NHF programozói dokumentáció

Tóth Mihály Balázs – OAYOF1

A program egy amőbajáték menetét valósítja meg.

A forrásfájlok a *main.c,* a *hatter.c,* a *pvpjatek.c* és a *pvejatek.c.*

A *main* egy keretrendszert biztosít és a főmenü első szintjét tartalmazza. Elsődleges feladata a magasabb szintű függvények meghívása.

A *map* egy *palya* típusú struktúra, felépítése a következő:

- *hely*: egy char típusú duplapointer, a dinamikusan foglalt mátrixra mutat
- *meret*: egy integer, a pálya oldalhosszát tárolja
- *gyozelem*: egy integer, azt tárolja, hogy hány elemet kell elhelyeznünk egy sorban a győzelemhez
- *pvp*: egy logikai változó, azt tárolja, hogy játékosok közti vagy gép elleni játékról van-e szó (ha igaz, akkor játékosok, ha hamis, akkor egy játékos és a gép játszik egymás ellen)
- *soron*: azt tartja számon, hogy ki van éppen soron lépésben (ha 1, akkor player 1, ha 2 akkor player 2)
- *szabad:* egy integer, azt számolja, hogy hány szabad mező van még a pályán
- *prioritás:* integerek mátrixa, 20x20 elemű. A pve játékhoz tárolja az egyes mezők prioritását

A hatter.c függvényei a következők

- *kilep*: nem kap paramétert, visszakérdez a felhasználónak, hogy biztos ki akar-e lépni, ha a menüből a kilépés opciót választotta. Ha igen, akkor igazat ad vissza
- *palyafoglal*: paraméterül kap egy palya struktúrát, annak oldalhossza szerint dinamikusan lefoglal egy pointertömböt(ha nem nulla méretű), aminek elemeit szintén dinamikusan foglalt karaktertömbökre állítja
  A dinamikus területen lévő dinamikus cím értéke kétszeres indexeléssel érhető el
  A függvény kijelzi, hogy sikeres volt-e a memóriafoglalás és visszaadja az új struktúrát
- *palyafree*: felszabadítja a paraméterként kapott pálya által foglalt memóriát
- *ujjatek*: nem kap paramétert, létrehoz egy palya típusú változót, beállítja a struktúra megfelelő adatait a felhasznló által megadott bemenetek alapján, meghívja a *randomkezd* függvényt, majd a *palyafoglal* függvény segítségével lefoglalja a pályának a memóriát és a kész struktúrát visszaadja
- *palyakiir*: paraméterül kap egy palya struktúrát és kiírja a képernyőre az aktuális táblát
- *palyament*: paraméterül kap egy palya struktúrát és elmenti azt egy felhasználó által megadott nevű txt fájlba a dokumentum végén található leírásnak megfelelően
- *palyabetolt*: nem kap paramétert, futása során bekér egy fájlnevet és abból a fájlból betölti egy palya struktúrába az adatokat, ezt adja vissza. Amennyiben sikertelen a beolvasás, megkérdi a felhasználót, hogy újrapróbálja-e. Ha igen akkor bekér egy új fájlnevet és megpróbála beolvasni, ha nem próbálja újra, visszatér a főmenübe.

A pvpjatek.c függvényei a következők:

- *soron_valt*: cím szerint átvesz egy palya struktúrát, híváskor átváltja a struktúra *soron* adatsorát
- *szabad:* cím szerint átvesz egy palya struktúrát és két egész számot, ellenőrzi, hogy az adott pályán a megadott koordináták szerinti lépés érvényes-e, ha igen, iagazat, egyébként hamist ad vissza
- *place_x:* cím szerint átvesz egy palya struktúrát és két egész számot. A *szabad* függvény segítségével ellenőrzi, hogy szabad-e oda karaktert elhelyezni, és ha igen, akkor elhelyez egy X-et és a *soron_valt* függvény segítségével átváltja a soron lévő játékost az ellenfélre
- *place_o:* működése a *place_x*-el azonos, azzal a különbséggel, hogy nem X-et hanem O-t helyez el a pályán
- *playertochar:* cím szerint átvesz egy palya struktúrát, az aktuálisan soron lévő játékos karakterét (X vagy O) adja vissza
- *kelet:* cím szerint átvesz egy palya struktúrát, ezt bejárva megkeresi az első karaktert, amit az aktuálisan soron lévő játékos rakott le, majd ettől a tömbben „keletre" haladva (vízszintesen jobbra) megszámolja, elég azonos karakter van-e már egy sorban a játék megnyeréséhez, ha igen, akkor igaz értékkel tér vissza, ha nem akkor hamissal
- *delkelet:* működése a *kelet*-tel azonos, azzal a különbséggel, hogy „délkeletre" (átlósan jobbra lefelé) keresi az elemeket
- *del:* működése az előbbiekhez hasonlít, ez „délre" (függőlegesen lefelé) keres
- *delnyugat:* működése az előbbiekhez hasonlít, ez „délnyugatra" (átlósan balra lefelé) keres (ezekkel a függvényekkel az összes irányt lefedtük, mivel az egész tömböt bejárjuk vele)
- *randomkezd:* cím szerint átvesz egy palya struktúrát, pszeudorandom módon meghatározza, hogy ki kezdjen
- *jatekkilep:* cím szerint átvesz egy palya struktúrát, megkérdezi a felhasználót, hogy biztosan ki akar-e lépni, amennyiben igen, megkérdezi, hogy szeretné-e elmenetni az aktuális állást: ha igen akkor meghívja a *palyament* függvényt majd visszatér igaz értékkel, ha nem kért mentést a felhasználó, akkor csak visszatér igaz értékkel, és amennyiben mégse akar kilépni a felhasználó akkor igaz értékkel tér vissza
- *beolvas_es_rak:* cím szerint átvesz egy palya struktúrát, beolvas a képernyőről egy stringet, aminek tartalma vagy „K", ezesetben meghívja a *jatekkilep* függvényt, ellenkező esetben kiolvassa a stringből a lépés koordinátáit, átalakítja őket a felhasználó által értelmezhető formáról a program által használt alakra, majd átadja azokat a soron lévő játékosnak megfelelően a *place_o* vagy *place_x* függvénynek
- *nyert:* cím szerint átvesz egy palya struktúrát, meghatározza, hogy az aktuális állásban van-e nyertes, ha igen, akkor igaz értékkel tér vissza, ha nem akkor hamissal
- *betelt:* cím szerint átvesz egy palya struktúrát, meghatározza, hogy betelt-e az összes mező, ha igen, akkor igaz értékkel tér vissza, ha nem akkor hamissal
- *vege:* cím szerint átvesz egy palya struktúrát, eldönti, hogy vége van-e a játéknak, és ha igen akkor hogyan:
    - ha még nincs vége, akkor -1-et ad vissza
    - ha döntetlen (betelt a tábla) akkor 0-t
    - ha player 1 nyert játékos ellen akkor 1-et

    - ha player 1 nyert gép elleni játékan akkor 11-et
    - ha player 2 nyert játékos ellen akkor 2-t
    - ha player 2 nyert gép elleni játékban akkor 12-t
- *pvp:* cím szerint átvesz egy palya struktúrát, a függvény feladata a játékosok között zajló játék lebonyolítása:
  Kiírja az üres pályát, és hogy ki van soron. Meghívja a beolvas_es_rak függvényt, visszatérési értékét egy logikai változóban tárolja, ami játék ciklusának egyik feltétele. A lépéseket a kíírástól a karakter elhelyezésig addig ismétli, amíg be nem telik a pálya, ki nem lép a felhasználó, vagy valaki nem nyer.
  Ha vége a játéknak, a *vege* függvény segítségével kiírja, hogy hogyan lett vége.
- *visszavago:* cím szerint átvesz egy palya struktúrát, meghíváskor (a játék végén) megkérdezi a felhasználót, hogy akar-e visszavágót játszani. Amennyiben igen, felszabadítja a jelenlegi játék pályáját majd újrafoglal egyet azonos paraméterekkel.

A *pvejatek.c* forrásfájl a pve játék lebonyolításáért felel. A *legjobb* struktúrát használja, ennek felépítése a következő:

- *x:* egy egész szám, a meghatározott legjobb lépés x koordinátáját tartalmazza
- *y:* egy egész szám, a meghatározott legjobb lépés y koordinátáját tartalmazza
- *ertek:* egy egész szám, a meghatározott legjobb lépés kiszámított értéket tartalmazza

függvényei a következők:

- *init:* cím szerint átvesz egy palya struktúrát, a *prioritas* mátrix értékeit a következők szerint állítja be: először minden mezőt -1-re állít, aztán a ténylegesen pályán lévő mezők értékét 1-re változtatja, így biztosan elkerülve, hogy esetlegesen egy nem a játéktéren lévő mezőt jelöljön ki a program legjobb lépésként
- *pveszabad:* cím szerint átvesz egy palya struktúrát és két egész számot. Ellenőrzi, hogy szabad-e az adott mező, amennyiben az, igazat ad vissza, egyéb esetben hamist.
- *pvekelet:* működése a *kelet* függvényéhez hasonló; Cím szerint átvesz egy palya struktúrát és egy karaktert, majd a pályát bejárva megkeresi az első karaktert, ami a kapott paraméternek megfelel, majd ettől „keletre” haladva (vízszintesen jobbra) megszámolja hány ilyen karakter van egy sorban. Ezután a sor előtti és a sor utáni mező prioritását megnöveli az alábbi képlet szerint

$$\mathrm{p} = \frac{100}{gy - s}$$

  ahol p a prioritást, gy a gőzelemhez szükséges karakterek számát, s pedig az aktuálisan talált sor hosszát jelöli
  pl gy= 5 esetén p értéke:

sor hossza	p értéke
1	25
2	33
3	50
4	100

- *pvedelkelet, pvedel, pvedelnyugat:* az előbbi függvényhez hasonlóan működnek, irányuk: „délkelet”, azaz jobbra le; „dél”, azaz függőlegesen le; „délnyugat”, azaz balra le
- *elemzo:* cím szerint átvesz egy palya struktúrát, meghívja az *init* függvényt, hogy az előző elemzés elavult adatait kiürítse, majd meghívja a *pvekelet, pvedel,* stb. függvényeket mind O-ra, mind X-re. Mivel ezek a függvények nem foglalkoznak a foglalt mezőkkel, így azokat a pályát bejárva a *pveszabad* függvény segítségével nullázza, majd megkeresi a legnagyobb prioritású mezőt (azonos prioritások esetén az elsőt). Visszatérési értéke egy *legjobb* struktúra
- *pve_beolvas_es_rak:* cím szerint átvesz egy palya struktúrát és egy *legjobb* struktúrát. Ha a számítógép van soron akkor lerakja az előzőleg kiszámított legjobb lépést és visszatér egy hamis értékkel.
  Ellenkező esetben beolvas a képernyőről egy stringet, aminek tartalma vagy „K”, ezesetben meghívja a *jatekkilep* függvényt, ellenkező esetben kiolvassa a stringből a lépés koordinátáit, átalakítja őket a felhasználó által értelmezhető formáról a program által használt alakra, majd átadja azokat a *place_o* függvénynek
- *pve:* cím szerint átvesz egy palya struktúrát, ez a gép elleni játék megvalósításának fő függvénye
  Kiírja az üres pályát, és hogy ki van soron. Meghívja a pve_beolvas_es_rak függvényt, visszatérési értékét egy logikai változóban tárolja, ami játék ciklusának egyik feltétele. A lépéseket a kíírástól a karakter elhelyezésig addig ismétli, amíg be nem telik a pálya, ki nem lép a felhasználó, vagy valaki nem nyer.
  Ha vége a játéknak, a *vege* függvény segítségével kiírja, hogy hogyan lett vége

Mentés után a következő módon van eltárolva egy állás egy txt fájlban:

g vagy j	gép vagy játékos ellen játszunk
p1 vagy p2	ki van soron lépésben (gép elleni játék esetén a játékos p1, a gép p2)
5 és 20 közötti egész szám	tábla oldalhossza
3 és 7 közötti egész szám	győzelemhez szükséges karakterek száma
innen a tábla képe soronként, elválasztó karakter tabulátor	
„X” vagy „O” vagy „ ”	X karakter, O karakter, üres mező
! jelöli a fájl végét	