Programozás alapjai 1.

Amőba

Programozói dokumentáció

**Amőba**

A program egy amőba játékot hoz létre. A felhasználó tud játszani a számítógép ellen. A program létrehozza a játékhoz szükséges vizuális megjelenítést. Mentett játék visszahozatalára is képes fájlból a program újbóli indításakor. Felismeri, ha egyik játékos megnyerte vagy elvesztette a játékot. A program képes a felhasználót végig navigálni a program futása során. A felhasználó tud több stílusú ellenfél ellen játékot folytatni.

**A programban használt enumerációk:**

A program két helyen használ enumerációt:

* A környezet headerben található range tömbben használt Irany enumeráció, itt az ellenorzes függvényhez szükséges irányokat határozza meg.
* A csomopont headerben található a Kimenetel enumeráció itt a megjelenites függvényhez szükséges játék vég fajták vannak felsorolva.

**A programhoz felhasznált külső könyvtárak:**

* econio:  
  Konzolos megjelenítéshez szükséges [konzol kiürítése(econio:clrscr()), konzolos kiíratás során egy adott fix pontra ugrás és ott kiíratás (econio\_gotxy()), színes kiíratás konzolban (econio\_textcolor()), színes betű háttér konzolban (econio\_textbackground())].
* debugmalloc:  
  Dinamikus memóriafoglalás helyes működését ellenőrző segéd könyvtár. Amennyiben memória szivárgás lép fel a programban automatikus jelzi ezt a program futása végén. Amennyiben nem lép fel memóriaszivárgás a program végén ezt is jelzi és jelzi a program futása során memóriafoglalás összegét.

**Program függvényei:**

**int main()**  
 A main függvényben meghívásra kerül a menu függvény és a konzol kódolását átállítja úgy, hogy UTF-8 -s karaktereket is jól letudjon kezelni a program.

**void menu()**  
 A menu függvény segítségével tudja felhasználó kiválasztani, hogy hogyan szeretné használni a programot, új játékot kezd, mentett játékot folytat, megnézi a program útmutatóját vagy kilép a programból. Új játék esetén az adatbe, mentett játék esetén a beol, útmutató estén az utmutato függvény hívódik meg. Amennyiben a felhasználó a kilépés mellett dönt hagyjuk, hogy a program kilépjen a függvényből és befejeződjön, semmilyen más esetben nem ér véget a program.

**void adatbe()**  
 Az adatbe függvény bekéri a felhasználótól a játékhoz szükséges adatokat, mint például név, pálya mérete és a nehézségi szint. Pálya méret tudatában a foglal függvény segítségével lefoglaljuk a játékhoz szükséges memóriaterületet. A csomopont függvénnyel futatni kezdjük a játékot. A csomopont függvény visszatérési értéke szabja meg, hogy induljon-e új játék vagy sem és lépjen vissza a felhasználó a menübe.

**int \*\*foglal(int p\_meret)**  
 A foglal függvény a pályaméret megadásával lefoglalja a memória területet a játékhoz szükséges pályának. A foglal a teljes lefoglalt tömb elemeinek értékét kinullázza, véletlenül se maradjon memóriaszemét bennük. Visszatérési értéke a kétdimenziós foglalt tömb pointere. A memory\_management headerben található.

**void felszab(int \*\*palya,int meret)**  
 A felszabadításért felelős. A játék végével felszabadítja pálya számára foglalt memória területet. A memory\_management headerben található.

**bool csomopont(int \*\*palya, int meret,char \*nev,int level)**  
 A játék közvetlen futásáért felelős függvény. A függvény központi eleme addig fut ameddig a játékot valamelyik felhasználó meg nem nyeri. Előszöris program fix lépése megtörténik, mely a pálya közepén helyezkedik el. A megjelenites függvény segítségével megjelenik az aktuális állapotnak megfelelő pályakép a konzolon. Majd elindul a központi elem, a felhasználótól bekérjük a következő lépését, mely vagy kilep vagy egy betű és szám kombináció lehet, máskülönben hibás lépésnek számít. Kilépés esetén kiir függvény segítségével elmentjük az aktuális játék állapotot fájlba, majd megszakítjuk a központi elemet. A felhasználótól bekérjük, hogy akar-e új játékot kezdeni a uj\_jatek\_szandek függvény segítségével. A visszatérési értékével tér vissza a csomopont függvény. Ha lépésnek tűnő kombinációt ír be felhasználó akkor különböző szempontok szerint ellenőrizzük, hogy léphet-e oda a játékos, ha minden szempontnak megfelel akkor elhelyeződik a lépés. Amennyiben valamelyik feltételnek nem felel meg hibás lépésnek számítjuk és ha eléri az egymás utáni hibás lépések a korlátot megszakítjuk a játékot és a programot hirdetjük ki győztesnek. Helyes lépés esetén elhelyeződik a lépés a pályán, majd ellenorzes függvény kerül meghívásra. Ha ellenorzes függvény igaz értéket ad vissza megjelenites függvény meghívásra kerül, majd uj\_jatek\_szandek és majd a felszab. Hamis érték esetén a program következik, attól függően, hogy felhasználó melyik nehézségi szintet választotta egy logika headersben található függvények közül meghívódik (konnyu/kozep/nehez). Majd az ellenorzo függvény és a visszakapott értékétől függően, igaz értéknél megjelenites, uj\_jatek\_szandek majd felszab függvények, hamis esetén a függvény központi része újra indul az elejétől. Minden egyes lépés követően csökkentjük a pályán maradt üres helyek számontartására létrehozott változót, amikor nem marad üres hely a pályán véget vettünk a játéknak. A függvény központi ciklusa akkor ér véget, ha „kilep” utasítást használja a felhasználó, ellenorzo függvény igaz értékkel tér vissza, betelik a pálya vagy hibás lépések száma eléri a maximumot. A csomopont függvény visszatérési értéke a uj\_jatek\_szandek függvény és „kilep” utasítástól függ, „kilep” utasításnál alapból hamis értékkel térünk vissza, uj\_jatek\_szandek függvényben a felhasználón múlik, hogy milyen utasítást ad meg.

**bool ellenorzes(int meret, int \*\*palya, int oszlop, int sor)**  
 Az ellenorzes függvény ellenőrzi, hogy a játékot valamely felhasználó megnyerte-e. A függvény oszlop és sor változókba megkapja az utolsó lépés pontos helyét. A kornyezet headerben található range tömb felhasználásával bejárja az utolsó lépés környezetét, ahol lehetséges, hogy győzelem keletkezhetett. Mindig a range-ből származó „koordináta kiegészítések” szablyák a kiindulási helyet és range-ből származó enumerációk pedig az ellenőrzés irányát. Kiindulási helyből elindul a függvény a bejárási irányba, amikor a kiinduló karaktertől eltérő karakterhez ér, üres helyre ér vagy pályán kívülre mutató helyre ér megszakad a kis ellenőrző rész. Ha egymásután négyszer nem szakad meg kis ellenőrző rész akkor igaz értékkel visszatérhet a függvény, mert akkor nyert az adott játékos. A függvény amennyiben nem tér vissza igaz értékkel egyszer sem hamis értéket kap legvégén és ezzel is tér vissza. A függvény érték változtatással megkülönbözteti azokat a helyeket, ahol a győzelem történt.

**void kiir(char \*nev,int meret,int szint,int \*\*palya)**  
 A kiir függvény a paraméterként kapott adatokat kiírja fájlba, ezzel menti a játékot. Az információkat a „jatek.txt” fájlba írja ki. Kiírást követően meghívja a felszab függvényt.

**void beol()**  
 A beol függvény a mentett játék vissza hozatalára szolgál. Miután a függvény megismeri a mentett játék pályaméretét foglal függvény segítségével foglaltat neki memória területet és majd ebbe fogja beolvasni a későbbiekben a játék mentett állapotát. Majd a csomopont visszatérési értékétől függően (melynek fő funkciója igazából a mentett játék futtatása) elindít egy adatbe függvényt.

**void megjelenites(char \*nev,int meret,int level,int \*\*palya, int veg)**  
 A megjelenites függvény a pálya és az információk megjelenítéséért felelős. A pálya tömb elemei értéke alapján jeleníti meg a konzolon az aktuális állást. A függvény használja az econio könyvtárat. Amennyiben a paraméterként átvett veg változó semmi állapotban van az azt jelenti, hogy a játék még fut és nem ért véget, ha bármi más állapotban van akkor a függvény kiírja, hogy hogyan ért véget a játék.

**void konnyu(int meret,int \*\*palya,int \*lepet)**  
 A konnyu függvény a paraméterként kapott pályán megnézi, mennyi üres hely található, majd ”random” generál számot, majd a random generált számmal kiválasztunk egy üres helyet maradékos osztással. A lépést a kiválasztott helyre helyezük el. A lépést mentjük a paraméterként kapott tömb pointerbe, hogy lehessen ellenőrizni az utolsó lépés alapján.

**void kozep(int meret,int \*\*palya,int \*lepet)**  
 A kozep függvény a paraméterként megkapott pályán az adott elem mellé tud lerakni elemet. Keress egy olyan helyet, ahol az adott mellett üres hely található. Ha talál elhelyezi ott a lépést. A paraméterként kapott tömbbe eltárolja a lépést, így ellenőrizhető lesz az utolsó lépés alapján. Amennyiben nem tudott lépést tenni akkor meghívja a konnyu függvényt.

**void nehez(int meret, int \*\*palya,int \*lepet)**  
 A nehez függvény a paraméterként kapott pályán helyezi el a lépését. A lépés kiválasztásához létrehoz egy „tükör” tömböt, melynek értékeket ad az alap tömb állásához mérten. A kornyezet headerben található range tömb segítségével. A függvény végigmegy a tömbön és ha talál egy elemet melynek értéke egyik játékoshoz tartozó értéknek megfeleltethető a környezetébe értékeket ad a tükör tömbnek. A függvény ellenőrzi, hogy az alap tömb környezetében értéket érdemlő helyek még a pályán helyezkednek-e el. Amennyiben igen értékeket kapnak, értékek ahhoz mérten vannak megszabva, hogy mennyi pozícióból jöhet ki nyerő állás. Értékadás mindig hozzáadás és nem érték megadás, így, ha több irányból is ideális lépésnek számít a lépés nagyobb értéket kap. Miután különböző kombinációk mind megadták az értékeket a tükör tömbnek megkeresi a legnagyobb értékű elemét. Erre a helyre elhelyezi a lépést. A paraméterként kapott tömbbe (lepet) eltárolja a lépést, így majd lehet ellenőrizni az utolsó lépés alapján. Majd a függvény felszabadítja a tükör tömböt.

**bool uj\_jatek\_szandek()**  
 Az uj\_jatek\_szandek függvény felelős azért, hogy a felhasználótól bekérje, hogy akar-e új játékot kezdeni vagy sem. Ha a felhasználó „igen” parancsot ad be akkor a függvény igaz értékkel tér vissza, amennyiben nem választ adja be a felhasználó a függvény hamis értékkel tér vissza, ha nem megfelelő utasítást gépel be a felhasználó a függvény újra lefut ameddig nem megfelelő utasítást ad be a felhasználó. A függvény az econio könyvtárat használja a többszöri lefutás során a konzol kiürítéshez.

**void utmutato()**  
 Az utmutato függvény arra szolgál, hogy tudassa felhasználóval a program használatát. Utasításokat melyeket tud használni és program helyes működését garantálják. Az utmutato függvény az econio könyvtárat használja a kép teljes törléséhez, helytelen bemenet esetén. A felhasználó a vissza parancsol tud visszalépni, amennyiben nem helyes parancsot ad be az útmutató újra töltődik. Az útmutató szövegét a szabalyok.txt fájlból olvassa be a program.

**int range[25][4]**  
 Ebben a tömbben vannak eltárolva koordináta kiegészítések, logika headerben található nehez függvényhez szükséges érték információk és az ellenőrzéshez szükséges enumerált értékek. Minden sor a tömbben egy összetartozó rész, első oszlopban található az oszlop koordináta kiegészítés, a második oszlopban a sor koordináta kiegészítések, harmadik oszlopban, hogy milyen messzire van az adott koordináta kiegészítésekkel a központi elemtől a pont egy mátrixban, a negyedik oszlopban található, hogy a koordináta kiegészítésekkel milyen irányba található a központi elem.

**A program által használt szöveges fájlok:**

szabalyok.txt:  
 Ebben a txt fájlban találhatóak az Utmutato menüpontban kiírt információk.

jatek.txt:  
 Ebben a txt fájlban van eltárolva a mentett játék. Első információ a játékhoz szükséges pályaméret, második a felhasználó neve, harmadik a program ügyességi szintje és utána következik a mentés állapotának megfelelő pályakép.

**A program által használt nem txt fájlok:**econio.c  
 Ez a c fájl tartalmazza az econio külső könyvtárat.

* adat\_kezeles.c
  + Ez a c fájl tartalmazza az adatbe, kiir, beol, foglal, felszab függvényeket.
* csomopont.c
  + Ez a c fájl tartalmazza a csomopont, ellenorzes függvényeket és a range tömböt.
* logika.c
  + Ez a c fájl tartalmazza a konnyu, kozep, nehez függvényeket.
* seged\_fuggvenyek.c
  + Ez a c fájl tartalmazza a megjelenites, utmutato és uj\_jandek\_szandek függvényeket.
* econio.c
  + Ez a c fájl tartalmazza az econio külső könyvtárat.
* adat\_kezeles.h
  + Ez a c fájl tartalmazza az adatbe, kiir, beol, foglal, felszab függvények meghatározását.
* csomopont.h
  + Ez a c fájl tartalmazza a csomopont, ellenorzes függvények meghatározását, Kimenetel és Irany enumerációt és a range tömb meghatározását.
* debugmalloc.h
  + Ez a header fájl tartalmazza az debugmalloc külső könyvtár meghatározását és a debugmalloc külső könyvtárat.
* econio.h
  + Ez a header fájl tartalmazza az econio külső könyvtár meghatározását.
* logika.h
  + Ez a header fájl tartalmazza a konnyu, kozep, nehez függvények meghatározását.
* seged\_fuggvenyek.h
  + Ez a header fájl tartalmazza az uj\_jatek\_szandek, utmutato és megjelenites függvények meghatározását.
* main.c
  + Ez a c fájl a main függvényt tartalmazza.