VASS CSONGOR - AHQQ3A

Programozás alapjai 2.

Barkochba

2020. 05. 17.

Tartalom

1. Feladat rövid leírása

Egy bináris fában tárolt barkochba játék, melyben a csomópontok a kérdések, a levelek a rákérdezések. Hibás válasz esetén a játékos megadhat új kérdést hozzá tartozó válaszokkal, így a program tanul. Fájlba menthető és abból kiolvasható az aktuális állapot.

2. Osztályok

String

A String alaposztály fogja a szövegkezelést végezni, ebben tároljuk majd a kérdéseket. „+” és „=” operátorokat definiálunk a stringekkel való műveletekhez, állítunk be gettereket és konstruktorokat, valamint gondoskodunk a memória felszabadításáról.

BinFa

Sablonnal valósítjuk meg, ez praktikus lehet, ha más osztállyal szeretnénk majd a barkóbát játszani. Jelenleg majd a *BinFaElem*-ek String osztályúak lesznek, örökölve annak tulajdonságait. Egy elem tartalmazza a saját kérdését, és a tőle jobbra illetve balra irányuló elemek pointerjét. A BinFa tárolja a gyökerére és az aktuális elemre mutató pointert. Függvényeivel változtathatók az aktuális adatai, meg tudja mondani, hogy levélnél vagyunk-e, illetve felszabadítják a dinamikusan foglalt területet.

Serializable

Fájlba mentéshez létrehozunk egy Serializable osztályt, amiben szerepelnek virtuális függvényként a fájlba írás, fájlból olvasás a destruktora.

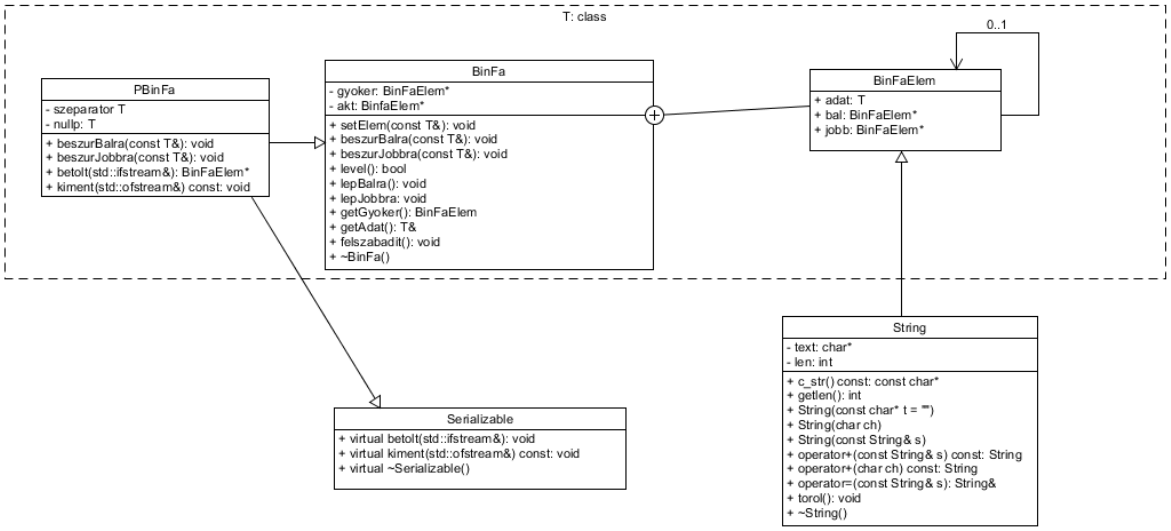
PBinFa

A BiNFa és a Serializable keresztezésével létrehozzuk a PBinFa osztályt, ami egy fájlba menthető és onnan betölthető bináris fa. In-order bejárást használunk, mentésnél a NULL pointereket # jellel jelöljük majd.

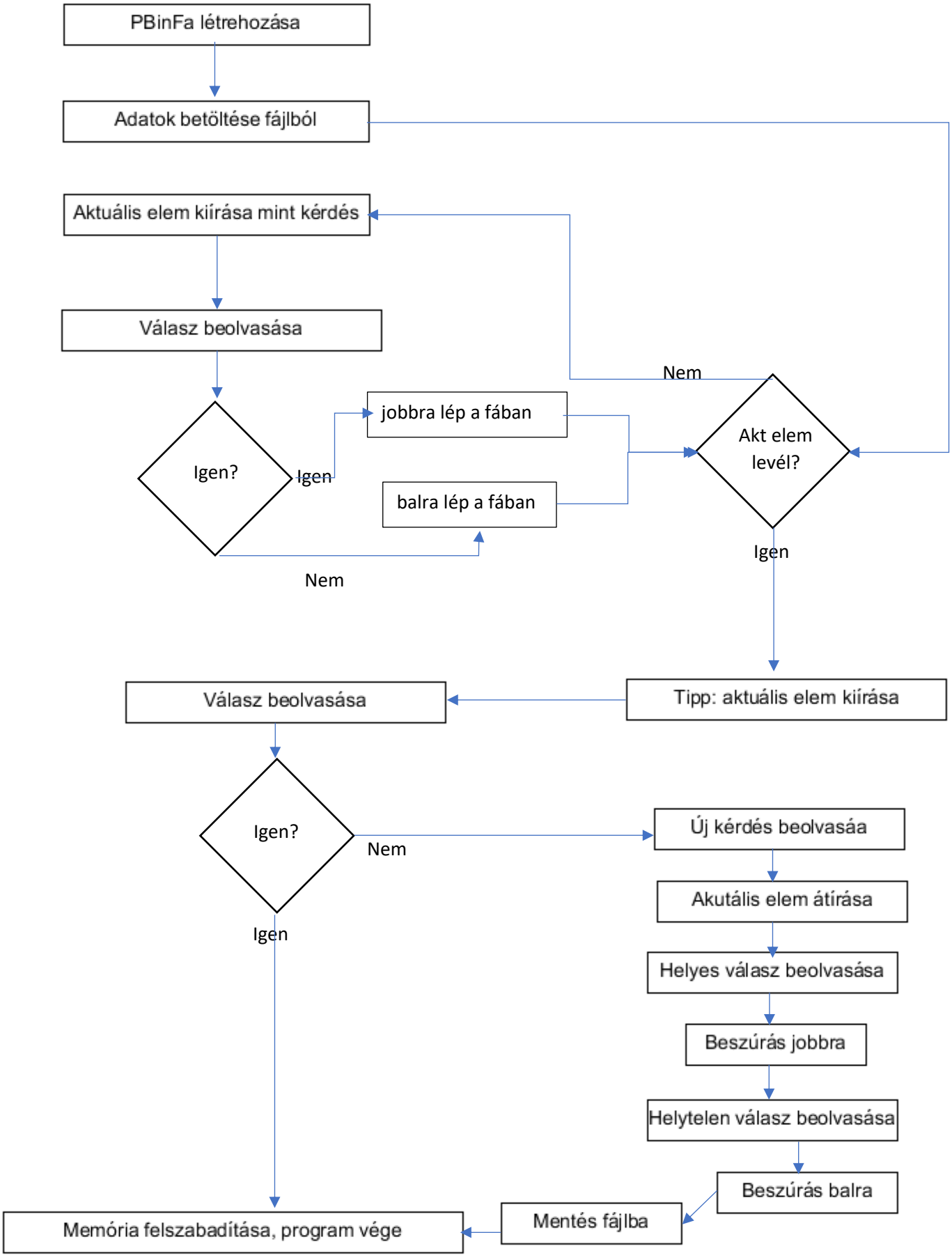
3. Egyéb függvények

A „==”, „!=”, „<<” és „>>” operátorokat definiáljuk String osztályhoz.

4. UMLT ábra


T: class
0..1
PBinFa
- szeparator T
- nullp: T
+ beszurBalra(const T&): void
+ beszurJobbra(const T&): void
+ betolt(std::ifstream&): BinFaElem*
+ kiment(std::ofstream&) const: void
BinFa
- gyoker: BinFaElem*
- akt: BinfaElem*
+ setElem(const T&): void
+ beszurBalra(const T&): void
+ beszurJobbra(const T&): void
+ level(): bool
+ lepBalra(): void
+ lepJobbra: void
+ getGyoker(): BinFaElem
+ getAdat(): T&
+ felszabadit(): void
+ ~BinFa()
BinFaElem
+ adat: T
+ bal: BinFaElem*
+ jobb: BinFaElem*
Serializable
+ virtual betolt(std::ifstream&): void
+ virtual kiment(std::ofstream&) const: void
+ virtual ~Serializable()
String
- text: char*
- len: int
+ c_str() const: const char*
+ getlen(): int
+ String(const char* t = "")
+ String(char ch)
+ String(const String& s)
+ operator+(const String& s) const: String
+ operator+(char ch) const: String
+ operator=(const String& s): String&
+ torol(): void
+ ~String()

5. Algrotimus



6. Osztályok és tagfüggvényeik részletes megvalósítása

a. String

Szöveges adatok tárolására alkalmas osztály. Dinamikus területen tárolja a karaktereket és a karaktersorozat hosszát is jegyzi.

Tagfüggvények:

- String(const char* t = "") *konstruktor C sztringből, paraméter nélkül üres string*
- String(char ch) *konstruktor karakterből*
- String(const String& s) *másoló konstruktor*
- const char* c_str() const *a szöveg elejére mutató pointert ad vissza*
- int getlen() *a szöveg hosszát adja vissza*
- String operator+(const String& s) const *stringhez stringet fűz*
- String operator+(char ch) const *stringhez karaktert fűz*
- String& operator=(const String& s) *értékadó operátor*
- void torol() *felszabadítja a lefoglalt memóriaterületet*
- ~String() *destruktor*

Globálisan definiált függvények stringhez:

- std::ostream& operator<<(std::ostream& os, const String& s) *ostreamre írás*
- std::istream& operator>>(std::istream& is, String& s) *istreamről olvasás*
- bool operator==(const String&, const char*) egyenlőség operátor C stringgel
- bool operator==(const String&, const String&) egyenlőség operátor Stringgel
- bool operator!=(const String&, const char*) *nem egyenlő operátor*

b. BinFa

Generikus bináris fa, melyet tetszőleges adattípussal használhatunk. A csomópontok **BinFaElem** struktúrák, amelyek tartalmaznak egy megfelelő típusú adatot, két irányba pointereket. Paraméter nélküli konstruktora a két pointert NULL értékűre állítja. A BinFa osztály tárolja a legelejére mutató gyoker pointert, és az aktuális helyzetet megadó akt pointert, melyem BinFaElem típusúak.

Tagfüggvények:

- BinFa() *konstruktor, létrehoz egy új BinFaElem-et és az akt-ot rá állítja*
- void setElem(const T& p) *aktuális elem adatát állítja át*
- void beszurBalra(const T&, BinFaElem<T>*) *új elem létrehozása és aktuális elem bal pointerének ráállítása*
- void beszurBalra(const T& dat) *egyparaméteres beszurBalra*
- void beszurJobbra(const T&, BinFaElem<T>*) *beszurBalra párja jobb oldallal*
- void beszurJobbra(const T& dat) *egyparaméteres beszurJobbra*
- bool level() *visszaadja, hogy levélre mutat-e az akt pointer*
- void lepBalra() *akt pointer balra léptetése*
- void lepJobbra() *akt pointer jobbra léptetése*
- BinFaElem<T>* getGyoker() *visszaadja a binfa legelejét*
- T& getAdat() *visszaadja azt aktuális elemet*
- void felszabadit(BinFaElem<T> *) *felszabadítja a lefoglalt területeket (in oreder)*
- ~BinFa() *destruktor*

c. PBinFa

Perzisztens bináris fa, mely örököl a generikus BinFa osztálytól és a Serializable osztálytól. Képes egy streamre kimenteni az adatait, valamint egy streamről felépíteni magát. Tárol két felépítéshez szükséges elválasztót (szeparator, nullp), amit a konstruktorban lehet megadni. Ha a kívánt típus nem kezel const char* típusú adatokat, akkor kötelező ezeket megadni.

Tagfüggvények:

- PBinFa(T sz = "\n", T n = "#") *konstruktor, létrehoz egy binfát és beállítja a szeparator és nullp értékeit (ha nincs megadva, akkor „\n” és „#” az alapértelmezett*
- void kiment(std::ostream&, BinFaElem<T> *) *kiírja az adatokat in order sorrendben egy adatfolyamra*
- bool betolt(std::istream&, BinFaElem<T> *) *betölti az adatokat egy adatfolyamról in order sorrendben, minden adat után a szeparator karaktert írja, nullpointer esetén a nullp karaktert*
- void kiment(std::ostream& os) *egyparaméteres kiment*
- void betolt(std::istream& is) *egyparaméteres betölt*

d. Serializable

Perzisztens adatok tárolásához használt struktúra, mely virtuális függvényeket tartalmaz.

Tagfüggvények:

- virtual void write(std::ostream &) const *virtuális adatfolyamra írás*
- virtual void read(std::istream &) *virtuális adatfolyamról olvasás*
- virtual ~Serializable() *virtuális destruktor*

7. Tesztesetek, a program futása

Négy teszteset fut, melyből az első három az osztályok tesztelését végzi, minden függvényt meghív legalább egyszer, a negyedik pedig a tényleges játék futása.

a. teszt_String()

```
String osztaly fuggvenyeinek tesztelese

Parameter nelkuli konstruktor:
Karakterbol string: s
C sztringbol String: teszt string
String hossza: 12
Masolo konstruktor: teszt string
A masolt hozza: 12
Stringek osszefuzese: teszt stringteszt string Hossza: 24
Stringhez karakter fuzese: sz Hossza: 2
Ertekado operator: sz Hossza: 2
Beolvasas: sz
Beolvasott: sz Merete: 2
Egyenloseg operatorok teszelese:
A beolvasott szoveg nem: teszt
A ket sztring egyezo
```

b. teszt_BinFa()

```
BinFa osztaly tesztelese:
String binfa letrehozva
Double binfa letrehozva
String beallitva adatnak: Teszt
Double bealltiva adatnak: 3.14
String beszurva balra
A bal oldalt levo string: Teszt bal
Felepitunk egy valos szamokat tarolo binfat
Ezzel teszteltuk a jobbra beszurast is.
```

c. teszt_PBinFa()

```
Letrehoztunk egy Stringeket tartalmazo binfat
Kimentjuk a BinFat egy fajlba
A kimentett adatok: teszt
tesztbal
#
tesztbaljobb
#
#
tesztjobb
#
#
Visszatoltjuk a fajlbol egy masik faba
A beolvasott fa elemei: teszt
tesztbal
#
tesztbaljobb
#
#
tesztjobb
#
#
```

d. teszt_jatek()

```
Eloleny? (I/N)
I
Allat? (I/N)
I
Nyavog? (I/N)
N
Vizben el? (I/N)
nem
I-t vagy N-t irj be!
Vizben el? (I/N)
N
Fejik? (I/N)
N
Van szárnya? (I/N)
N
Tipp: Madár (I/N)
N
Segits fejlodnom! Adj meg egy uj kerdest a tippem helyett es a hozza tartozo valszokat!
Kerdes: Bogar?
Igen: Pok
Nem: Madar
Koszonom a segitseget
```

```
Eloleny? (I/N)
I
Allat? (I/N)
I
Nyavog? (I/N)
N
Vizben el? (I/N)
N
Fejik? (I/N)
N
Van szárnya? (I/N)
N
Bogar? (I/N)
I
Tipp: Pok (I/N)
I
Koszonom a jatekot!
```