Nagyházifeladat Receptkönyv- Programozói dokumentáció

Szükséges környezet

Windows 10, mingw64 gcc compiler bundle

Standard könyvtárak

- String.h -> alap string-kezelő függvények használatához
- Stdbool.h -> igaz/hamis érték
- Ctype.h -> karakterkezeléshez szükséges
- Stdarg.h -> végtelen paraméterű függvények kezeléséhez szükséges
- Math.h ->matematikai függvények
- Stdlib.h -> memóriafoglalás
- Time.h -> időt adja meg, szükséges a randomszám generátorhoz
- Stdio.h -> Kiírás a standard outputra, bekérés a standard input-ról
- Conio.h -> Menükezeléshez, a 2 byte-os karakter bekéréséhez
- Debugmalloc.h -> memóriaszivárgás ellenőrzésére

Fájlok tartalma

string.c -> ez tartalmazza a saját string kezelő függvényeket
components.c -> ez tartalmazza a fő adatszerkezeteket, és az azokhoz tartozó függvényeket
menu.c -> ez tartalmazza a menükezeléshez szükséges adatszerkezeteket és függvényeket
file_operations.c ->ez tartalmazza a fájlkezeléshez szkséges függvényeket
main.c -> ebben van a fő program, adatszerkezetek deklarálása, inicializálása, menü felépítése, navigáció, fájl olvasás/írás és felszabadítás

Struktúrák

typedef char* String

String néven lehet kezelni a karaktertömböket. Csak esztétikai haszna van.

typedef struct IngredientList

Ez egy láncolt lista amiben tárolja a program a megadott összetevőket.

typedef struct IngredientQList

Láncolt lista, hasonlít az előzőhöz, de ez tárolja a megadott összetevő mértékegységét és mennyiségét. A program ilyen listában tárolja a kamra elemeit, illetve a receptekhez szükséges hozzávalókat.

typedef struct InstructionList

Láncolt lista, A program ebben tárolja a receptek instrukcióit.

typedef struct StrList

Láncolt lista, menükezelésnél van rá szükség. Stringekből álló lista, ebben kapja meg a kiírandó listát vagy beolvasandó listát egy adott menü, és ebben adja vissza a beolvasott elemek értékét.

typedef struct RecipeListFound

Kereséshez szükséges láncolt lista. A keresés használatakor ebbe gyűjti össze a program a talált recepteket, és ez alapján jeleníti meg a mü a kiválasztható opciókat.

typedef enum Navigation

Enumerátor, navigációhoz szükséges, a nyílbillentyűk alias-aként használja a program

typedef enum MenuType
Enumerátor, a menü megjelenítéséhez szükséges, az egységes menümegjelenítő függvény ez alapján írja a képernyőre a tevékenységet (pl felhasználói bekérés vagy lista kiírás)

typedef struct MenuList
Láncolt lista, egy adott menü menüpontjait tartalmazza.

typedef struct Menu
Láncolt lista, egy adott menüt ír le. Tárolja az útvonalát, menüpontjait, előző menüjét, megjelenítési típusát, megjelenítendő/bekérendő listáját.

typedef struct Response
Egy menü visszatérési értéke. Amikor kiválaszt a felhasználó egy menüpontot, az egységes menümegjelenítő-függvény egy ilyen típussal tér vissza, amely tartalmazza a választott menüpontot, a bekért elemek listáját, vagy egy elkészített láncolt lista pointerét.

Függvények

STRING.C

String strcopy(String _inpt);
A paraméterként kapott stringet lemásolja, dinamikusan foglal neki egy pontosan akkora méretet amekkora kell, és visszatér a hely címével.

int strlength(String _inpt);
Megadja a paraméterként kapott string hosszát. NULL értékre van benne hibakezelés

bool strcompare(String _first, String _second);
Összehasonlítja a paraméterként kapott két stringet, és amennyiben megegyeznek true értékkel tér vissza. Ellenkező esetben pedig false-szal.

String trim(String _inpt);
A paraméterként kapott stringből kitörli a szóközöket.

bool substr(String haystack, String needle);
A paraméterként kapott első stringben megkeresi a második string első előfordulását. Amennyiben benne van true-val tér vissza, ha nincs false-szal.

String strToLower(String input);
A paraméterként kapott string minden karaktérét kicsire váltja át.

COMPONENTS.C

bool isIngredientInArray(IngredientList* list, String name);
Megkeresi a listában a megadott nevű elemet és igaz/hamis-sal tér vissza.

void freeIngredientList(IngredientList* list);
Felszabadítja a paraméterként kapott IngredientList typusú láncolt listát

IngredientList* findIngredientByName(IngredientList* ingredients, String name);
A megadott listában megkeresi a megadott nevű elemet, és ha van benne, visszatér annak címével. Ha nincs NULL-lal tér vissza.

IngredientList* findIngredientById(IngredientList* ingredients, int id);
A megadott listában megkeresi a megadott sorszámú elemet, és ha van benne, visszatér annak címével. Ha nincs NULL-lal tér vissza.

IngredientList* deleteIngredient(IngredientList* list, IngredientList* item);
A megadott láncolt listából törli a szintén paraméterben megadott listaelemet.

IngredientList* lastIngredient(IngredientList* start);
A megadott listának az utolsó elemével tér vissza. Ha ürse NULL-lal.

IngredientList* insertIngredient(IngredientList* list, String name);
A megadott listához fűz hozzá egy újonnan létrehozott elemet.

IngredientQList Függvények
Ezek a függvények teljes mértékben ugyan úgy működnek, mint a fentebb leírt IngredientList típusúak, csak más paraméterben kapott listával, és más visszatérési értékkel. Azért van erre szükség, mert a C nyelv nem rendelkezik öröklődéssel, ezért nem lehet egyfajta láncolt listát készíteni, aminek a tulajdonságait és függvényeit öröklik a leszármazottak.

InstructionList Függvények
Ezek a függvények teljes mértékben ugyan úgy működnek, mint a fentebb leírt IngredientList típusúak, csak más paraméterben kapott listával, és más visszatérési értékkel. Azért van erre szükség, mert a C nyelv nem rendelkezik öröklődéssel, ezért nem lehet egyfajta láncolt listát készíteni, aminek a tulajdonságait és függvényeit öröklik a leszármazottak.

RecipeList Függvények (egy része)
Ezek a függvények teljes mértékben ugyan úgy működnek, mint a fentebb leírt IngredientList típusúak, csak más paraméterben kapott listával, és más visszatérési értékkel. Azért van erre szükség, mert a C nyelv nem rendelkezik öröklődéssel, ezért nem lehet egyfajta láncolt listát készíteni, aminek a tulajdonságait és függvényeit öröklik a leszármazottak.

RecipeList* isRecipeInArray(RecipeList* list, RecipeList* item);
A paraméterként kapott listában ellenőrzi hogy szerepel-e a szintén paraméterként kapott listaelem. Ha igen, visszatér annak pointerével, ellenkező esetben NULL-lal.

RecipeListFound Függvények (egy része)
Ezek a függvények teljes mértékben ugyan úgy működnek, mint a fentebb leírt IngredientList típusúak, csak más paraméterben kapott listával, és más visszatérési értékkel. Azért van erre szükség, mert a C nyelv nem rendelkezik öröklődéssel, ezért nem lehet egyfajta láncolt listát készíteni, aminek a tulajdonságait és függvényeit öröklik a leszármazottak.

RecipeListFound* findRecipesByIngredient(RecipeList* list, String ingredient);
A paraméterben megadott receptlistában keresi az olyan recepteket, amik tartalmazzák a megadott hozzávalót. Ezekből csinál egy listát, és annak értékével tér vissza. 0 találat esetén NULL-lal.

RecipeListFound* findRecipesByName(RecipeList* list, String needle);
A paraméterben megadott receptlistában keresi az olyan recepteket, amiknek a neve tartalmazza a megadott stringet. Ezekből csinál egy listát, és annak értékével tér vissza. 0 találat esetén NULL-lal.

RecipeList* randomRecipe(RecipeList* list);
A paraméterben megadott receptlistából véletlenszerűen kiválaszt egy receptet, és annak értékével tér vissza.

StrList Függvények
Ezek a függvények a menü bemenet/kimenet listájaként használatosak. A láncolt lista függvények ugyan úgy működnek mint a feljebb leírtak, csak ennek a típusnak a paraméterével és visszatérési értékével.

StrList* from... függvények
A paraméterben megadott láncolt listákból készít egy egységes, a menüvezérlő számára megjeleníthető listát. Mindegyik függvény a parméterlistából alakítja át egyetlen sornyi string-gé őket.

MENU.C

Menu* lastMenu(Menu* menuList);
A megadott lista utolsó elemével tér vissza, ha üres a lista NULL-lal.

Menu* insertMenu(Menu* menuList, int id, String path, MenuList* nav, Menu* prev, StrList* strlist, MenuType type);
A megadott listához fűz hozzá egy újonnan létrehozott elemet.

Menu* findMenuById(Menu* menuList, int id);
A megadott listában keresi meg a paraméterben megadott azonosítójú elemet. Annak címével tér vissza, vagy NULL-lal.

MenuList* lastMenuList(MenuList* list);
A megadott lista utolsó elemével tér vissza, ha üres NULL-lal.

MenuList* insertMenuList(MenuList* list, String name);
A megadott navigációs láncolt listához fűz hozzá egy új elemet.

MenuList* createMenuList(int args, ...);
A megadott paraméterekből hoz létre egy navigációs láncolt listát. Az első paraméter az argumentumok száma, utána pedig annyi stringet vár a függvény.

Response display(Menu* menu);
Menüvezérlő függvény. Paraméterben kapja meg a kiírandó menüt, azt a konzolja írja, elvégzi az esetleges felhasználói bekéréseket, és navigációs kiválasztásnál visszetér egy Response típusú változóval, amiben tárolja a választásokat és bemeneti dolgokat.

void freeMenu(Menu* menu);
Felszabadítja a megadott Menu típusú láncolt listát.

void freeMenuList(MenuList* list)
Felszabadítja a megadott MenuList typusú láncolt listát.

void printTitlePath(Menu* menu)
A menümegjeleítő függvény használja, rekurzívan kiírja a jelenlegi menünek az előző menüpontjait, a könnyebb navigáció érdekében.

Navigation getNavKey()
A menümegjelenítő használja, ennek a segítségével regisztrálja a kurzormozgatásokat (lépések a menüpontok között), és a menüpont kiválasztást.

FILE_OPERATIONS.C

void writeRecipeFile(RecipeList* recipes);
A receptek.txt fájlba menti a receptek adatszerkezetet, amit paraméterként kap meg. Visszatérési értéke nincs.

void writeIngredientlistFile(IngredientList* ingredients);
A hozzavalok.txt fájlba menti a hozzávalók adatszerkezetet, amit paraméterként kap meg. Visszatérési értéke nincs.

void writePantrylistFile(IngredientQList* pantry);
A kamra.txt fájlba menti a kamra adatszerkezetet, amit paraméterként kap meg. Visszatérési értéke nincs.

RecipeList* readRecipeFile();
Receptek.txt fájlból beolvassa a sorokat, és felépít egy RecipeList láncolt listát, amelynek értékével tér vissza.

IngredientList* readIngredientlistFile();
Hozzavalok.txt fájlból beolvassa a sorokat, és felépít egy IngredientList láncolt listát, amelynek értékével tér vissza.

IngredientQList* readPantrylistFile();
Kamra.txt fájlból beolvassa a sorokat, és felépít egy IngredientQList láncolt listát, amelynek értékével tér vissza.

MAIN.C

Main(void)

A program belépési pontja. Ez a függvény inicializálja a fő adatszerkezetet, meghívja a fájlbeolvasást/kiírást, felépíti a menüszerkezetet, kezeli a menük közti váltogatást, az adatszerkezetek módosítását, kilépés után fájlba menti az adatszerkezetet, és felszabadítja a memóriát.