Határidőnapló

Fejlesztői dokumentáció

Ezen dokumentum célja, hogy bemutassa a határidőnapló alkalmazás működését, érthetővé tegye az egyes forrásfájlok feladatait és segítse a jövőbeli változtatásokat, további fejlesztéseket.

A projekt ismertetése

A program C programnyelven íródott, feladata egy menüvezérelt határidőnapló megvalósítása. Az alkalmazás parancssoros, ékezetmentes magyar nyelven működik. A felhasználó a menüben a menüpontok melletti szám beírásával lépkedhet, szükség esetén szöveg begépelésére is kéri a program (pl.: esemény nevének beírása).

Felépítés

A program indításkor a main.c main függvénye fut le, ami létrehozza a láncolt lista adatszerkezet első elemét. Ezen kívül a fájlban csak a memóriafelszabadítással foglalkozó függvény van. Ez a program vége előtt lefut. Meghívásra kerül main_menu() függvény, mely átvezet a menu.c fájlba. A program további futását ez a fájl vezérli. Ez tartalmazza mindazon függvényeket, melyek a menü vezérléséért, kiíratásokért felelős. Emellett az event.c fájl tartalmazza az adatszerkezetet, illetve ezek kezeléséhez közel álló függvényeket. Vannak esetek, mikor a felhasználótól egy esemény adatait kérjük, ezeknek helyességét vizsgáljuk. Ekkor az io.c függvényeit használjuk. A program indulásakor egy txt-fájlból betöltődnek a korábbi mentések, bezáráskor ez a fájl frissül. Ezt a filehandler.c két függvénye végzi. Mindezek mellett a debugmalloc.c külső forrásfájl ügyel a memóriaszivárgás elkerülésére, az econio.c pedig segít a konzol kezelésében.

Adatszerkezet

Az események számát előre nem tudjuk meghatározni, ezért ez egy dinamikusan bővülő méretű láncolt lista. Ezen adatszerkezet könnyen bővíthető, egyszerű belőle a törlés, eseményeink egyébként is jól tárolhatók listaszerűen. Ez struktúrákat szed listába, melyek egy-egy eseményt definiálnak.

```
typedef struct event
{
    int id;          // azonosító
    char name[101];  // név
    date eventDate; // dátum
    time eventTime; // időpont
    char place[101]; // helyszín
    char desc[101];  // megjegyzés
} event;
```

Saskó Máté

Az eseményeknek 3 szöveges(karaktertömb) változója van, ezek mérete fix. Ennek oka, hogy egy határidőnaplóban a bejegyzés nevének, helyének és a hozzá fűzött megjegyzésnek hossza ritkán több 100 karakternél, tehát ez jól behatárolható, nincs szükség a dinamikus bővülésre. Rendelkeznek továbbá egyedi id(azonosító) változóval, melynek típusa integer. Ennek frissítését a menu.c idhandler() függvénye végzi. A dátumot és az időpontot egy-egy struktúra csoportosítja, ezekben integerekben vannak tárolva az értékek, így azok kellő pontossággal megadhatók, könnyen kezelhetők.

Futtatás

A projekt buildeléséhez a GCC compiler 6.3.0-ás verziója javasolt. A .c fájlok mellett található egy include mappa, mely a .h fejléc fájlokat tartalmazza. A mappát parancssorral megnyitva a build utasítás a következő:

```
gcc econio.c menu.c event.c main.c filehandler.c io.c -o hataridonaplo.exe
```

Ez létrehozza a hataridonaplo.exe futtatható fájlt.

Külső könyvtárként az econio és a debugmalloc részei a projektnek. Az econio a konzolos megjelenítés megkönnyítéséért felel, a debugmalloc pedig segít a memóriaszivárgás kiszűrésében. A projekthez mellékelt fájlokon kívül nem szükséges más külső könyvtár.

Fájlkezelés, save.txt

A program az adatokat a save.txt-be menti. Az egyszerű kezelés érdekében az adatokat sortörés választja el egymástól. Így a több szóból álló kifejezések is könnyedén elválaszthatók és eltárolhatók. Ha egy adatmező üres, akkor a helyén csak egy sortörés áll.

Az adatok sorban:

- azonosító
- név
- dátum(év.hónap.nap)
- idő(óra:perc)
- helyszín
- megjegyzés

Hibakezelés

A hibakezelés a program több kritikus pontján is jelen van. A malloc memóriafoglalásnál le van kezelve az az eset, mikor a memóriafoglalás sikertelen.

```
new = (listItem *)malloc(sizeof(listItem));
    if (new == NULL)
        return NULL;
```

Ezen felül a fájlkezelésnél is meg van valósítva a nullpointer vizsgálata.

```
fptr = fopen("save.txt", "r");
    if (fptr == NULL)
    {
        perror("Fájl megnyitása sikertelen");
        return NULL;
    }
```

A felhasználói input dátumok és időpontok esetén vizsgálva van. A listát használó függvényeknél vizsgálva van, ha az első elemre mutató pointer esetleges NULL lenne, azaz a lista üres lenne.

Forrásfájlok részletesen

Az alábbiakban az egyes forrásfájlok részletes leírása és bemutatása olvasható.

main.c

Létrehozza a lista első elemét és meghívja a main_menu() függvényt, valamint felszabadítja a memóriát.

```
void free_memory(listItem *first);
```

Visszatérési érték nélküli függvény, mely paraméterként kéri a felszabadítandó lista első elemét és felszabadítja az összes listaelemet. Csak lokálisan elérhető.

event.c

A fájl célja, hogy definiálja az eseményt, illetve a hozzá tartozó egyéb fogalmakat, mint a dátum és az idő. Ezen felül tartalmazza a láncolt lista struktúráját és függvényeket az események kezeléséhez.

Az egyes struktúrák az alábbiak:

```
typedef struct date
{
    int year;  // év
    int month; // hónap
    int day;   // nap
} date;
```

Struktúra, mely definiál egy dátumot. Globálisan használható.

```
typedef struct time
{
    int hour;   // óra
    int minute; // perc
} time;
```

Struktúra, mely definiál egy időpontot. Globálisan használható.

```c
typedef struct event
{
    int id;         // azonosító
    char name[101];  // név
    date eventDate; // dátum
    time eventTime; // időpont
    char place[101]; // helyszín
    char desc[101];  // megjegyzés
} event;
```
Struktúra, mely definiál egy eseményt. Globálisan használható.

```c
typedef struct listItem
{
    event data;            // egy listaelem
    struct listItem *next; // következő listaelem
} listItem;
```
Struktúra a láncolt lista létrehozásához. Globálisan használható.

Az egyes függvények az alábbiak:

```c
event createEvent(char *name, date evDate, time evTime, char *plc, char
*desc);
```
Esemény típusú függvény, mely paraméterként kéri egy esemény adatait és visszaadja a belőlük kreált esemény változót. A paraméterek sorban: esemény neve (karaktertömb), esemény dátuma (date változó), esemény ideje (time változó), esemény helye (karaktertömb), megjegyzés(karaktertömb).
Globálisan használható.

```c
listItem *addListItem(listItem *first, event a);
```
Függvény, mely segítségével új elemet fűzhetünk a láncolt listánkhoz. Típusa listaelem, paraméterként kéri az első listaelemet és a hozzáfűzendő eseményt. Globálisan használható.

```c
listItem *removeEvent(listItem *first, int id);
```
Függvény, mely segítségével törölhetjük a listánk egy elemét. A törlendő elemet az id változó adja meg. Típusa listaelem, paraméterként kéri az első listaelemet és egy egész számot, mely kijelöli a törlendő elemet. Globálisan használható.

menu.c

A fájl a főmenüt és az almenüket vezérli, kezeli a felhasználó válaszait.
Az egyes függvények az alábbiak:

```c
void custom_clear();
```
Visszatérési érték nélküli függvény, melynek feladata a konzol törlése, ASCII art kiíratása. Globálisan elérhető.

```
void printEvent(event a);
```
Visszatérési érték nélküli függvény, mely kiírja a paraméterként kapott esemény adatait. Csak lokálisan elérhető.

```
listItem *idHandler(listItem *first);
```
Függvény, mely kezeli az események id-ját. A lista sorrendje szerint ad nekik egyedi értéket. Paraméterként kéri az első listaelemet és visszatér az új első listaelemmel. Csak lokálisan elérhető.

```
void printEvents_menu(listItem *first);
```
Függvény, mely az eseménykeresést vezérli. Paraméterként kéri az első listaelem pointerét, nincs visszatérési értéke. Csak lokálisan elérhető.

```
void printEventsAll(listItem *first);
```
Függvény, mely kiíratja az összes eseményt. Paraméterként kéri az első listaelem pointerét, nincs visszatérési értéke. Csak lokálisan elérhető.

```
void printEventsByMonth(listItem *first);
```
Függvény, mely kiíratja egy adott hónap eseményeit. Paraméterként kéri az első listaelem pointerét, nincs visszatérési értéke. Csak lokálisan elérhető.

```
void printEventsByWeek(listItem *first)
```
Függvény, mely kiíratja egy adott hét eseményeit. Paraméterként kéri az első listaelem pointerét, nincs visszatérési értéke. Csak lokálisan elérhető.

```
void printEventsByDay(listItem *first);
```
Függvény, mely kiíratja egy adott nap eseményeit. Paraméterként kéri az első listaelem pointerét, nincs visszatérési értéke. Csak lokálisan elérhető.

```
void findEvents(listItem *first);
```
Függvény, mely az eseménykeresést vezérli. Paraméterként kéri az első listaelem pointerét, nincs visszatérési értéke. Csak lokálisan elérhető.

```
listItem *addEvent_menu(listItem *first);
```
Függvény, mely az új esemény hozzáadásának menüjét vezérli. Paraméterként kéri az első listaelem pointerét és visszaadja az új első elemet. Csak lokálisan elérhető.

```
listItem *removeEvent_menu(listItem *first);
```
Függvény, mely az események törlésének menüjét vezérli. Paraméterként kéri az első listaelem pointerét és visszaadja az új első elemet.

```
listItem *modifyEvent_menu(listItem *first);
```
Függvény, mely az új esemény hozzáadásának menüjét vezérli. Paraméterként kéri az első listaelem pointerét és visszaadja az új első elemet. Csak lokálisan elérhető.

```
listItem *modifyEvent(listItem *first, int id);
```
Függvény, mely segítségével módosíthatjuk a listánk egy elemét. A módosítandó elemet az id változó adja meg. Típusa listaelem, paraméterként kéri az első listaelemet és egy egész számot, mely kijelöli a módosítandó elemet. Csak lokálisan használható.

```
listItem *main_menu(listItem *first);
```
Függvény, mely a főmenü kezeléséért és a felhasználói interakcióért felelős. Paraméterként kéri az első listaelem pointerét, visszatérési értéket az első listaelem. Globálisan elérhető.

filehandler.c

A fájl a program fájlkezeléséhez szükséges függvényeit tartalmazza.

```
void writeToFile(listItem *first);
```
Visszatérési érték nélküli függvény, mely elmenti a mentést tartalmazó txt-be a lista eseményeit. Ha a mentés még nem létezik, akkor létrehozza. Paraméterként kéri a lista első elemét. Globálisan elérhető.

```
listItem *readFromFile(listItem *first);
```
Függvény, mely beolvassa a mentést tartalmazó txt fájl tartalmát és hozzáfűzi a benne lévő eseményeket a láncolt listához. Ha a mentés nem létezik, nem csinál semmit. Paraméterként kéri a lista első elemét és visszaadja az új első elemet. Globálisan elérhető.

io.c

A fájl az eseményfelvételhez szükséges felhasználói interakciót valósítja meg.

```
void getName(char *name);
```
Visszatérési érték nélküli függvény, mely bekéri a felhasználótól egy esemény nevét és elmenti azt a paraméterként bekért karaktertömbben. Globálisan elérhető.

```
void getPlace(char *place);
```
Visszatérési érték nélküli függvény, mely bekéri a felhasználótól egy esemény helyszínét és elmenti azt a paraméterként bekért karaktertömbben. Globálisan elérhető.

```
void getDate(date *el);
```
Visszatérési érték nélküli függvény, mely bekéri a felhasználótól egy esemény dátumát és elmenti azt a paraméterként bekért dátum változóban. Globálisan elérhető.

```
void getTime(time *el);
```
Visszatérési érték nélküli függvény, mely bekéri a felhasználótól egy esemény idejét és elmenti azt a paraméterként bekért idő változóban. Globálisan elérhető.

```
void getDesc(char *desc);
```
Visszatérési érték nélküli függvény, mely bekéri a felhasználótól az eseményhez fűzött megjegyzést és elmenti azt a paraméterként bekért karaktertömbben. Globálisan elérhető.