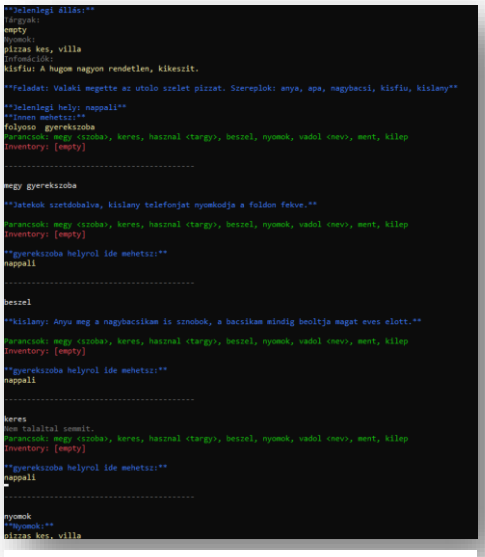
Szöveges kalandjáték programozói dokumentáció



A program kinézete

A program célja:

A program egy tetszőleges számú és összeköttetésű helyszínből álló nyomozós kalandjáték, melyben a megszerzett nyomok, tárgyak és információk segítségével rá kell jönnünk, ki a tettes, mindezt színes szövegeket használó konzolos felületen.

Használt függvények és struktúrák modulok szerint csoportosítva:

game.c/game.h (a játék működésének függvényei)

- typedef struct Lista{

  char *adat;

  struct Lista *kov;

  }Lista;

Dinamikusan sztringeket tároló láncolt lista

- typedef struct Allapot{

  char *nev, *hely, *jatekmod, ***palya, ***story, *tettes;

  int helyek_szama, szovegek_szama;

  bool eredmeny;

  Lista *inventory, *nyom, *info;

  }Allapot;

A játék lényeges adatait összefoglaló struktúra. Előnye: egyben adhatjuk át a függvényeknek az adatokat, könnyen kezelhető. Sztringek dinamikusan foglaltak, a pályát és a „storyt” háromszoros(!) indirekcióval adjuk meg, mivel 2D dinamikus sztring tömbök. A helyek és szövegek számát praktikus tárolni, ciklusoknál könnyebbség. Az eredmény egyedül a játék sikerességét vizsgálja. Továbbá láncolt listák találhatók még itt

- void jatek(Allapot *allapot, bool *vege, char *parancsok);

  A megfelelő játékmechanikai parancsokat hívja meg, és loopolja a játékot.

- void start(bool *vege, Allapot *allapot);

  Elindítja a játékot. Átveszi a vege logikai változóra mutató pointert, amit megváltoztat sikeres esetben, valamint belső függvényei betöltik az adatokat az allapotba.

- void uj_jatek(bool *helyes, Allapot *allapot);

  Új játékot indít, melyben választanunk kell játékmódot és mentésnevet. Feltölti az allapotot az üres értékekre. Sikeres esetben a helyes pointert igazra állítja.

- void betolt (bool *helyes, Allapot *allapot);

JÁTÉKMECHANIKA

Betölt egy korábban választott játékállást, amit a felsorolásból választottunk. Ha nincs ilyen, átirányít az uj_jatek függvényre. A mentett állás adatait betölti az allapotba, a helyest igazra állítja.

- void megy(Allapot *allapot);

  Helyváltoztatás, ha a jelenlegi hely és a választott cél összeköttetésben van.

- void keres(Allapot *allapot);

  Ha a feltételek teljesülnek, az adott helyen inventory/nyom listába rakja a talált dolgokat.

- void hasznal(Allapot *allapot);

  Ha lehetséges, használja a választott inventoryban lévő tárgyat, melynek hatására új tárgyakat/információkat kaphatunk.

- void beszel(Allapot *allapot);

  Ha a feltételek teljesülnek kapunk az adott helyen lévő szereplőtől információt, ha azzal még nem rendelkezünk.

- void nyomok(Allapot *allapot);

  Kilistázza az eddig megszerzett nyomokat, információkat.

- void vadol(bool *vege, Allapot *allapot);
- A játék végcélja. Ha jó embert gyanúsít meg, a játék véget ér. Egyébként csak void ment(Allapot *allapot);
  Menti a játékállást.
- void kilep(bool *vege, Allapot *allapot);

  Felajánl egy mentést, majd kilép a játékból.

datahandle.c/datahandle.h (adatok, listák, bemenetek kezelése)

- void beolvas_string(char *str);

  Kihagy egy sort majd beolvas egy sztringet a billentyűzetről.

- void beolvas_enterig(char *str);

  Kihagy egy sort, majd beolvas egy sort a billentyűzetről.

- void csonkit(char *szo, int levagando);

  Paraméterként megadott sztringből levág paramétreként megadott számú karaktert.

- bool bemenet_ellenorzes(char *bemenet, Lista *elvart);

Ellenőrzi, hogy a bemenetként kapott érték megtalálható-e a paraméterként adott listában, majd visszatér ennek logikai értékével.

- void listabatesz(char *mit, Lista **lista);

  Hozzáfűz egy paraméterként megadott sztringet a paraméterként adott láncolt listához.

- void listakiir(Lista *lista, Formatum forma);

  Kiírja a konzolra a lista elemeit paraméterként megadott formátumban.

- void felszabadit_lista(Lista *lista);

  Felszabadít egy láncolt listát.

- void felszabadit_allapot(Allapot *allapot);

  Felszabadítja az allapot változó minden dinamikusan foglalt elemét.

- void hova_mehet(Allapot *allapot);

  Kilistázza, hogy az adott helyről hova mehet tovább a játékos.

- void info_feltetel(Allapot *allapot, int i, char* feltetel);

  Berakja a feltetel sztringbe a story megfelelő sorának elemeit megfelelő formátumban, hogy lehessen ellenőrizni, hogy teljesül-e.

filehandle.c/filehandle.h (fájlok kezelése)

- bool has_txt_extension(char const *name);

  Ellenőrzi, hogy az adott fájl txt kiterjesztésű-e.

- void mentesek_listaz(Lista **lista);

  Kiírja a korábbi mentéseket a konzolra.

- void jatek_betolt(Allapot *allapot);

  Betölti a pálya és story 2D dinamikus tömbökbe az adatokat.

- void mentes_betolt(char *fajlnev, Allapot *allapot);

  Betölti az allapotba a kívánt játékállás adatait.

- void mentes_letrehoz(Allapot *allapot);

  Létrehoz egy új mentést, amelybe beleírja az allapot adatait. Korábbi mentés felülírására is ezt lehet használni.

display.c/display.h (megjelenítés)

- typedef enum Formatum{

  sima,

  story,

  inventory,

  commands,

  listae,

  helytelen

  }Formatum;

Felsortolt lista, melynek elemei a printf_formazott megfelelő kiírási módját tudják megadni.

- void printf_formazott(Formatum formatum, char *str);

  Különböző formátumú kiíratások. Elérők benne a színek, szüveghatárok.

  <u>Fejlesztési lehetőség:</u> Amennyiben a játék grafikus formába átültetésre kerülne, ebbe a modulba jönnének annak függvényei.

<u>Fájlkezelés:</u>

- gamefiles mappa
  - gm1 mappa
  - gm2 mappa
  - gm3 mappa
  - saves mappa

helyszinek.txt -> szomszédságmátrix

tortenet.txt -> attribútumos mátrix

tortenet.txt felépítése paraméteresen:

```
tettes
start   NULL    NULL    szoveg  NULL    NULL    NULL
megy    hely    NULL    szoveg  NULL    NULL    NULL
beszel  hely    NULL    szoveg  ki_mondja       NULL    NULL
beszel  hely    info_sora       szoveg  ki_mondja       NULL    info
keres   hely    NULL    szoveg  nyom    nyomba_kerül    NULL
keres   hely    NULL    szoveg  inventory       inventoryba_kerül       NULL
keres   hely    info_sora       szoveg  inventory/nyom  inventoryba/nyomba_kerül        info
hasznal hely    eszkoz  szoveg  szerzett_inventoryba    inventory       NULL
hasznal hely    eszkoz  szoveg  ki_mondja       info    NULL
```

Az első sorba kerül a tettes, utána pedig a felsorolt elemek közül tetszőleges számú. Fontos azonban, hogy a sorok között komoly összefüggések lehetnek, illetve a valami megszerzéséhez kötött sorokat hierarchikusan kell felépíteni, hogy a már korábban kiírásra kerültek ne ismétlődjenek. Első oszlop a parancs, második a játékos helyes, harmadik az

információhoz kötött soroknál a FELTÉTEL információ sorának SZÁMA. Az info, nyom és inventory beírandó, és nem helyettesítendő adatok! A többi értelemszerűen.

Szomszédságmátrix példa:

```
honnan/hova     folyoso konyha  furdoszoba      haloszoba       dolgozoszoba    nappali gyerekszoba     udvar
folyoso 0       1       1       1       1       1       0       1
konyha  1       0       0       0       0       0       0
furdoszoba      1       0       0       0       0       0       0       0
haloszoba       1       0       0       0       1       0       0       0
dolgozoszoba    1       0       0       1       0       0       0       0
nappali 1       0       0       0       0       0       1       0
gyerekszoba     0       0       0       0       0       1       0       0
udvar   1       0       0       0       0       0       0       0
```

FONTOS! A legutolsó elem után is kell rakni egy tabulátort a szomszédságmátrixnál.

Mentés példa:

```
Doc
Sun Dec 01 19:03:02 2019
true
gm2
rendelo
kulcs   gyogyszerek
ugyeleti rend (n: r,f; e: o,a)  korlap
Teszt_Elek: Nagyon nagy fajdalmaim vannak, de az orvos nem kuld nyugatot.
orvos: A recepcios nagyon kedves asszony. Neha komolyan azt hiszem, szed valamit.
recepcios: Szerintem az emberek tul buskomorak manapsag, a fajdalmakat nem gyogyszerekkel kellene kezelni, hanem spiritualisan megtisztulni.
apolo: A fonover mostanaban nagyon nyuzott es furcsa, szerintem rosszban santikal.
fonover: Az apolo mindig elviszi a zarhato szekreny kulcsat, vajon mit rejtegethet ott?
orvos: Ezek a Teszt Elek nevu paciens elveszett gyogyszerei!
```

Sorrend kötött, de ezt a program automatikusan kezeli, szabad kézzel nem kell belenyúlni.

A fájlkezelés során elválasztásokra tabulátort kell használni. Fontos, hogy ne hagyjunk ki elemeket, a **helyes formátum az adatokat betápláló felelőssége**! *Fejlesztési lehetőség: táblázatos adatbázis használata.*

Használt eszközök, források:

A program a Code::Blocks fejlesztői környezetben készült. Felhasznált külső könyvtárak az InfoC oldalon megtalálható debugmalloc és c-econio.

További linkek:

Mentés idejének beírásához

txt kiterjesztés szűréséhez

Megjegyzés:

A program a hibás bemeneteket kezeli és jelzi.

Előzetesen jelzésre került, hogy a progam nem tud tökéletesen ékezeteket kezelni. Ezért végig törekedjünk a következetes ékezetmentességre, de a programba drótozott szövegek helyesen íródnak ki, így azok problémát nem okoznak.

Egyéb fejlesztési lehetőség: gamemode-ok helyett játékcímek