Nagy házi feladat: Receptkönyv

Készült a Doxygen segítségével
A programot írta: Grób László

1. fejezet

Adatszerkezet-mutató

1.1. Adatszerkezetek

Az összes adatszerkezet listája rövid leírásokkal:

2. fejezet

Fájlmutató

2.1. Fájllista

Az összes dokumentált fájl listája rövid leírásokkal:

3. fejezet

Adatszerkezetek dokumentációja

3.1. Egyedi_osszetevok struktúrareferencia

Egyedi összetevőket tartalmazó lista minden összetevő egyszer kell hogy szerepeljen benne, tartalmazza az összetevők számát és az összetevők tömbjére mutató pointert. A benne levő összetevők rendszerint nem tartalmaznak mennyiségeket csak a nevet és típust, de a lehetőség adott rá.

`#include <file_utils.h>`

Adatmezők

- Osszetevo * **egyedi_osszetevok**
- int **egyedi_osszetevok_szama**

3.1.1. Részletes leírás

Egyedi összetevőket tartalmazó lista minden összetevő egyszer kell hogy szerepeljen benne, tartalmazza az összetevők számát és az összetevők tömbjére mutató pointert. A benne levő összetevők rendszerint nem tartalmaznak mennyiségeket csak a nevet és típust, de a lehetőség adott rá.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- file_utils.h

3.2. Etel struktúrareferencia

Étel struktúra tartalmazza az étel nevét(max 50 karakter) az összetevőinek számát, az összetevők tömbjének pointerét, és az elkészítési útmutatót(max 1000 karakter)

`#include <file_utils.h>`

Adatmezők

- char **nev** [51]
- int **osszetevok_szama**
- Osszetevo ∗ **osszetevok**
- char **elkeszites** [1001]

3.2.1. Részletes leírás

Étel struktúra tartalmazza az étel nevét(max 50 karakter) az összetevőinek számát, az összetevők tömbjének pointerét, és az elkészítési útmutatót(max 1000 karakter)

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- file_utils.h

3.3. Osszetevo struktúrareferencia

Összetevők struckt tartalmazza az összetevő nevét (max 50 karakter), típusát(max 50 karakter), és mennyiségét(double)

`#include <file_utils.h>`

Adatmezők

- char **nev** [51]
- char **tipus** [51]
- double **mennyiseg**

3.3.1. Részletes leírás

Összetevők struckt tartalmazza az összetevő nevét (max 50 karakter), típusát(max 50 karakter), és mennyiségét(double)

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- file_utils.h

3.4. Receptkonyv struktúrareferencia

Receptkönyv struktúra tartalmazza a benne levő ételek számát és az ételek tömbjére egy mutatót.

`#include <file_utils.h>`

Adatmezők

- int **etelek_szama**
- Etel ∗ **etelek**

3.4.1. Részletes leírás

Receptkönyv struktúra tartalmazza a benne levő ételek számát és az ételek tömbjére egy mutatót.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- file_utils.h

4. fejezet

Fájlok dokumentációja

4.1. file_utils.c fájlreferencia

Ebben a modulban kaptak helyet a filokból beolvasó illetve azokba kiíró fv-k, illetve az általánosan használt stdin/stdout olvasó író fv-ek.

```
#include "debugmalloc.h"
#include "file_utils.h"
```

Függvények

- int recept_letezik (Receptkonyv ∗r, const char ∗etel_neve)

  Kap egy receptek structot és egy stringet és megnézi hogy a string egyezik e valamely r.etelek.nev stringgel elvileg case független, de mivel utf karakterek ezért néha bugos, az esetek 99ában működik.
- void receptet_fileba_ment (Receptkonyv ∗r, char ∗file)

  Elmenti a kapott r struktúrát az előre megadott formátumban a kapott paraméter nevű fileba, ha nem létezik létrehozza a program gyökérkönyvtárába.
- Receptkonyv ∗ receptek_beolvas (char ∗file)

  beolvassa a recepteket az előre megadott receptek.txt fileból az r receptkönyv struktúrába a működése egyszerű csak felbloatolja egy rakás hibakezelés.
- void receptkonyv_felszabadit (Receptkonyv ∗r)

  felszabadítja a kapott r struktúrát és minden alstruktúráját
- void recept_kiir (Etel ∗m)

  Kiírja az adott étel struktura adatait(név összetevők elkészítés)
- int osszetevo_letezik (Egyedi_osszetevok ∗e, const char ∗osszetevo_neve)

  Kap egy egyedi osszetevok structot és egy stringet és megnézi hogy a string egyezik-e valamely r.osszetevok.nev stringgel elvileg case független, de mivel utf karakterek ezért néha bugos, az esetek 99ában működik.
- void egyedi_osszetevo_felszabadit (Egyedi_osszetevok ∗e)

  Felszabadítja a kapott e struktúrát és minden alstruktúráját.
- Egyedi_osszetevok ∗ osszetevo_beolvas (void)

  Beolvassa az összetevőket az előre megadott osszetevok.txt fileból az e egyedi összetevők tömbbe.
- void osszetevo_fileba_ment (Egyedi_osszetevok ∗e, Receptkonyv ∗r)

  Összefésüli az e struktúrában található összetevőket a receptkönyv struktúra ételeinek az összetevőivel, hogy az egyedi összetevő struktúra pontosan egyszer tartalmazzon minden összetevőt, majd elmenti azt az előre megadott formátumban az osszetevok.txt fileba, ha nem létezik létrehozza a program gyökérkönyvtárába.

4.1.1. Részletes leírás

Ebben a modulban kaptak helyet a filokból beolvasó illetve azokba kiíró fv-k, illetve az általánosan használt stdin/stdout olvasó író fv-ek.

Dátum

2024-11-08

4.1.2. Függvények dokumentációja

4.1.2.1. egyedi_osszetevo_felszabadit()

```
void egyedi_osszetevo_felszabadit (
            Egyedi_osszetevok * e)
```

Felszabadítja a kapott e struktúrát és minden alstruktúráját.

Paraméterek

e	

4.1.2.2. osszetevo_beolvas()

```
Egyedi_osszetevok * osszetevo_beolvas (
            void )
```

Beolvassa az összetevőket az előre megadott osszetevok.txt fileból az e egyedi összetevők tömbbe.

Visszatérési érték

Egyedi_osszetevok*

4.1.2.3. osszetevo_fileba_ment()

```
void osszetevo_fileba_ment (
            Egyedi_osszetevok * e,
            Receptkonyv * r)
```

Összefésüli az e struktúrában található összetevőket a receptkönyv struktúra ételeinek az összetevőivel, hogy az egyedi összetevő struktúra pontosan egyszer tartalmazzon minden összetevőt, majd elmenti azt az előre megadott formátumban az osszetevok.txt fileba, ha nem létezik létrehozza a program gyökérkönyvtárába.

Paraméterek

e	
r	

4.1.2.4. osszetevo_letezik()

```
int osszetevo_letezik (
            Egyedi_osszetevok * e,
            const char * osszetevo_neve)
```

Kap egy egyedi osszetevok structot és egy stringet és megnézi hogy a string egyezik-e valamely r.osszetevok.nev stringgel elvileg case független, de mivel utf karakterek ezért néha bugos, az esetek 99ában működik.

Paraméterek

e	Az összetevőket tartalmazó struktúra.
osszetevo_neve	Keresett string.

Visszatérési érték

int Visszatér a keresett elem tömbbeli poziciójával+1el, vagy nullával ha nem található.

4.1.2.5. recept_kiir()

```
void recept_kiir (
            Etel * m)
```

Kiírja az adott étel struktura adatait(név összetevők elkészítés)

Paraméterek

m	Az étel struktúra amit ki akarunk írni.

4.1.2.6. recept_letezik()

```
int recept_letezik (
            Receptkonyv * r,
            const char * etel_neve)
```

Kap egy receptek structot és egy stringet és megnézi hogy a string egyezik e valamely r.etelek.nev stringgel elvileg case független, de mivel utf karakterek ezért néha bugos, az esetek 99ában működik.

Paraméterek

r	receptek struct
etel_neve	keresett string

Visszatérési érték

int Visszatér a keresett elem tömbbeli indexével plusz 1(i+1-el), vagy nullával ha nem található (azért kell az i+1 hogy a 0. elemet is helyesen kezelje).

4.1.2.7. receptek_beolvas()

```
Receptkonyv * receptek_beolvas (
            char * file)
```

beolvassa a recepteket az előre megadott receptek.txt fileból az r receptkönyv struktúrába a működése egyszerű csak felbloatolja egy rakás hibakezelés.

Paraméterek

f	A kapott filenév ahonnan beolvas

Visszatérési érték

Receptkonyv*

4.1.2.8. receptet_fileba_ment()

```
void receptet_fileba_ment (
            Receptkonyv * r,
            char * file)
```

Elmenti a kapott r struktúrát az előre megadott formátumban a kapott paraméter nevű fileba, ha nem létezik létrehozza a program gyökérkönyvtárába.

Paraméterek

r	Receptkönyv amit menteni szeretnénk
f	A file neve, amibe menti

4.1.2.9. receptkonyv_felszabadit()

```
void receptkonyv_felszabadit (
            Receptkonyv * r)
```

felszabadítja a kapott r struktúrát és minden alstruktúráját

Paraméterek

r	

4.2. file_utils.h fájlreferencia

A filekezelő fv-ek deklarációi, a program által használt structok deklarációi, és a konzol kiszínezéséhez használt ansi karakterkódok.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "debugmalloc.h"
```

Adatszerkezetek

- struct Osszetevo

  Összetevők struckt tartalmazza az összetevő nevét (max 50 karakter), típusát(max 50 karakter), és mennyiségét(double)
- struct Etel

  Étel struktúra tartalmazza az étel nevét(max 50 karakter) az összetevőinek számát, az összetevők tömbjének pointerét, és az elkészítési útmutatót(max 1000 karakter)
- struct Receptkonyv

  Receptkönyv struktúra tartalmazza a benne levő ételek számát és az ételek tömbjére egy mutatót.
- struct Egyedi_osszetevok

  Egyedi összetevőket tartalmazó lista minden összetevő egyszer kell hogy szerepeljen benne, tartalmazza az összetevők számát és az összetevők tömbjére mutató pointert. A benne levő összetevők rendszerint nem tartalmaznak mennyiségeket csak a nevet és típust, de a lehetőség adott rá.

Makródefiníciók

- #define **ANSI_COLOR_RED** "\x1b[31m"

  A konzol kiszinezésére bevezetett konstansok.
- #define **ANSI_COLOR_GREEN** "\x1b[32m"
- #define **ANSI_COLOR_YELLOW** "\x1b[33m"
- #define **ANSI_COLOR_BLUE** "\x1b[34m"
- #define **ANSI_COLOR_MAGENTA** "\x1b[35m"
- #define **ANSI_COLOR_CYAN** "\x1b[36m"
- #define **ANSI_COLOR_RESET** "\x1b[0m"
- #define **COLOR_RESET** "\033[0m"
- #define **COLOR_BOLD** "\033[1m"
- #define **COLOR_UNDERLINE** "\033[4m"
- #define **COLOR_BLACK** "\033[0;30m"
- #define **COLOR_RED** "\033[0;31m"
- #define **COLOR_GREEN** "\033[0;32m"
- #define **COLOR_YELLOW** "\033[0;33m"
- #define **COLOR_BLUE** "\033[0;34m"
- #define **COLOR_MAGENTA** "\033[0;35m"
- #define **COLOR_CYAN** "\033[0;36m"
- #define **COLOR_WHITE** "\033[0;37m"
- #define **COLOR_BRIGHT_BLACK** "\033[1;30m"
- #define **COLOR_BRIGHT_RED** "\033[1;31m"
- #define **COLOR_BRIGHT_GREEN** "\033[1;32m"
- #define **COLOR_BRIGHT_YELLOW** "\033[1;33m"
- #define **COLOR_BRIGHT_BLUE** "\033[1;34m"
- #define **COLOR_BRIGHT_MAGENTA** "\033[1;35m"
- #define **COLOR_BRIGHT_CYAN** "\033[1;36m"
- #define **COLOR_BRIGHT_WHITE** "\033[1;37m"
- #define **BG_BLACK** "\033[40m"
- #define **BG_RED** "\033[41m"
- #define **BG_GREEN** "\033[42m"
- #define **BG_YELLOW** "\033[43m"
- #define **BG_BLUE** "\033[44m"
- #define **BG_MAGENTA** "\033[45m"
- #define **BG_CYAN** "\033[46m"
- #define **BG_WHITE** "\033[47m"
- #define **BG_BRIGHT_BLACK** "\033[100m"
- #define **BG_BRIGHT_RED** "\033[101m"

- #define **BG_BRIGHT_GREEN** "\033[102m"
- #define **BG_BRIGHT_YELLOW** "\033[103m"
- #define **BG_BRIGHT_BLUE** "\033[104m"
- #define **BG_BRIGHT_MAGENTA** "\033[105m"
- #define **BG_BRIGHT_CYAN** "\033[106m"
- #define **BG_BRIGHT_WHITE** "\033[107m"

Típusdefiníciók

- typedef struct Osszetevo **Osszetevo**
- typedef struct Etel **Etel**
- typedef struct Receptkonyv **Receptkonyv**
- typedef struct Egyedi_osszetevok **Egyedi_osszetevok**

Függvények

- Receptkonyv ∗ receptek_beolvas (char ∗file)

  beolvassa a recepteket az előre megadott receptek.txt fileból az r receptkönyv struktúrába a működése egyszerű csak felbloatolja egy rakás hibakezelés.
- void receptet_fileba_ment (Receptkonyv ∗r, char ∗file)

  Elmenti a kapott r struktúrát az előre megadott formátumban a kapott paraméter nevű fileba, ha nem létezik létrehozza a program gyökérkönyvtárába.
- void receptkonyv_felszabadit (Receptkonyv ∗r)

  felszabadítja a kapott r struktúrát és minden alstruktúráját
- int recept_letezik (Receptkonyv ∗r, const char ∗etel_neve)

  Kap egy receptek structot és egy stringet és megnézi hogy a string egyezik e valamely r.etelek.nev stringgel elvileg case független, de mivel utf karakterek ezért néha bugos, az esetek 99ában működik.
- void recept_kiir (Etel ∗m)

  Kiírja az adott étel struktura adatait(név összetevők elkészítés)
- Egyedi_osszetevok ∗ osszetevo_beolvas (void)

  Beolvassa az összetevőket az előre megadott osszetevok.txt fileból az e egyedi összetevők tömbbe.
- void osszetevo_fileba_ment (Egyedi_osszetevok ∗e, Receptkonyv ∗r)

  Összefésüli az e struktúrában található összetevőket a receptkönyv struktúra ételeinek az összetevőivel, hogy az egyedi összetevő struktúra pontosan egyszer tartalmazzon minden összetevőt, majd elmenti azt az előre megadott formátumban az osszetevok.txt fileba, ha nem létezik létrehozza a program gyökérkönyvtárába.
- void egyedi_osszetevo_felszabadit (Egyedi_osszetevok ∗e)

  Felszabadítja a kapott e struktúrát és minden alstruktúráját.
- int osszetevo_letezik (Egyedi_osszetevok ∗e, const char ∗osszetevo_neve)

  Kap egy egyedi osszetevok structot és egy stringet és megnézi hogy a string egyezik-e valamely r.osszetevok.nev stringgel elvileg case független, de mivel utf karakterek ezért néha bugos, az esetek 99ában működik.
- Osszetevo **o_beolvas1** (void)
- Osszetevo **o_beolvas2** (void)
- Osszetevo **o_beolvas3** (void)
- Etel **i_beolvas** (void)

4.2.1. Részletes leírás

A filekezelő fv-ek deklarációi, a program által használt structok deklarációi, és a konzol kiszínezéséhez használt ansi karakterkódok.

Dátum

2024-11-14

4.2.2. Függvények dokumentációja

4.2.2.1. egyedi_osszetevo_felszabadit()

```
void egyedi_osszetevo_felszabadit (
            Egyedi_osszetevok * e)
```

Felszabadítja a kapott e struktúrát és minden alstruktúráját.

Paraméterek

e	

4.2.2.2. osszetevo_beolvas()

```
Egyedi_osszetevok * osszetevo_beolvas (
            void )
```

Beolvassa az összetevőket az előre megadott osszetevok.txt fileból az e egyedi összetevők tömbbe.

Visszatérési érték

Egyedi_osszetevok*

4.2.2.3. osszetevo_fileba_ment()

```
void osszetevo_fileba_ment (
            Egyedi_osszetevok * e,
            Receptkonyv * r)
```

Összefésüli az e struktúrában található összetevőket a receptkönyv struktúra ételeinek az összetevőivel, hogy az egyedi összetevő struktúra pontosan egyszer tartalmazzon minden összetevőt, majd elmenti azt az előre megadott formátumban az osszetevok.txt fileba, ha nem létezik létrehozza a program gyökérkönyvtárába.

Paraméterek

e	
r	

4.2.2.4. osszetevo_letezik()

```
int osszetevo_letezik (
            Egyedi_osszetevok * e,
            const char * osszetevo_neve)
```

Kap egy egyedi osszetevok structot és egy stringet és megnézi hogy a string egyezik-e valamely r.osszetevok.nev stringgel elvileg case független, de mivel utf karakterek ezért néha bugos, az esetek 99ában működik.

Paraméterek

e	Az összetevőket tartalmazó struktúra.
osszetevo_neve	Keresett string.

Visszatérési érték

int Visszatér a keresett elem tömbbeli poziciójával+1el, vagy nullával ha nem található.

4.2.2.5. recept_kiir()

```
void recept_kiir (
            Etel * m)
```

Kiírja az adott étel struktura adatait(név összetevők elkészítés)

Paraméterek

m	Az étel struktúra amit ki akarunk írni.

4.2.2.6. recept_letezik()

```
int recept_letezik (
            Receptkonyv * r,
            const char * etel_neve)
```

Kap egy receptek structot és egy stringet és megnézi hogy a string egyezik e valamely r.etelek.nev stringgel elvileg case független, de mivel utf karakterek ezért néha bugos, az esetek 99ában működik.

Paraméterek

r	receptek struct
etel_neve	keresett string

Visszatérési érték

int Visszatér a keresett elem tömbbeli indexével plusz 1(i+1-el), vagy nullával ha nem található (azért kell az i+1 hogy a 0. elemet is helyesen kezelje).

4.2.2.7. receptek_beolvas()

```
Receptkonyv * receptek_beolvas (
            char * file)
```

beolvassa a recepteket az előre megadott receptek.txt fileból az r receptkönyv struktúrába a működése egyszerű csak felbloatolja egy rakás hibakezelés.

Paraméterek

f	A kapott filenév ahonnan beolvas

Visszatérési érték

Receptkonyv*

4.2.2.8. receptet_fileba_ment()

```
void receptet_fileba_ment (
            Receptkonyv * r,
            char * file)
```

Elmenti a kapott r struktúrát az előre megadott formátumban a kapott paraméter nevű fileba, ha nem létezik létrehozza a program gyökérkönyvtárába.

Paraméterek

r	Receptkönyv amit menteni szeretnénk
f	A file neve, amibe menti

4.2.2.9. receptkonyv_felszabadit()

```
void receptkonyv_felszabadit (
            Receptkonyv * r)
```

felszabadítja a kapott r struktúrát és minden alstruktúráját

Paraméterek

r	

4.3. file_utils.h

Ugrás a fájl dokumentációjához.

```
00001
00006 #ifndef FILE_UTILS_H
00007 #define FILE_UTILS_H
00008 #include <stdio.h>
00009 #include <stdlib.h>
00010 #include <string.h>
00011 #ifdef _WIN32
00012 #include <windows.h>
00013 #include <locale.h>
00014 #include <wchar.h>
00015 #include <windows.h>
00016 #include <locale.h>
00017 #include <fcntl.h>
00018 #include <io.h>
00019 #endif
00020 #include "debugmalloc.h"
00024 #define ANSI_COLOR_RED     "\x1b[31m"
00025 #define ANSI_COLOR_GREEN   "\x1b[32m"
```

```
00026 #define ANSI_COLOR_YELLOW  "\x1b[33m"
00027 #define ANSI_COLOR_BLUE    "\x1b[34m"
00028 #define ANSI_COLOR_MAGENTA "\x1b[35m"
00029 #define ANSI_COLOR_CYAN    "\x1b[36m"
00030 #define ANSI_COLOR_RESET   "\x1b[0m"
00031
00032   // Text colors
00033 #define COLOR_RESET     "\033[0m"
00034 #define COLOR_BOLD      "\033[1m"
00035 #define COLOR_UNDERLINE "\033[4m"
00036
00037 #define COLOR_BLACK     "\033[0;30m"
00038 #define COLOR_RED       "\033[0;31m"
00039 #define COLOR_GREEN     "\033[0;32m"
00040 #define COLOR_YELLOW    "\033[0;33m"
00041 #define COLOR_BLUE      "\033[0;34m"
00042 #define COLOR_MAGENTA   "\033[0;35m"
00043 #define COLOR_CYAN      "\033[0;36m"
00044 #define COLOR_WHITE     "\033[0;37m"
00045
00046 // Bright (bold) text colors
00047 #define COLOR_BRIGHT_BLACK   "\033[1;30m"
00048 #define COLOR_BRIGHT_RED     "\033[1;31m"
00049 #define COLOR_BRIGHT_GREEN   "\033[1;32m"
00050 #define COLOR_BRIGHT_YELLOW  "\033[1;33m"
00051 #define COLOR_BRIGHT_BLUE    "\033[1;34m"
00052 #define COLOR_BRIGHT_MAGENTA "\033[1;35m"
00053 #define COLOR_BRIGHT_CYAN    "\033[1;36m"
00054 #define COLOR_BRIGHT_WHITE   "\033[1;37m"
00055
00056 // Background colors
00057 #define BG_BLACK     "\033[40m"
00058 #define BG_RED       "\033[41m"
00059 #define BG_GREEN     "\033[42m"
00060 #define BG_YELLOW    "\033[43m"
00061 #define BG_BLUE      "\033[44m"
00062 #define BG_MAGENTA   "\033[45m"
00063 #define BG_CYAN      "\033[46m"
00064 #define BG_WHITE     "\033[47m"
00065
00066 // Bright (bold) background colors
00067 #define BG_BRIGHT_BLACK   "\033[100m"
00068 #define BG_BRIGHT_RED     "\033[101m"
00069 #define BG_BRIGHT_GREEN   "\033[102m"
00070 #define BG_BRIGHT_YELLOW  "\033[103m"
00071 #define BG_BRIGHT_BLUE    "\033[104m"
00072 #define BG_BRIGHT_MAGENTA "\033[105m"
00073 #define BG_BRIGHT_CYAN    "\033[106m"
00074 #define BG_BRIGHT_WHITE   "\033[107m"
00075   /*Az általános adatstruktúráim, a kettős indirekció ott van benne,
00076      *hogy egy ételnek bárhány összetevője lehet illetve bárhány étel lehet a receptkönyvben.*/
00077
00083 typedef struct Osszetevo
00084 {
00085     char nev[51];
00086     char tipus[51];
00087     double mennyiseg;
00088 } Osszetevo;
00094 typedef struct Etel
00095 {
00096     char nev[51];
00097     int osszetevok_szama;
00098     Osszetevo* osszetevok;
00099     char elkeszites[1001];
00100 } Etel;
00105 typedef struct Receptkonyv
00106 {
00107     int etelek_szama;
00108     Etel* etelek;
00109 } Receptkonyv;
00110
00117 typedef struct Egyedi_osszetevok
00118 {
00119     Osszetevo* egyedi_osszetevok;
00120     int egyedi_osszetevok_szama;
00121 } Egyedi_osszetevok;
00122
00123
00124 Receptkonyv* receptek_beolvas(char* file);
00125 void receptet_fileba_ment(Receptkonyv* r, char* file);
00126 void receptkonyv_felszabadit(Receptkonyv* r);
00127 int recept_letezik(Receptkonyv* r, const char* etel_neve);
00128 void recept_kiir(Etel* m);
00129
00130 Egyedi_osszetevok* osszetevo_beolvas(void);
00131 void osszetevo_fileba_ment(Egyedi_osszetevok* e, Receptkonyv* r);
00132 void egyedi_osszetevo_felszabadit(Egyedi_osszetevok* e);
```

```
00133 int osszetevo_letezik(Egyedi_osszetevok* e, const char* osszetevo_neve);
00134
00135 Osszetevo o_beolvas1(void);
00136 Osszetevo o_beolvas2(void);
00137 Osszetevo o_beolvas3(void);
00138 Etel i_beolvas(void);
00139
00140 #endif
```

4.4. k_menu.h fájlreferencia

A qol és kedvencek/bevlista almenük fv deklarációi és include headerei.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <stdbool.h>
#include "file_utils.h"
#include "debugmalloc.h"
#include "o_menu.h"
```

Függvények

- void recept_random (Receptkonyv ∗r)

  Random választ egy receptet, kiírja a nevét, majd ha azt elfogadja a felhasználó kiírja a receptjét.
- void qol_almenu (Receptkonyv ∗r)

  A qol almenü vezérlője, két fő opciót tartalmaz, random keresést, és összetevő alapján történő keresést.
- void receptek_szures (Receptkonyv ∗r)

  Bekér egy összetevőt majd kiírja a listát amit arra az összetevőre szűrt, ha talált elemetket amik tartalmazzák az adott összetevőt akkor megkérdezi hogy folytassa-e a szűkitést. A szűréshez készit egy boolean-okból álló tömböt és ezzel maszkolja az eredeti listát. Ha igen akkor megint kér egy elemet, ha nem akkor egy étel nevét kéri aminek kiírja a receptjét vagy nullát a visszalépéshez.
- void bl_almenu (Receptkonyv ∗r)

  A bevásárlólista almenü vezérlője, három fő opciót tartalmaz, étel összetevőinek hozzáadása a listához a recept neve alapján, a lista törlését, és a lista file-ba mentésének lehetőségét. Ebben az almenüben kaptak helyet a kedvencek is, mivel így kényelmesebb rögtön a bevásálólistához adni őket mintha menüket kellene váltani. A kedvencek szekció a következő almenüket tartalmazza: étel hozzáadása és törlése a kedvencek listából, illetve a kedvencek lista kiírása.
- void bl_hozzaad (Receptkonyv ∗r, Egyedi_osszetevok ∗e)

  Az összetevő hozzáad fv mintájára megnézi hogy az étel létezik e, majd megnézi hogy az összetevő a listában van e, ha igen akkor hozzáadja a mennyiséget ha nem akkor az egészet hozzáadja. Realloccal megy, nem a legszebb de nem kifejezetten lassú, csak ezért nem gyártottam ll-t.
- void bl_torol (Egyedi_osszetevok ∗e)

  Kitörli a bevásárlólista tartalmát, felszabadítja ha volt benne összetevő, beállítja a pointerüket null-ra, és az összetevők számát is null-ra.
- void bl_ment (Egyedi_osszetevok ∗e)

  Elmenti a kész bevásárló listát egy blista.txt file-ba. Ha létezett korábban akkor felülírja.
- void kedvenc_hozzáad (Receptkonyv ∗r, Receptkonyv ∗∗k)

  Hozzáadja a beírt ételt az r-ből a k structba, amennyiben az szerepel az r-ben és még nem szerepel a k-ban. Amennyiben a k null pointer létrehoz egy üres k-t és ahhoz adja hozzá.

4.4.1. Részletes leírás

A qol és kedvencek/bevlista almenük fv deklarációi és include headerei.

Dátum

2024-11-14

4.4.2. Függvények dokumentációja

4.4.2.1. bl_almenu()

```
void bl_almenu (
            Receptkonyv * r)
```

A bevásárlólista almenü vezérlője, három fő opciót tartalmaz, étel összetevőinek hozzáadása a listához a recept neve alapján, a lista törlését, és a lista file-ba mentésének lehetőségét. Ebben az almenüben kaptak helyet a kedvencek is, mivel így kényelmesebb rögtön a bevásálólistához adni őket mintha menüket kellene váltani. A kedvencek szekció a következő almenüket tartalmazza: étel hozzáadása és törlése a kedvencek listából, illetve a kedvencek lista kiírása.

Paraméterek

r	A korábban beolvasott receptek listáját tartalmazó Receptköny struct.

4.4.2.2. bl_hozzaad()

```
void bl_hozzaad (
            Receptkonyv * r,
            Egyedi_osszetevok * e)
```

Az összetevő hozzáad fv mintájára megnézi hogy az étel létezik e, majd megnézi hogy az összetevő a listában van e, ha igen akkor hozzáadja a mennyiséget ha nem akkor az egészet hozzáadja. Realloccal megy, nem a legszebb de nem kifejezetten lassú, csak ezért nem gyártottam ll-t.

Paraméterek

r	A receptek listája
e	A bevásárló lista

4.4.2.3. bl_ment()

```
void bl_ment (
            Egyedi_osszetevok * e)
```

Elmenti a kész bevásárló listát egy blista.txt file-ba. Ha létezett korábban akkor felülírja.

Paraméterek

e	bevásárló lista structja

4.4.2.4. bl_torol()

```
void bl_torol (
            Egyedi_osszetevok * e)
```

Kitörli a bevásárlólista tartalmát, felszabadítja ha volt benne összetevő, beállítja a pointerüket null-ra, és az összetevők számát is null-ra.

Paraméterek

e	A bevásárlólista strucktja

4.4.2.5. kedvenc_hozzáad()

```
void kedvenc_hozzáad (
            Receptkonyv * r,
            Receptkonyv ** k)
```

Hozzáadja a beírt ételt az r-ből a k structba, amennyiben az szerepel az r-ben és még nem szerepel a k-ban. Amennyiben a k null pointer létrehoz egy üres k-t és ahhoz adja hozzá.

Paraméterek

k	kedvencek struktúra
r	receptkönyv struktúra

4.4.2.6. qol_almenu()

```
void qol_almenu (
            Receptkonyv * r)
```

A qol almenü vezérlője, két fő opciót tartalmaz, random keresést, és összetevő alapján történő keresést.

Paraméterek

r	A korábban beolvasott receptek listáját tartalmazó Receptköny struct.

4.4.2.7. recept_random()

```
void recept_random (
            Receptkonyv * r)
```

Random választ egy receptet, kiírja a nevét, majd ha azt elfogadja a felhasználó kiírja a receptjét.

Paraméterek

r	receptkönyv struktúra, tartalmazza a recepteket

4.4.2.8. receptek_szures()

```
void receptek_szures (
            Receptkonyv * r)
```

Bekér egy összetevőt majd kiírja a listát amit arra az összetevőre szűrt, ha talált elemetket amik tartalmazzák az adott összetevőt akkor megkérdezi hogy folytassa-e a szűkitést. A szűréshez készit egy boolean-okból álló tömböt és ezzel maszkolja az eredeti listát. Ha igen akkor megint kér egy elemet, ha nem akkor egy étel nevét kéri aminek kiírja a receptjét vagy nullát a visszalépéshez.

Paraméterek

r	

4.5. k_menu.h

Ugrás a fájl dokumentációjához.

```
00001
00006 #ifndef K_MENU_H
00007 #define K_MENU_H
00008 #include <stdio.h>
00009 #include <stdlib.h>
00010 #include <string.h>
00011 #include <time.h>
00012 #include <stdbool.h>
00013 #ifdef _WIN32
00014 #include <windows.h>
00015 #endif
00016 #include "file_utils.h"
00017 #include "debugmalloc.h"
00018 #include "o_menu.h"
00019
00020 void recept_random(Receptkonyv* r);
00021 void qol_almenu(Receptkonyv* r);
00022 void receptek_szures(Receptkonyv* r);
00023 void bl_almenu(Receptkonyv* r);
00024 void bl_hozzaad(Receptkonyv* r, Egyedi_osszetevok* e);
00025 void bl_torol(Egyedi_osszetevok* e);
00026 void bl_ment(Egyedi_osszetevok* e);
00027 void kedvenc_hozzáad(Receptkonyv* r, Receptkonyv** k);
00028
00029 #endif
```

4.6. o_menu.c fájlreferencia

Az összetevők és receptek almenühöz tartozó fv-ek, egy külön modulba szétszedve.

```
#include "o_menu.h"
```

Függvények

- void **o_menu_kiir** (void)

  Kilistázza az összetevők menüopciókat, stdio-n.
- void osszetevok_almenu (Egyedi_osszetevok ∗∗e)

  Az összetevők almenü vezérlőegysége, azért kell ∗∗e-t kapnia hogy hozzáférjen az eredeti pointerhez ha azon akar módosítani Tartalmaz egy összetevő felvétel, törlés, listázás, és keresés menüpontot.
- void osszetevo_felvesz (Egyedi_osszetevok ∗∗e)

  Megnézi hogy van e összetevő a listában, ha nincs akkor inicializálja a listát. Ezután beolvas egy nevet, ha tartalmazza már a tömb akkor nem engedi újra felvenni egyébként hozzáadja a tömbhöz átméretezés után.
- void osszetevo_kiir (Egyedi_osszetevok ∗e)

  Kiírja az argumentumban kapot struct összetevő elemeinek a listáját.
- void osszetevok_keres (Egyedi_osszetevok ∗e)

  Keres az adott struktúrában a név alapján amit stdin-ről olvas be.
- void osszetevo_torol (Egyedi_osszetevok ∗e)

  Törli az adott összetevőt amit a nevével kell megadni, amennyiben létezik az e structban. Készít egy ideiglenes struckt pointert, lefoglalja neki az új hoszzát, majd átmásolja az összes elemet kivéve a törlendőt, végül felszabadítja az eredetit.
- void **r_menu_kiir** (void)

  Kilistázza a receptek menüopciókat, stdio-n.
- void receptek_almenu (Receptkonyv ∗∗r)

  A receptek almenü vezérlőegysége, azért kell ∗∗r-t kapnia hogy hozzáférjen az eredeti pointerhez ha azon akar módosítani Tartalmaz egy receptek felvétel, törlés, listázás, és keresés menüpontot.
- void recept_felvesz (Receptkonyv ∗∗r)

  Megnézi hogy van-e recept a listában, ha nincs akkor inicializálja a listát. Ezután beolvas egy nevet, ha tartalmazza már a tömb akkor nem engedi újra felvenni, egyébként hozzáadja a tömbhöz átméretezés után.
- void recept_keres (Receptkonyv ∗r)

  Keres az adott struktúrában a név alapján amit stdin-ről olvas be.
- void recept_listaz (Receptkonyv ∗r)

  Kiírja az argumentumban kapot struct receptekben lévő ételek neveit.
- void recept_torol (Receptkonyv ∗r)

  Törli az adott receptet amit az étel nevével kell megadni, amennyiben létezik az r structban. Készít egy ideiglenes struckt pointert, lefoglalja neki az új hoszzát, majd átmásolja az összes elemet kivéve a törlendőt, végül felszabadítja az eredetit.

4.6.1. Részletes leírás

Az összetevők és receptek almenühöz tartozó fv-ek, egy külön modulba szétszedve.

Dátum

2024-11-08

4.6.2. Függvények dokumentációja

4.6.2.1. osszetevo_felvesz()

```
void osszetevo_felvesz (
            Egyedi_osszetevok ** e)
```

Megnézi hogy van e összetevő a listában, ha nincs akkor inicializálja a listát. Ezután beolvas egy nevet, ha tartalmazza már a tömb akkor nem engedi újra felvenni egyébként hozzáadja a tömbhöz átméretezés után.

Paraméterek

e	Receptek listája

4.6.2.2. osszetevo_kiir()

```
void osszetevo_kiir (
            Egyedi_osszetevok * e)
```

Kiírja az argumentumban kapot struct összetevő elemeinek a listáját.

Paraméterek

e	

4.6.2.3. osszetevo_torol()

```
void osszetevo_torol (
            Egyedi_osszetevok * e)
```

Törli az adott összetevőt amit a nevével kell megadni, amennyiben létezik az e structban. Készít egy ideiglenes struckt pointert, lefoglalja neki az új hoszzát, majd átmásolja az összes elemet kivéve a törlendőt, végül felszabadítja az eredetit.

Paraméterek

e	

4.6.2.4. osszetevok_almenu()

```
void osszetevok_almenu (
            Egyedi_osszetevok ** e)
```

Az összetevők almenü vezérlőegysége, azért kell $**$e-t kapnia hogy hozzáférjen az eredeti pointerhez ha azon akar módosítani Tartalmaz egy összetevő felvétel, törlés, listázás, és keresés menüpontot.

Paraméterek

e	A korábban beolvasott összetevők listáját tartalmazó egyedi összetevők struct.

4.6.2.5. osszetevok_keres()

```
void osszetevok_keres (
            Egyedi_osszetevok * e)
```

Keres az adott struktúrában a név alapján amit stdin-ről olvas be.

Paraméterek

e	

4.6.2.6. recept_felvesz()

```
void recept_felvesz (
            Receptkonyv ** r)
```

Megnézi hogy van-e recept a listában, ha nincs akkor inicializálja a listát. Ezután beolvas egy nevet, ha tartalmazza már a tömb akkor nem engedi újra felvenni, egyébként hozzáadja a tömbhöz átméretezés után.

Paraméterek

r	Receptek listája

4.6.2.7. recept_keres()

```
void recept_keres (
            Receptkonyv * r)
```

Keres az adott struktúrában a név alapján amit stdin-ről olvas be.

Paraméterek

r	

4.6.2.8. recept_listaz()

```
void recept_listaz (
            Receptkonyv * r)
```

Kiírja az argumentumban kapot struct receptekben lévő ételek neveit.

Paraméterek

r	

4.6.2.9. recept_torol()

```
void recept_torol (
            Receptkonyv * r)
```

Törli az adott receptet amit az étel nevével kell megadni, amennyiben létezik az r structban. Készít egy ideiglenes struckt pointert, lefoglalja neki az új hoszzát, majd átmásolja az összes elemet kivéve a törlendőt, végül felszabadítja az eredetit.

Paraméterek

r	

4.6.2.10. receptek_almenu()

```
void receptek_almenu (
            Receptkonyv ** r)
```

A receptek almenü vezérlőegysége, azért kell ∗∗r-t kapnia hogy hozzáférjen az eredeti pointerhez ha azon akar módosítani Tartalmaz egy receptek felvétel, törlés, listázás, és keresés menüpontot.

Paraméterek

r	A korábban beolvasott receptek listáját tartalmazó Receptköny struct.

4.7. o_menu.h fájlreferencia

Az összetevők és receptek almenü fv-einek a deklarációja, és az include headerek.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "file_utils.h"
#include "debugmalloc.h"
```

Függvények

- void osszetevok_almenu (Egyedi_osszetevok ∗∗e)

  Az összetevők almenü vezérlőegysége, azért kell ∗∗e-t kapnia hogy hozzáférjen az eredeti pointerhez ha azon akar módosítani Tartalmaz egy összetevő felvétel, törlés, listázás, és keresés menüpontot.
- void osszetevo_felvesz (Egyedi_osszetevok ∗∗e)

  Megnézi hogy van e összetevő a listában, ha nincs akkor inicializálja a listát. Ezután beolvas egy nevet, ha tartalmazza már a tömb akkor nem engedi újra felvenni egyébként hozzáadja a tömbhöz átméretezés után.
- void osszetevo_kiir (Egyedi_osszetevok ∗e)

  Kiírja az argumentumban kapot struct összetevő elemeinek a listáját.
- void osszetevok_keres (Egyedi_osszetevok ∗e)

  Keres az adott struktúrában a név alapján amit stdin-ről olvas be.
- void osszetevo_torol (Egyedi_osszetevok ∗e)

  Törli az adott összetevőt amit a nevével kell megadni, amennyiben létezik az e structban. Készít egy ideiglenes struckt pointert, lefoglalja neki az új hoszzát, majd átmásolja az összes elemet kivéve a törlendőt, végül felszabadítja az eredetit.
- void receptek_almenu (Receptkonyv ∗∗r)

  A receptek almenü vezérlőegysége, azért kell ∗∗r-t kapnia hogy hozzáférjen az eredeti pointerhez ha azon akar módosítani Tartalmaz egy receptek felvétel, törlés, listázás, és keresés menüpontot.
- void recept_felvesz (Receptkonyv ∗∗r)

  Megnézi hogy van-e recept a listában, ha nincs akkor inicializálja a listát. Ezután beolvas egy nevet, ha tartalmazza már a tömb akkor nem engedi újra felvenni, egyébként hozzáadja a tömbhöz átméretezés után.

- void recept_keres (Receptkonyv ∗r)

  Keres az adott struktúrában a név alapján amit stdin-ről olvas be.
- void recept_torol (Receptkonyv ∗r)

  Törli az adott receptet amit az étel nevével kell megadni, amennyiben létezik az r structban. Készít egy ideiglenes struckt pointert, lefoglalja neki az új hoszzát, majd átmásolja az összes elemet kivéve a törlendőt, végül felszabadítja az eredetit.
- void recept_listaz (Receptkonyv ∗r)

  Kiírja az argumentumban kapot struct receptekben lévő ételek neveit.

4.7.1. Részletes leírás

Az összetevők és receptek almenü fv-einek a deklarációja, és az include headerek.

Dátum

2024-11-14

4.7.2. Függvények dokumentációja

4.7.2.1. osszetevo_felvesz()

```
void osszetevo_felvesz (
            Egyedi_osszetevok ** e)
```

Megnézi hogy van e összetevő a listában, ha nincs akkor inicializálja a listát. Ezután beolvas egy nevet, ha tartalmazza már a tömb akkor nem engedi újra felvenni egyébként hozzáadja a tömbhöz átméretezés után.

Paraméterek

e	Receptek listája

4.7.2.2. osszetevo_kiir()

```
void osszetevo_kiir (
            Egyedi_osszetevok * e)
```

Kiírja az argumentumban kapot struct összetevő elemeinek a listáját.

Paraméterek

e	

4.7.2.3. osszetevo_torol()

```
void osszetevo_torol (
            Egyedi_osszetevok * e)
```

Törli az adott összetevőt amit a nevével kell megadni, amennyiben létezik az e structban. Készít egy ideiglenes struckt pointert, lefoglalja neki az új hoszzát, majd átmásolja az összes elemet kivéve a törlendőt, végül felszabadítja az eredetit.

Paraméterek

e	

4.7.2.4. osszetevok_almenu()

```
void osszetevok_almenu (
            Egyedi_osszetevok ** e)
```

Az összetevők almenü vezérlőegysége, azért kell ∗∗e-t kapnia hogy hozzáférjen az eredeti pointerhez ha azon akar módosítani Tartalmaz egy összetevő felvétel, törlés, listázás, és keresés menüpontot.

Paraméterek

e	A korábban beolvasott összetevők listáját tartalmazó egyedi összetevők struct.

4.7.2.5. osszetevok_keres()

```
void osszetevok_keres (
            Egyedi_osszetevok * e)
```

Keres az adott struktúrában a név alapján amit stdin-ről olvas be.

Paraméterek

e	

4.7.2.6. recept_felvesz()

```
void recept_felvesz (
            Receptkonyv ** r)
```

Megnézi hogy van-e recept a listában, ha nincs akkor inicializálja a listát. Ezután beolvas egy nevet, ha tartalmazza már a tömb akkor nem engedi újra felvenni, egyébként hozzáadja a tömbhöz átméretezés után.

Paraméterek

r	Receptek listája

4.7.2.7. recept_keres()

```
void recept_keres (
            Receptkonyv * r)
```

Keres az adott struktúrában a név alapján amit stdin-ről olvas be.

Paraméterek

r	

4.7.2.8. recept_listaz()

```
void recept_listaz (
            Receptkonyv * r)
```

Kiírja az argumentumban kapot struct receptekben lévő ételek neveit.

Paraméterek

r	

4.7.2.9. recept_torol()

```
void recept_torol (
            Receptkonyv * r)
```

Törli az adott receptet amit az étel nevével kell megadni, amennyiben létezik az r structban. Készít egy ideiglenes struckt pointert, lefoglalja neki az új hoszzát, majd átmásolja az összes elemet kivéve a törlendőt, végül felszabadítja az eredetit.

Paraméterek

r	

4.7.2.10. receptek_almenu()

```
void receptek_almenu (
            Receptkonyv ** r)
```

A receptek almenü vezérlőegysége, azért kell ∗∗r-t kapnia hogy hozzáférjen az eredeti pointerhez ha azon akar módosítani Tartalmaz egy receptek felvétel, törlés, listázás, és keresés menüpontot.

Paraméterek

r	A korábban beolvasott receptek listáját tartalmazó Receptköny struct.

4.8. o_menu.h

Ugrás a fájl dokumentációjához.

```
00001
00007 #ifndef O_MENU_H
00008 #define O_MENU_H
00009 #include <stdio.h>
00010 #include <stdlib.h>
00011 #include <string.h>
00012 #ifdef _WIN32
00013 #include <windows.h>
00014 #endif
00015 #include "file_utils.h"
00016 #include "debugmalloc.h"
00017
00018 void osszetevok_almenu(Egyedi_osszetevok** e);
00019 void osszetevo_felvesz(Egyedi_osszetevok** e);
00020 void osszetevo_kiir(Egyedi_osszetevok* e);
00021 void osszetevok_keres(Egyedi_osszetevok* e);
00022 void osszetevo_torol(Egyedi_osszetevok* e);
00023
00024 void receptek_almenu(Receptkonyv** r);
00025 void recept_felvesz(Receptkonyv** r);
00026 void recept_keres(Receptkonyv* r);
00027 void recept_torol(Receptkonyv* r);
00028 void recept_listaz(Receptkonyv* r);
00029
00030 #endif
```

Tárgymutató