Programozói dokumentáció

(Hajdú Patrik Zsolt – RP329D – utolsó frissítés dátuma: 2024.11.19)

1. Projekt Felépítése

A projekt forráskódjai a következő fájlokban találhatóak:

- **debugmalloc.h**: Segéd fájl a dinamikus memóriakezelés hibáinak detektálására.
- **struktura.h**: Az adatszerkezetek és típusdefiníciók deklarációit tartalmazza.
- **seged.c**: A programhoz általánosan szükséges függvények implementációját tartalmazza.
- **seged.h**: A programhoz általánosan szükséges függvények deklarációit tartalmazza.
- **szotar_muveletek.c**: A szótár műveletek végrehajtásához szükséges függvények implementációja.
- **szotar_muveletek.h**: A szótár műveletekhez szükséges függvények deklarációit tartalmazza.
- **lista_muveletek.c**: A láncolt lista műveletek végrehajtásához szükséges függvények implementációja.
- **lista_muveletek.h**: A láncolt lista műveletekhez szükséges függvények deklarációit tartalmazza.
- **main.c**: A fő forráskód, amely a program működését megvalósítja.

2. Projektben Használt Könyvtárak

A program C nyelven íródott és szükség van a következő könyvtárakra:

- **stdio.h**: Be és kimeneti műveletekhez.
- **stdlib.h**: Dinamikus memóriakezeléshez.
- **string.h**: Karakterláncműveletekhez.
- **ctype.h**: Karaktervizsgálati műveletekhez.
- **stdbool.h**: Logikai típus használatához.
- **time.h**: Időkezeléshez.

3. Projekt Fájlkezelése

Fájlnév formai követelmény:

I. A fájlnevének helyes megadása során a leendő új szótár file névének **első szava azonosítja a szótár első nyelvét**, majd **_ jellel** elválasztva a **második szava azonosítja a szótár második nyelvét**.

II. Illetve a program a fájl beolvasására képes akkor is, ha a fájlnév végén „**.txt**” szerepel, és abban az esetben is, ha a fájlnév végén nincs „**.txt**” kiterjesztés megadva.

III. Helyes fájlnév például: **magyar_olasz.txt** vagy **magyar_olasz**

Fájl struktúrájának formai követelménye:

A struktúra a következő formátumban van: `nyelv1;nyelv2;ny1-szó,ny2-szó;`

IV. Először az *első nyelv neve* (nyelv1), majd egy **;** **jellel** elválasztva a *második nyelv neve* (nyelv2).

V. Ezután jön az *első nyelv* egy *szava*, amit egy **,** **jellel** elválasztva követ a *második nyelv* megfelelő *szava*.

VI. Végül egy **;** **jel** következik, és innentől az **V.** lépés ismétlődik a további szópárokra.

A program feltételezi, hogy a **fájlok ékezetmentes karaktereket tartalmaznak**, és **minden szónak van megfelelő párja.** Illetve azt is feltételezi, hogy a fájlban **legalább egy szópár szerepel** a nyelvek helyes megadása mellett.

4. Főmenü Dokumentációja

4.1 main.c tartalma:

Főmenü

- Létrehozza a Fo_Lista láncolt listát, amelyet feltölt 3 előre beállított szótárral, majd ezekhez kapcsolódó szópárokkal, így biztosítva, hogy a program a nyelvek közötti szópárok kezelésére készen álljon az indulástól kezdve.
- A program egy folyamatos, „végtelen” ciklusban működik, amely lehetővé teszi, hogy a felhasználó a főmenü 0 és 4 közötti opcióiból válasszon.
- A bemeneti menüpontokat karakterként olvassa be a program, majd ellenőrzi, hogy a bevitel érvényes szám-e (0 és 4 között). Az érvényes számokat a rendszer konvertálja és eltárolja a menüpont választás kivitelezésére.
- A kiválasztott menüpont funkcióját switch-case struktúra kezeli.

Case 0 - Elérhető szótárak

Ez az opció lehetővé teszi a programban elérhető szótárak megtekintését és különféle műveletek végrehajtását az egyes szótárakon:

- **Szótárak listázása:** Megvizsgálja, hogy van-e legalább egy szótár, amelyen dolgozhatunk; ha nincs, hibaüzenetet ad, és visszalép a főmenübe.
- **Szótár műveletek:** A felhasználó kiválaszthat egy szótárt további műveletekre:
    - **Szópárok listázása:** A kiválasztott szótár szópárjait listázza ki.
    - **Szópárok fájlba írás:** A szótár összes szópárját fájlba menti a megadott névvel.
    - **Új szópár felvétele:** Új szópárt ad a szótárhoz, amelyet a felhasználó által megadott szavak alkotnak.
    - **Szópárok szerkesztése:** A kiválasztott szópár szavait szerkeszti a felhasználó bemenete alapján.
    - **Szópárok törlése:** Egy kiválasztott szópárt töröl a szótárból.
    - **Quiz:** A felhasználónak egy „quiz” lehetőséget kínál, amely során a program véletlenszerűen kiválaszt egy szót a gyakorláshoz, és visszajelzést ad a helyes és helytelen válaszokról.
    - **Kilépés**: Ez az opció a jelenlegi menüpontból való visszalépést biztosítja, visszairányítva a felhasználót a főmenübe.

Case 1 - Új szótár létrehozása

Ez az opció lehetőséget nyújt egy új szótár létrehozására a következő műveletekkel:

- **Új szótár nyelveinek beolvasása**: A felhasználó megadja az új szótárhoz tartozó nyelvpárokat. Ha már létezik egy ilyen nyelvpárokkal rendelkező szótár, hibaüzenetet ad.
- **Szópárok felvétele az új szótárhoz**: A szótárhoz hozzáadja az első szópárt a felhasználó által megadott szavak alapján.

Case 2 - Meglevő szótár törlése

Ez az opció lehetővé teszi egy kiválasztott szótár törlését:

- **Szótár törlése**: A felhasználónak ki kell választania egy szótárt, amelyet törölni kíván. A program ellenőrzi, hogy a kiválasztott szótár létezik-e; ha igen, törli, és üzenetben tájékoztatja a felhasználót.

Case 3 - Szótár fájlból olvasása

Ez az opció lehetőséget biztosít egy szótár beolvasására egy fájlból:

- **Fájlnév beolvasása**: A felhasználónak meg kell adnia a beolvasandó fájl nevét, és a program ellenőrzi, hogy a fájl létezik-e. Ha a fájl nem található, hibaüzenetet ad.
- **Szótár és szópárok beolvasása**: A program megvizsgálja, hogy a fájl tartalmaz-e már létező szótárakat; ha nem, a fájl tartalmát beolvassa, és a szótár listához adja.

Case 4 - Kilépés

- Ez a lehetőség lezárja a programot. A program ekkor kiírja a kilépési üzenetet, lezárja a menü ciklusát, és a dinamikusan foglalt memória felszabadítása után befejezi működését.

Memóriakezelés

A program végén felszabadítja a „Fo_Lista” nevű láncolt lista összes elemét, amely a program futása alatt dinamikusan lefoglalt memóriaterületeket tartalmazza. Ez megakadályozza a memória szivárgását, biztosítva, hogy minden lefoglalt memória felszabaduljon, amikor a program kilép.

Hibakezelés

Ha a felhasználó olyan bemenetet ad meg, amely nem felel meg a 0, 1, 2, 3 vagy 4 értékeknek, a program hibaüzenetet jelenít meg, amely tájékoztatja a felhasználót a helytelen bemenetről, majd visszatér a főmenübe a helyes bemenet újbóli megadásához.

5. Adatszerkezetek Dokumentációja

5.1 struktura.c / struktura.h tartalma:

Szótár struktúra

Célja: A Szótár struktúra tárolja a két nyelv szavait és azok megfeleltetéseit.

- **char *nyelv1**: Az első nyelv neve (dinamikusan foglalt karakterlánc).
- **char *nyelv2**: A második nyelv neve (dinamikusan foglalt karakterlánc).
- **char szoparok_tomb[2][1000][50+1]**: Kétdimenziós tömb, amely két nyelv közötti szópárokat tartalmazza (mindegyik szópár legfeljebb 50 karakter hosszú).
- **int szoparok_szama**: Az aktuális szótárban található szópárok száma.

Lista struktúra

Célja: A láncolt lista alapvető eleme, amely minden szótárat külön csomópontban tartalmaz.

- **Szotar szotar**: Az adott listaelemhez tartozó szótár.
- **struct Lista *kov**: A láncolt lista következő elemére mutató pointer.

6. Függvények Dokumentációja

6.1 seged.c / seged.h tartalma:

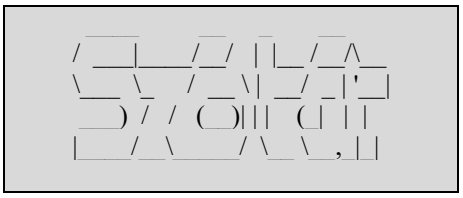
A felhasználói felület elemeinek megjelenítéséért felelős következő függvények: Konzol_Torles(), Szotar_Felirat(), Torol_Szotar_Felirat(), Tovabb_Gombra(), Menu_Opciok() és a Szotar_Opciok().

Konzol_Torles() – függvény:

Ez a függvény felelős a konzol kiürítéséért, és platformfüggetlen módon működik mind Windows, mind Linux rendszereken. A system(„cls”) parancsot használja Windows rendszerekhez, és a system(„clear”) parancsot Linux rendszerekhez.

Szotar_Felirat() – függvény:

Ez a függvény ASCII karakterekkel jeleníti meg a „Szótár” feliratot, amely javítja a program esztétikai megjelenését és egyedi hangulatot kölcsönöz neki.



Torol_Szotar_Felirat() – függvény:

Ez a függvény ASCII karakterekkel jeleníti meg a „Szótár” feliratot, majd kitisztítja a konzolt, ezzel fokozva a program esztétikai megjelenését és a felhasználói élményt.

Tovabb_Gombra() – függvény:

A függvény kap egy string bemenetet, amit ki ír. Majd vár egy billentyűleütésre, ami kiüríti a bemeneti buffert és a konzolt. Ezáltal a felhasználó kényelmesen folytathatja a program használatát.

Használat:

```
void Tovabb_Gombra(const char *Szoveg_Ki_Ir);
```

Példa:

```
Tovabb_Gombra("\tHIBA: Nincs szotar a programban, igy nem lehet veluk dolgozni!");
```

Menu_Opciok() – függvény:

Ez a függvény felelős a főmenü pontjainak megjelenítéséért.

Szotar_Opciok() – függvény:

Ez a függvény felelős a „**[0] - Elérhető szótárak**” főmenüpont kiválasztása után a szótárral kapcsolatos elérhető menüpontok megjelenítéséért.

Szoveg_Beolvas() – függvény:

A függvény kiír egy kérdést, majd beolvassa a választ. A válasz szövegének ellenőrzése során megvizsgálja, hogy nem tartalmaz-e számot; ha igen, akkor a megadott hibaüzenetet adja vissza. Ha az „**int Spec_Esett**” **paraméter értéke 1**, a program a „**[5] – QUIZ**” opcióban egy speciális kérdést tesz fel, amely csak ebben az esetben jelenik meg; más érték esetén ez a kérdés nem kerül kiírásra.

Használat:

```
void Szoveg_Beolvas(bool Kerdes_Kell_E, const char *Kerdes, char *Be_Szoveg, int
Spec_Esett, const char *Spec_Out, bool Hiba_Kell_E, const char* Hiba);
```

Példa:

```
Szoveg_Beolvas(true, "Add meg a file nevet: ", Ki_Ir_File_Neve_Input, -1, "", true,
"Nem tartalmazhat egy file neve szamot!");
```

Szam_Beker_Intervallumban() – függvény:

A függvény bekér egy számot egy adott intervallumban. Ha a szám az intervallumon belül van, visszaadja; ha nem, akkor kiírja a megadott hibaüzenetet, és addig ismétli a bekérést, amíg az intervallumba eső számot nem kap. Az „**int Tipus**” paraméter csupán egy extra szöveg megjelenítését befolyásolja, jelezve az aktuális alkalmazás célját.

- **Tipus = 0**: Egy szótár kiválasztása
- **Tipus = 1**: Szótár törlése
- **Tipus = 2**: Egy szótárral kapcsolatos művelet kiválasztása
- **Tipus = 3**: Szótár sorszámának megadása
- **Tipus = Egyéb érték**: Csak bekéri a számot, és ellenőrzi, hogy az intervallumba esik-e.

Használat:

```
int Szam_Beker_Intervallumban(Lista *Fo_Lista, int Also_Hatar, int Felso_Hatar, int Tipus, bool Kell_E_Hiba_Uzenet, const char *Hiba_Uzenet);
```

Visszatérési érték[1]:

A visszaadott érték egész szám, amely a megadott intervallumba esik.

Példa:

```
int Hanyadik_Szopar_Szerkeszt = Szam_Beker_Intervallumban(NULL, 0, (Akt_Szopar_Javit->szotar.szoparok_szama - 1), 3, true, "Ervenytelen szopar szamot adott meg!");
```

Van_E() – függvény:

A függvény ellenőrzi, hogy a „**char *File_Neve**” paraméterben megadott nevű szótár már szerepel-e a tárolt adatok között. (Feltételezzük, hogy ide csak a fájlkezelési struktúrának megfelelő stringek kerülnek megadásra.)

Használat:

```
bool Van_E(Lista *Fo_Lista, char *File_Neve);
```

Visszatérési érték:

A visszaadott logikai érték (true vagy false), jelzi, hogy a beolvasott nevű szótár már létezik-e a tárolt szótáraink között.

Példa:

```
bool Van_E_Mar_Ilyen = Van_E(Fo_Lista, Be_Olvas_File_Neve);
```

File_Megtalalhato_E() – függvény:

A függvény ellenőrzi, hogy a „**const char *File_Neve**” paraméterben megadott nevű fájl megtalálható-e a számítógépen.

Használat:

```
bool File_Megtalalhato_E(const char *File_Neve);
```

Visszatérési érték:

A visszaadott logikai érték (true vagy false), azt jelzi, hogy a megadott fájl megtalálható-e vagy sem.

Példa:

```
bool Letezik_E_File = File_Megtalalhato_E(Be_Olvas_File_Neve);
```

[1] Visszatérési érték: Csak a nem void típusú függvények esetében van megadva, amely leírja, mit ad vissza a függvény.

Txt_Vizsgalo() – függvény:

A függvény a megadott „**char *Input_Nev**” stringet ellenőrzi, hogy tartalmazza-e a „**.txt**” kiterjesztést. Ha a fájlnév már tartalmazza a „**.txt**” végződést, akkor a függvény visszaadja az eredeti stringet. Ha nem, akkor a függvény kiegészíti a stringet „**.txt**”-el, és ezt az új stringet adja vissza. (Feltételezzük, hogy ide csak a fájlkezelési struktúrának megfelelő stringek kerülnek megadásra.)

Használat:

```
char *Txt_Vizsgalo(char *Input_Nev);
```

Visszatérési érték:

A visszaadott érték egy string, amely „**.txt**” kiterjesztéssel végződik.

Példa:

```
char *Ki_Ir_File_Neve = Txt_Vizsgalo(Ki_Ir_File_Neve_Input);
```

6.2 szotar_muveletek.c / szotar_muveletek.h tartalma:

A programban a szótárakkal végzett műveletekhez az alábbi függvények állnak rendelkezésre: Ki_Listazo(), Uj_Szo_Add(), Szopar_Torlo() és Szopar_Javito().

Ki_Listazo() – függvény:

Ez a függvény felsorolja a megadott szótár összes szópárját, és a „**bool Fileba_Listaz**” paraméter segítségével eldönti, hogy az eredményt fájlba írja-e vagy a konzolra.

- **Ha Fileba_Listaz értéke true**, akkor a szópárokat egy fájlba menti.
- **Ha Fileba_Listaz értéke false**, akkor a szópárokat a konzolra írja ki.

Használat:

```
void Ki_Listazo(Lista *Fo_Lista, int Melyik_Szotar, bool Fileba_Listaz, const char *File_Neve);
```

Példa:

```
Ki_Listazo(Fo_Lista, Valasztott_Szotar, true, "szotar_kiiras.txt");
```

Uj_Szo_Add() – függvény:

A függvény segítségével új szópárt adhatunk hozzá a megadott szótárhoz, feltéve, hogy a szótár nem érte el a maximális kapacitását. Ha a szótár már 1000 szópárt tartalmaz, a függvény nem tesz hozzá új elemet, és hibaüzenetet küld.

Használat:

```
void Uj_Szo_Add(Lista *Fo_Lista, int Melyik_Szotar, const char *Szo_Nyelv1, const char *Szo_Nyelv2);
```

Példa:

```
Uj_Szo_Add(Fo_Lista, Valasztott_Szotar, Uj_Szo_Nyelv1, Uj_Szo_Nyelv2);
```

Szopar_Javito() – függvény:

A függvény lehetővé teszi a kiválasztott szótárban egy adott szópár módosítását. A felhasználó döntheti el, hogy melyik elemet kívánja javítani az alábbi opciók szerint:

- **F vagy f**: Az első nyelv szavának módosítása.
- **S vagy s**: A második nyelv szavának módosítása.
- **B vagy b**: Mindkét szó javítása egyszerre.

A választott módosításokat a függvény végrehajtja a szótáron.

Használat:

```
void Szopar_Javito(Lista *Fo_Lista, int Melyik_Szotar, int Hanyadik_Szopar);
```

Példa:

```
Szopar_Javito(Fo_Lista, Valasztott_Szotar, Hanyadik_Szopar_Szerkeszt);
```

Szopar_Torlo() – függvény:

A függvény megadott szótár egy adott szópárját törli. A törlés hatására a többi szópár „fölcsúszik”, így a szótárban maradó szópárok folyamatosak maradnak.

Használat:

```
void Szopar_Torlo(Lista *Fo_Lista, int Melyik_Szotar, char Melyiket_Torol);
```

Példa:

```
Szopar_Torlo(Fo_Lista, Valasztott_Szotar, Hanyadik_Szopart_Torol);
```

6.3 lista_muveletek.c / lista_muveletek.h tartalma:

A programban a listán végzett műveletekhez az alábbi függvények állnak rendelkezésre: L_Beszur(), L_Felszabadit(), Lista_Hossza(),L_Kiir(), Szotar_Listabol_Torlo() és Lista_Lepteto().

L_Beszur() – függvény:

A függvény egy új elemet szúr be a láncolt lista végére. Az új listaelem tárolja a szótár két nyelvét, a szópárok tömbjét, és a szópárok számát.

Használat:

```
void L_Beszur(Lista **Fo_Lista, const char* nyelv1, const char* nyelv2);
```

Példa:

```
L_Beszur(&Fo_Lista, "francia", "magyar");
```

L_Felszabadit() – függvény:

A függvény felszabadítja a teljes láncolt listát, törölve ezzel a dinamikusan lefoglalt memóriát minden elemre vonatkozóan.

Használat:

```
void L_Felszabadit(Lista *Fo_Lista);
```

Példa:

```
L_Felszabadit(Fo_Lista);
```

Lista_Hossza() – függvény:

A függvény visszaadja a láncolt lista hosszát, vagyis a benne tárolt szótárak számát.

Használat:

```
int Lista_Hossza(Lista *Fo_Lista);
```

Visszatérési érték:

A visszaadott érték egész szám, amely a lista elemeinek számát reprezentálja.

Példa:

```
int Hany_Szotar = Lista_Hossza(Fo_Lista);
```

L_Kiir() – függvény:

A függvény kiírja a listában található szótárakat, mindegyiket egy sorszámmal ellátva, ahol a sorszámozás 0-tól kezdődik.

Használat:

```
void L_Kiir(Lista *Fo_Lista);
```

Példa:

```
L_Kiir(Fo_Lista);
```

Szotar_Listabol_Torlo() – függvény:

A függvény törli a listában található, megadott pozíciójú szótárat.

Használat:

```
void Szotar_Listabol_Torlo(Lista **Fo_Lista, int Melyik_Szotar);
```

Példa:

```
Szotar_Listabol_Torlo(&Fo_Lista, Valasztott_Szotar);
```

Lista_Lepteto() – függvény:

A függvény lépésről lépésre halad előre a láncolt listában, amíg el nem éri a megadott pozíciójú szótárt.

Használat:

```
Lista *Lista_Lepteto(Lista *Fo_Lista, int Valasztott_Szotar);
```

Visszatérési érték:

Visszaad egy pointert a listában található elemre a megadott index pozíción. Ha az indexen nem található elem, NULL értékkel tér vissza.

Példa:

```
Lista *Akt_Elem = Lista_Lepteto(Fo_Lista, Valasztott_Szotar);
```