Programozói dokumentáció

Adatstruktura.h:

Itt van a Strázsa(sentinel) és a láncolt lista egyik eleme(Node) definiálva:

```
typedef struct Node{
        char *data;
        struct Node *next;
}Node;
```

Itt tárolom a szövegeket, és a kövi lista elemét(láncolt lista miatt)

```
typedef struct sentinel{
        struct Node *first;
        struct Node *last;
}sentinel;
```

Itt a strázsát definiálom ami azért felelős hogy egyszerűbbek, és átláthatóbbak legyenek a láncolt listás függvények

```
sentinel *init(void);
int adder(sentinel **list, char *string);
void freeer(sentinel *s);
void freeall(sentinel **lista, int meret);
bool search(sentinel **lista, char *word);
void deleter(sentinel **lista, char *word);
void writer(sentinel **lista, int size);
void purger(sentinel **lista, int size);
```

- **sentinel:** ezzel tudom a láncolt listákat létrehozni(strázsát, első, utolsó elemet) emiatt nem kell neki megadni semmilyen adatot és a Strázsára mutató pointert adja vissza.
- **adder:** ez a függvény az adott adathalmazba(sentinel **list) belerakja a megadott szöveget(char *string), amit a megadott elrendezés szerint bele is rak, kimenet nem kéne, viszont a hibák kezelése miatt egy int értéket ad vissza aminél a 0-a minden rendben lefutott, az 1- a már benne van az adatok között a berakandó adat, és 2- az hogy a memória foglalásnál hiba lépett fel. Hiba észlelésekor a program a futásában megakad, és kiírja a hiba okát.
- **freenode:** felszabadítja azt az elemet amit megadunk neki.
- **freeallnodes:** a felszabadítja a megadott strukturaban az első x elemet, ez 255, mert a kibővített ASCII táblázat 256 karakterű de ezek indexek és emiatt ekkora a maximum tömb amit használ a program és emiatt 255-nyit szabadít fel, azért nem a függvényben van ez belekódolva mert így könnyebb lesz bővíteni a programot.
- **search:** bool típusú értéket ad vissz, azt hogy megtalálta-e az keresendő elemet az adatstruktúrában, vagy nem, true ha megtalálta, false ha nem. Ennek is az adatstruktúrát és a szót kell megadni amit keresünk.
- **deleteonenode:** Ez a függvény töröl egy elemet a listából. Ez a függvény fel is szabadít. Meg kell adni hogy melyik adatszerkezetben van a megadott szó, megkeresi, majd törli.

- **deleteallnodes:** meg kell adni azt az adatstruktúrát amit teljesen törölni akarunk, annyi marad meg benne ami inicializáláskor keletkezett, ehhez is meg kell adni a méretet ugyanazért az ok miatt mint fentebb, hogy könnyebb legyen majd fejleszteni.
- **writer:** kimenetek és bemenetek ugyanaz mint a fentebbiek nél, az adatstruktúra, mérete és void a visszatérési értéke. Ez a függvény kiírja az összes elemet ami az adatstruktúrában van benne. A kulcsot kiírja az elemek elé []-ok közé.

ifinput.h:

itt vannak azok a függvények ami a beolvasó függvényhez szükségesek.

```
char *reader(void);
```

- **reader:** A függvény beolvassa a terminálba beírt adatot, de ez bármilyen hosszú adat beolvasására képes, dinamikus memóriakezelés miatt, amit visszaad visszatérési értékként.

jsonloader.h:

itt tölti be meg az adatstruktura.h-t mert mindenhova kell az ahova ez és így biztos meglesz a programban ha ezt használja.

```
int loader(sentinel **lista,char *name);
int fwriter(char *file, sentinel **lista, int size);
int freader(char **stream, char *name);
int dataprocession(char *stream, sentinel **lista);
```

- **loader:** meg kell adni hogy melyik adatstruktúrába olvassa be melyik file nevét(a file név kiterjesztés nélküli, a program adja hozzá a kiterjesztést). Viszzatérési értéke int hogy a hibakezelést megkönnyítsem. 0-ás: a program rendesen lefutott, 1-es: file megnyitási hiba, 2-es memóriafoglalási hiba.Hiba észlelésekor a program a futásában megakad, és kiírja a hiba okát.
- **freader:** beolvassa a file tartalmát, majd ezt belerakja a stream változóba.
- **dataprocession:** Feldarabolja a stream változóból a szöveget és elraktározza az adatstruktúrában.
- **fwriter:** meg kell adni a fájlnevet amibe kiír(kiterjesztés nélkül), majd az adatstruktúrát amit ki akarunk írni, majd az adatstruktúra méretét a további bővítés lehetősége miatt. 0-ás: a program rendesen lefutott, 1-es: file megnyitási hiba, 2-es memóriafoglalási hiba. Hiba észlelésekor a program a futásában megakad, és kiírja a hiba okát.

A program tartalmazza a **menu.c** file-t.
Ez az a file ami a menü létrehozásáért felelős, emiatt ez az a file ami a függvényeket meghívja és így az egész programot irányítja.

Tartalmazza a **ifinput.c** file-t.
Ez a file azért felelős hogy testszőleges hosszúságú szövegeket be tudjak olvasni.

Tartalmazza a **jsonloader.h** file-t.
Ez a file felelős a külső adattárolás-ért azaz a fájlokba való írásért és olvasásért.

Tartalmazza még az **adatstruktura.h** file-t ami a legfontosabb file a programban. Ebbe a fileban vannak az adatstruktúra létrehozásáért felelős függvények, és az adatstruktúra manipulálásáért felelős függvények.

Továbbá használja a program az **stdio.h**-t ami a beolvasásért, kiírásért felelős.
A **stdlib.h**-t ami a terminál manipulálásáért, azaz operációs rendszereken lévő terminál parancsok futtatásáért felelős(a terminál törléséért). És a dinamikus memóriakezelés hez is kell.
Használja még a **stdbool.h**-t ami azért felelős hogy a program használhasoon boolean típusú változókat.
Tartalmazza a **string.h**-t ami azért felelős hogy a string típusú változókat manipulálhassa a program, stringek összehasonlítása.
És végül tartalmazza a **debugmalloc.h**-t ami azért felelős hogy szóljon ha memóriaszivárgás van a programban.

A program csak a **debugmalloc.h**-című külső könyvtárat használja, és a többi c-file hoz tartozó .h fájlok szükségesek. Egyébként csak a szabványos c függvényeket használja a program.

A fordításhoz szükséges a **menu.c**, **jsonloader.c**, **ifinput.c**, **adatstruktura.c**, c file-ok és a **jsonloader.h**, **ifinput.h**, **adatsruktura.h** file-ok. A fordítást a gcc(The GNU Compiler Collection) használatával lesz bemutatva. A program fordításához a compilernek a négy c file-t kell megadni paraméterként. Majd utána egy -o paraméter után a kimeneti(compileolt file nevét)kell megadni, fontos hogy a h-file-ok is egy mappában legyenek a futtatható bináris file-al. A lent látható formátumban lehet a c file-okat binárissá compileolni. És ezt a bináris file-t futtathatjuk.

```
gcc menu.c jsonloader.c adatsrtuktura.c ifinput.c -o HashTable
```