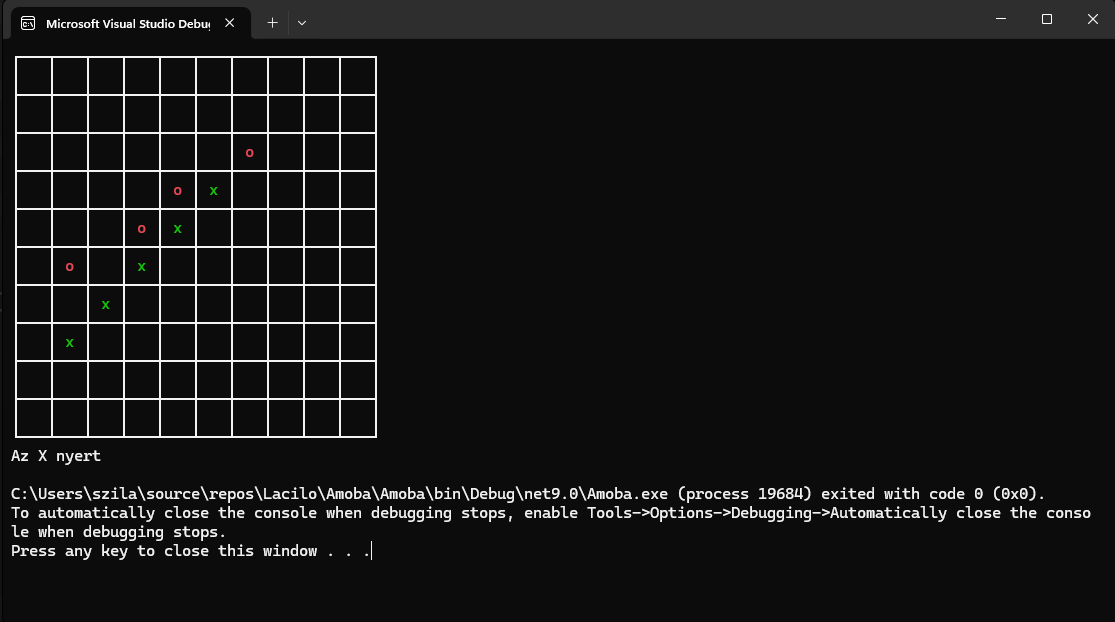
Inteket tároló tömb pl:

* kurzor pozítciója

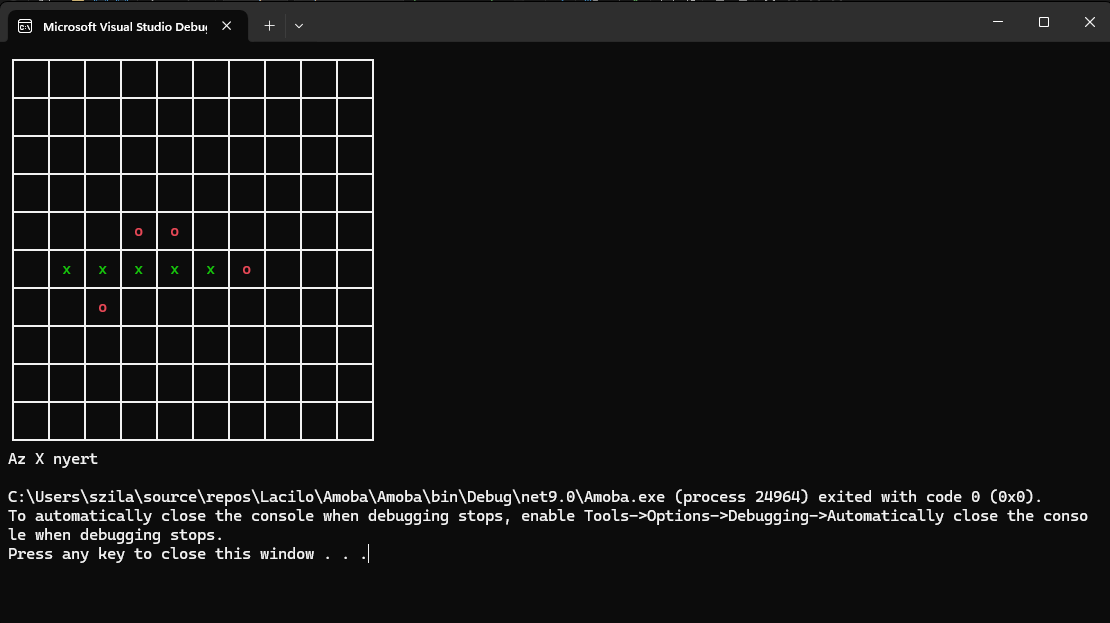
**Tesztek**

**Átlós tesztek:**

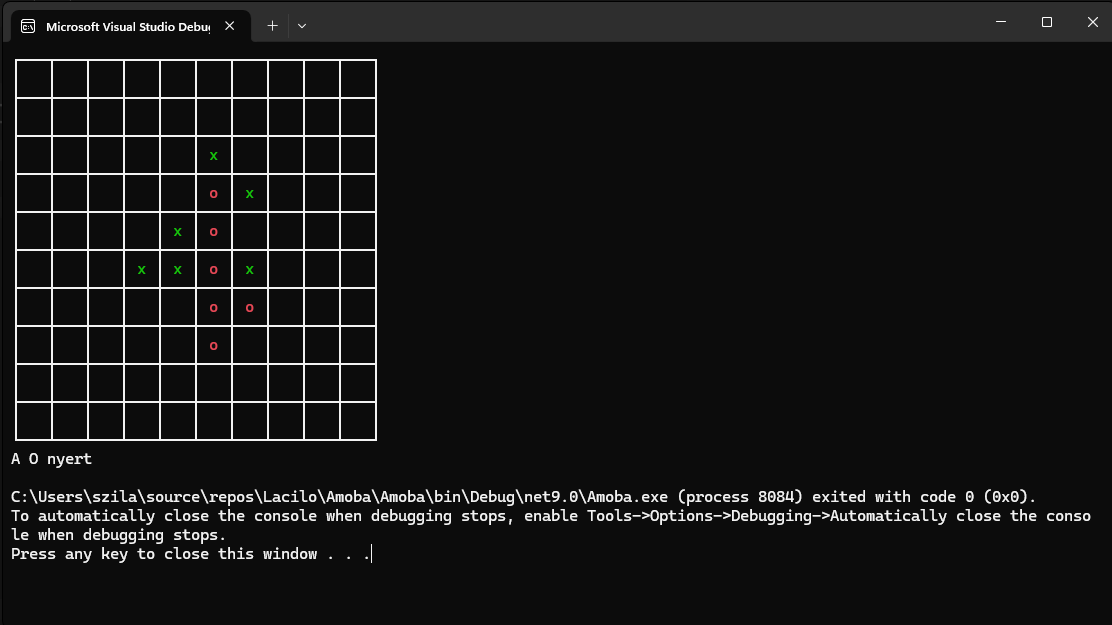




Horizontális



Vertikális



Helytelen input



