**Határidőnapló programozói dokumentáció**

Készítette: Áron Gergő

**Adatszerkezet**

Esemény:

A program központi adatszerkezete, mely tartalmazza az esemény leíró adatait és láncolt lista felépítéséhez használható következő eseményre mutató pointert.

Az ev, ho, nap, név, helyszin,megj adatokat a felhasználó adja meg. A het és a hetnapja adatot nem a felhasználótól kéri be a program, hanem a megadott adatokból számolja ki.

typedef struct Esemeny {

**int** ev, ho, nap, het, hetnapja;

**int** ora, perc;

**char**\* helyszin;

**char**\* nev ;

**char**\* megj;

**struct** Esemeny\* kov;

}Esemeny;

**Modulok**

**Main modul**

A program vezérlését biztosító fő logikát tartalmazza. Egy ciklus, amely a felhasználó válaszától függően végrehajtja az adott menüponthoz tartozó műveletet. A menüben történő közlekedés logikája itt található meg. A fő modul által meghívott függvények a többi modulban találhatók meg. Addig fut, amíg a bemenet nem 0.

**Dátum kezelés modul**

A dátumok kezeléshez szükséges funkciókat tartalmazza.

**bool** szokoev(**int** ev)

Megadja, hogy a paraméterben átadott év szökőév-e. Szökőév esetén **true**, egyébként **false** a visszatérési érték.

**int** hanyadik\_nap(**int** ev, **int** ho, **int** nap);

Megadja, hogy a paraméterben átadott dátum az adott év hányadik napja.

A számozás 1-gyel indul. a **szokoev** függvényt használja arra, hogy szökőév esetén is pontosan működjön. Erre a lokális februar változót használja a függvény, aminek deklarálásnál 28 értéket ad, de ha a **szokoev(ev)** igaz értékkel tér vissza, 29 értéket kap. A napszámlálás megvalósításához a függvény egy switch case-t használ a *ho* paraméterre. A visszatérési értéke az a szám, ahányadik napja a paraméterként kapott dátum az évnek.

**void** hanyadik\_het(**int** ev, **int** ho, **int** nap, **int** \*hanyadikhet, **int** \*milyennap)

Megadja, hogy a paraméterben átadott dátum az adott és hányadik hetére esik. A számozás 1-gyel indul (első hét).

Megadja továbbá, hogy az adott nap a hét hányadik napja.

A számozás 0-val indul.

0 - hétfő

1 — kedd

…

6 - vasárnap

A *napdb* lokális változóban eltárolja a **hanyadik\_nap** függvény segítségével, hogy hányadik napja a dátum az évnek. A *het* lokális változóban, aminek 1 a kezdőértéke, fogja a heteket számolni. Viszonlyag komplikált a hét napjait számolni, mivel az egymást követő évek nem ugyanazzal a nappal kezdődnek. Ezért kiindulási pontnak a függvény 2018-at veszi, aminek hétfő az első napja.

Az évek és az első napjaikról azt kell tudni, hogy normál év után egy nappal csúszik a következő és első napja (2018 - hétfő -> 2019 – kedd). Szökőévek után pedig kettőt.

így két ciklus segítségével ki tudjuk deríteni, hogy a paraméterként kapott év milyen nappal kezdődik.

Ezután a *napdb* segítségével ki tudjuk számolni, hogy a dátumunk milyen napra és az év hányadik hetére esik. Ezután ezt a két adatot visszaadja a függvény a paraméterként kapott pointerekbe.

**int** napkiir(Esemeny \*lista, **int** ev, **int** ho, **int** nap);

Soronként kiírja az ev/ho/nap paraméterben megadott napra eső eseményeket a lista elemei közül.

Visszatérési érték a találatok száma.

**int** hokiir(Esemeny \*lista, **int** ev, **int** ho);

Soronként kiírja az ev/ho paraméterben megadott hónapra eső eseményeket a lista elemei közül.

Visszatérési érték a találatok száma.

**int** hetkiir(Esemeny \*lista, **int** ev, **int** het);

Soronként kiírja az ev/het paraméterben megadott év adott hetére eső eseményeket a lista elemei közül.

Visszatérési érték a találatok száma.

**Eseménykezelés modul**

Az esemény struktúrához kötődő függvényeket tartalmazza

**void** esemeny\_init(Esemeny \*p)

Inicializálja a paraméterként kapott listaelemet. A szám adatok értékét 0-ra, a pointerekét NULL-ra állítja.

**void** esemeny\_destroy(Esemeny \*e)

Elpusztítja a paraméterként kapott listaelemet. Ha az adatra mutató pointerek nem nullpointerek, felszabadítja azt a memóriát, amire mutatnak, majd felszabadítja magát a listaelemet.

**void** felszabadit(Esemeny \*lista)

Felszabadítja a paraméterként kapott láncolt listát (a lista első elemére mutató pointer a paraméter). Erre az **esemeny\_destroy** függvényt használja.

**void** esemeny\_kiir(Esemeny \*e)

A paraméterben átadott listaelem adatait írja ki megfelelően formázva.

A kiírás a sor végén soremelést is tartalmaz.

**int \***keres(Esemeny \*eleje, char \*mit)

A paraméterként megadott láncolt listában keresi a paraméterben megadott szöveget az események nevei között. A találatok indexeit (hányadik eleme a listának) egy dinamikus tömbben adja vissza, aminek az utolsó eleme -1. Ha nincs találat, NULL pointert ad vissza.

**int** talalat\_kiir(Esemeny \*eleje, int \*tomb)

Paraméterként egy láncolt listát kap és egy dinamikus tömböt, aminek formátuma megegyezik a **keres** függvény visszatérési értékével, azaz az utolsó eleme -1. A föggvény végigmeg a listán, és kiírja azokat az elemeket, amiknek az indexe megegyezik a tömbben kapott indexek egyikével. A függvény számolja, hogy hány eseményt írt ki, azaz, hogy hány találat van, és ezt a számot adja vissza.

**void** teljeskiir(Esemeny \*lista)

A lista paraméterben átadott eseményeket listázza ki.

Ha a lista üres, akkor az “Üres lista!” üzenet jelenik meg.

**char** \*beolvas(**void**);

Beolvas egy tetszőleges hosszúságú szöveget, amit dinamikus sztringben tárol, és egy erre mutató pointert ad vissza. Sikertelen memóriafoglalás esetén ezt hibaüzenettel jelzi és NULL pointerrel tér vissza.

**void** esemeny\_beolvas(Esemeny \*mibe);

A paraméterben átadott ‘mibe’ eseménybe beolvassa egy esemény adatait.

a szöveges adatokhoz a **beolvas** függvényt használja. Ezután a **hanyadik\_het** függvénnyel kiszámolja a *het* és a *hetnapja* adatokat.

**bool** esemeny\_igazol(Esemeny \*e);

Ellenőrzi a paraméterként kapott listaelemről, hogy megfelelő adatokat tartalmaz-e. Amennyiben bármelyik adat hibás, **false** értékkel tér vissza. Ha nem talál hibát, **true** értékkel tér vissza.

Esemeny \*esemeny\_hozzafuz(Esemeny\* lista);

Bekér a felhasználótól egy eseményt, majd, ha nem talál hibát az adatokban, a paraméterként kapott listához fűzi. Amennyiben NULL pointert kap, létrehozza a láncolt lista első elemét. Ezekhez az **esemeny\_init, esemeny, beolvas, esemeny\_igazol** és **esemeny destroy** függvényeket használja.

**Fájlkezelés modul**

Ezek a függvények a fájba történő kiíráshoz, és a fájlból történő beolvasáshoz szükségekes.

**char** \*szovegfajlbeolvas(FILE \*fp)

Beolvas egy tetszőleges hosszúságú szöveget a paraméterként kapott fájlból, majd egy dinamikus sztringben tárolja azt. Egy erre mutató pointer a visszatérési értéke.

**void** fajlbol\_beolvas(Esemeny \*mibe, FILE \*fp)

Beolvas egy eseményt a paraméterként kapott fájlból a paraméterként kapott eseménybe. Erre a **szovegfajlbeolvas** és a **hanyadik\_het** függvényeket használja.

Esemeny \*fajlbol\_hozzafuz(Esemeny \*lista, FILE \*fp)

A paraméterként kapott fájlból beolvas egy eseményt a **fajlbol\_beolvas** függvény segítségével, majd a paraméterként kapott lista végére helyezi.

Esemeny \*fajlbol\_keszit(Esemeny \*lista, FILE \*fp)

Létrehoz egy láncolt listát a paraméterként kapott fájlból. A visszatérési értéke a beolvasott láncolt lista.

void fajlba\_kiir(Esemeny \*e, FILE \*fp)

kiírja a paraméterként kapott fájlba a paraméterként kapott láncolt listát. Az első sorba azt írja, hogy hány eseményt tartalmaz a lista.

**Menükezelés modul**

**int** fomenu()

A program főmenüjének alapja. Kiírja a főmenüből elérhető opciókat, majd addig kér be egy számot a felhasználótól, amíg nem kap érvényes bemenetet. Visszatérési értéke a felhasználó által kiválasztott parancs sorszáma.