Programozói dokumentáció

Adatszerkezetek választása:

A felhasználó által elhelyezett töltéseket tárolni kell, amelyhez adatszerkezet szükséges. A töltések száma bármennyi lehet, tehát dinamikus adatszerkezetet kell használni. A töltések egy láncolt listában tárolhatók. A láncolt lista könnyen bővíthető új töltéssel, és könnyen ki is lehet venni belőle elemeket.

A fájlból beolvasott elemeket el kell tárolni, hogy az új eredményt a megfelelő helyre tudjuk elhelyezni a fájlban. Ha már létezik egy adott felhasználóhoz eredmény, akkor hozzá kell csatolni az új eredményt. Ezt legkönnyebben egy láncolt listával tudjuk megtenni, aminek az elemei tárolják az egyes felhasználóneveket és a hozzájuk tartozó lövésszámokat. Minden kiíratásnál rendezetten írjuk ki az elemeket, így az újabb beolvasásnál mindig rendezetten kapjuk az adatokat.

A felhasználó neve fix méretű, ezért nem használunk a tárolására dinamikus tömböt.

A program működését vezérlő függvények:

void str_beolvas(char str[ ])

Beolvassa a felhasználónevet. A függvény paramétere a beolvasott név karaktertömbje. Karakterenként olvassa be a felhasználónevet. Ha a beírt felhasználónév túl hosszú vagy nem megfelelő karaktert tartalmaz, akkor olyan tömbbel tér vissza melynek az első eleme 0. Ha a beolvasott sztring első eleme a 0, akkor hibát jelez a főprogram.

A menüt vezérlő függvények:

int menu(char* felhasznalo)

Ez a függvény kezeli a menüt. Paraméterként kap egy sztringet, ami a felhasználónevet tárolja. Inicializálja az ablak méretét, az SDL ablakot, a menüpontokat és a lövésszámokat. A menüpontokat egy adott méretű tömbben tárolja. Kétféle fontot inicializál, egyiket a menüpontokba írt szövegekhez használja, másikat az utasítások kiírásához. Ezután kirajzolja a háttérképet és a menüpontokat. Egy ciklusban eseményt vár a felhasználótól. Switch-case szerkezet kezeli a választott eseményt. A fel-le nyilakkal változik a menüpont tömb indexe. Enter esetén újabb switch-case szerkezetben dönti el a program, hogy mit kell csinálnia, attól függően mi a menüpont tömb indexe. Ha kilép a ciklusból, bezárja a két fontot és az SDL ablakot.

void jatek(char* felh, int *loves, int meret, Ablak ablak, SDL_Surface *screen)

A függvény paraméterként kapja a felhasználónevet tartalmazó sztringet, a lövések tömbjét, a tömb méretét, az ablak adatait és az SDL ablakra mutató pointert. Lenullázza a lövésszámokat és meghívja a hozzájuk tartozó pálya függvényeket. Ha bármelyik lövésszámra 0-át kap, akkor visszalép a menübe.

SDL_Surface *screen_init(Ablak ablak)

A függvény paraméterként kapja az ablak méretét, majd inicializálja az SDL ablakot. Visszatérési értéke az inicializált SDL ablak.

void eredmeny_kiir(SDL_Surface *screen, Ablak ablak, TTF_Font *utas)

Ez a függvény kiírja az eddigi 10 legjobb eredményt. Paraméterként kapja az SDL ablakra mutató pointert, az ablak méretét, és az utasításhoz tartozó fontot. Inicializál egy fontot az eredmények kiírásához és egy képet a háttérképnek, majd kirajzolja a háttérképet. Megnyitja a fájlt, ha nem létezik a fájl, akkor kiírja, hogy nincsenek még eredmények. Ha létezik a fájl, akkor beolvassa egy láncolt listába a fájlban található adatokat. Kiírja a fejlécet és az utasítást, majd kilistázza az eredményeket egy ciklussal. Vár egy entert vagy bezárást, bezárja a fontot és a fájlt, majd felszabadítja a listát.

void fajlba_ir(char* user, int loves[], int meret)

Ez a függvény kiírja fájlba az elért eredményt. Paraméterként kapja a felhasználónevet tartalmazó sztringet, a lövések tömbjét és annak méretét. Ezután ha sikerült megnyitni a fájlt, akkor bemásolja egy láncolt listába. Végigiterál a lista elemein és összehasonlítja a felhasználóneveket a jelenlegi felhasználó nevével. Ha talál megfelelőt, akkor kifűzi a listából és felülírja az eredményt. Ha nem talál, akkor foglal neki egy új lista elemet. Ha eredetileg üres volt a lista, akkor a lista csak ezt az elemet tartalmazza. Ha nem volt üres, akkor befűzi a helyére az elemet. Ezután kiírja egy fájlba a megváltozott listát.

fajl* fajl_beolvas(FILE *fp)

Ez a függvény egy paraméterként kapott fájlból beolvassa az adatokat egy láncolt listába. A visszatérési értéke a létrehozott láncolt lista.

void fajl_felszabadit(fajl *f)

A függvény a paraméterként kapott láncolt listát felszabadítja.

int rosszabb(fajl* mi, fajl *minel)

A függvény paraméterként kap két listára mutató pointert, és logikai 1-el tér vissza, ha az első paraméterként kapott listaelemben tárolt eredmény rosszabb mint a második paraméterként kapott elem által tárolt.

A pályákat vezérlő függvények:

int palya1(char* felh, SDL_Surface *screen, Ablak ablak)

Ez a függvény paraméterként kapja a felhasználónevet tartalmazó sztringet, az SDL ablakra mutató pointert és az ablak méreteit. A függvény megadja a pályára jellemző adatokat, majd meghívja a palyakezel függvényt. Visszatérési értéke a lövésszám.

int palyakezel(char* felh, SDL_Surface *screen, Ablak ablak, Pont *fix, int fixdb, akadaly *akad, int akaddb, Pont cel, int palyaszam)

A függvény paraméterként kapja a felhasználónevet, az SDL ablakra mutató pointert, a fix töltéseket tartalmazó tömböt és annak méretét, az akadályok tömbjét és annak méretét, a cél adatait és a pálya sorszámát. A függvény megadja a felhasználható töltések adatait, a pálya szélén lévő sávoknak és a kilövő ágyúnak a méretét. Lenullázza a felhasználható töltésekhez tartozó click adattagot. Meghívja a palya_kirajzol függvényt, majd eseményt vár. Ha a felhasználó lenyomja a bal egérgombot, akkor megnézi, hogy a kurzor felhasználható töltésen van e. Ha igen, akkor annak a töltésnek a click adattagját 1-re változtatja, bemásolja a temp változóba az adatait, majd visszaállítja 0-ra a click adattagot. Ha jobb egérgombbal kattint a felhasználó, akkor megnézi hogy felhasznált töltésen van e a kurzor. Ha igen, akkor törli azt a töltést a listából. Ha felengedi a bal egérgombot a felhasználó és a temp click adattagja 1, tehát van kijelölve töltés, akkor megnézi, hogy a pályán belül helyezkedik e el a kurzor. Ha a pályán belül van és nincs másik töltésen vagy a célon, akkor belefűzi a felhasznalt listába a töltést. Ha a felhasználó mozgatja az egeret és a temp változó click adattagja 1, akkor letörli az előző helyről a töltést azzal, hogy újrarajzolja a pályát. Átadja a kurzor koordinátáit a temp változónak majd újrarajzolja a temp töltést. Ha felfele billentyűt nyom a felhasználó, akkor növeli az alpha szöget fél fokkal 90 fokig. Lefele billentyűleütésre csökkenti a szöget 90 fokig. 90 és -90 foknál megáll az ágyú, nem megy ki a pályáról. Space esetén a lövések számát növeli egyel, megadja a mozgó töltés első koordinátáit és kiszámítja az x és az y irányú sebességet. Amíg nem lép ki a felhasználó, vagy nem ütközik a mozgó töltés másik töltéssel, fallal, vagy akadállyal, addig meghívja a lovesek függvényt.

toltesek* torles(toltesek *l, toltesek *mit)

A függvény paraméterként kap egy láncolt listát, és egy a listában lévő elemre mutató pointert. A függvény kitörli a listából a paraméterként kapott elemet.

void felszabadit(toltesek *l)

A függvény felszabadítja a paraméterként kapott láncolt listát.

toltesek* uj_felhasznalt_elem(toltesek* l, Pont mit)

A paraméterként kapott láncolt listához hozzáfűzi a második paraméterként kapott elemet.

int nincs_ilyen(Pont temp, Pont cel, Pont fix[], int fixdb, toltesek *f)

A függvény paraméterként kapja a temp ideiglenes töltés változót, a cél adatait, a fix töltések tömbjét és méretét, a felhasznált töltések listáját. Logikai 1-el tér vissza, ha nem ütközik a temp töltés koordinátája egyik töltéssel sem és nincs rajta a célon.

A játékot vezérlő függvények:

int lovesek(Pont* toltes, Pont f[], toltesek *l, int meret, double *v_x, double *v_y, SDL_Surface *kep)

A függvény paraméterként kapja a mozgó töltés koordinátáit, a fix töltések tömbjét és méretét, a felhasználható töltések listáját, az x és y irányú sebességekre mutató pointereket és az SDL ablakra mutató pointert. Fehér színnel kirajzolja a mozgó töltést előző állapotát, növeli az x és y irányú koordinátáit vx és vy-nal. Kiszámítja a pályán lévő töltések által a mozgó töltésre kifejtett erő nagyságát, és ellenőrzi, hogy nem áll e körpályára a töltés, vagy nem zuhan e bele egy töltésbe. Ha a mozgó töltés és a fix töltés közti távolság kevesebb, mint 0,1-el változott, akkor meghívja a rontas függvényt. Kiszámítja a mozgó töltés új koordinátáit, majd kirajzolja azt.

void rontas(Pont *toltes, Pont f)

Az első paraméterként kapott töltés x koordinátáját úgy változtatja meg a függvény, hogy a második paraméterként kapott töltéshez közelebb legyen. Ha az két paraméterként kapott töltés közti távolság kisebb, mint 15 és negatív a fix töltés (második paraméter) polaritása, akkor zuhanjon bele a fix töltésbe, tehát egyezzenek meg a koordináták.

void palya_kirajzol(Palya* palya, Ablak abl, double szog, Pont cel, const char* user, toltesek *l, Pont haszn[], Pont f[], int fmeret, int hasznmeret, akadaly *akad, int akdb, int lov, int palyaszam, SDL_Surface *scr)

Ez a függvény rajzolja ki a pályát, amit a felhasználó is lát. Paraméterként kapja a pálya adatait, az ablak méretét, a kilövő szögét, a cél adatait, a felhasználónevet, a felhasznált töltések listáját, a fix töltések tömbjét és méretét, a használható töltések tömbjét és méretét, az akadályok tömbjét és méretét, a pálya sorszámát és az SDL ablakra mutató pointert. Belemásolja egy sztringbe a lövésszámot és a pályaszámot. Inicializál egy fontot a kiírandó szövegekhez és egy képet a háttérképhez. Kirajzolja a háttérképet, majd megváltoztatja a forrásterületet, mert legközelebb csak a kép bal sávjára lesz szükség. Kirajzolja a pályhoz tartozó elemeket, és újrarajzolja a bal szélső sávra a

háttérképet, hogy elfedje az ágyú felesleges részeit. Megváltoztatja a forrásterületet, legközelebb csak a kép felső sávja kell. Kirajzolja a használható töltéseket és kiírjuk alájuk azok méretét. Kirajzolja a felhasznált töltéseket és újrarajzolja a felső sávra is a képet. Megrajzolja a pályát elválasztó vonalakat és kiírja a felhasználónevet, a pályaszámot és a lövésszámot.

void cel_kirajzol(double x, double y, double r, SDL_Surface *kep)

Paraméterként kapja a cél koordinátáit és az SDL ablakra mutató pointert. A függvény kirajzolja a célt.

void kilovo_kirajzol(Palya* p, Ablak ablak, double a, SDL_Surface *kep)

Paraméterként kapja a pálya adatait, az ablak méretét és az SDL ablakra mutató pointert. A függvény kiszámolja a kilövő téglalap részéhez tartozó koordinátákat, majd kirajzolja a kilövőt.

double pontok_tavolsaga(Pont egyik, Pont masik)

A függvény kiszámolja a két paraméterként kapott pont közötti távolságot és visszatér a kiszámolt értékkel.

void erokiszamitas(double *v_x, double *v_y, int polaritas, double r, double x, double y, double tx, double ty, double tav)

Ez a függvény számolja ki azt az erőt, amivel egy töltés eltéríti a mozgó töltést. Paraméterként kapja az x és az y irányú sebességekre mutató pointereket, a fix töltés adatait, a mozgó töltés koordinátáit és a két töltés közti távolságot. A függvény 6 esetet különböztet meg aszerint, hogy milyen a két töltés helyzete. Ha a fix töltést vesszük origónak, akkor attól függően, hogy a mozgó töltés melyik sík negyedben van más lesz a beta szög. Emellett, ha a két töltés y koordinátája megegyezik, akkor az atan nem értelmezett, ezért külön kell megadni ott a szög értékét. A függvény az erő x és y irányú komponensének a nagyságával növeli a sebesség x és y irányú komponensét.