

Adatszerkezetek

01. Szeminárium

Vekov Géza

2023. február 27.



Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Projekt

A projektre vonatkozó általános követelmények

- Specifikálják és implementálják a projektként kapott AAT-t.
- Találjanak ki egy érdekes (egyedi) alkalmazást, amely fölhasználja az implementált AAT-t.

A projekt szerkezete

- Prezentáció (PowerPoint/ Prezi vagy PDF)
- A projekt kivonatolt bemutatása (lásd a következő táblázatot)
- Implementáció C/C++
- Egy alprogram = egy funkció, komment minden fejléc alatt és bárhol szükséges
- Barátságos interfész
- A felhasználó program legyen független az AAT implementációjától
- Tesztállományok - legkevesebb 5.

A projektre vonatkozó általános követelmények 2

- A példaprojektnek megfelelően kell strukturálni a beküldendő állományokat:
 - **ASz mappa:** az adatszerkezet header állománya és a definíciós állomány (külön!), specifikáció dokumentuma
 - **Bemutató mappa:** a bemutatót tartalmazza
 - **Feladat mappa:** tartalmazza a feladat specifikációját (.doc, .docx, .pdf formátumban), és az *implementált feladat* mappában a teljes működő alkalmazást, legalább 5 tesztállománnyal (ki és bemenet)
- A feltöltendő állomány: **CsNev_KNev_Téma.zip**

Feltöltés, határidők

- A megbeszélrt, és táblázatban látható határidő előtt fel kell tölteni a projektet.
- 1 hét késés: max 8-as projektjegy
- 2 hét késés: max 6-os projektjegy
- Ezt követően csak pótszesszióban lehet bemutatni max 5-ösért.

Projekt

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Fontos

Bármely témát ki lehet cserélni önállóan kiválasztott (a listában nem szereplő) *AAT*-sal, amit előbb egyeztetünk!

Témák1.

- Egydimenziós tömb
- Kétdimenziós tömb
- Háromszögű mátrix
- Ritka tömb
- Polinom (mindkét ábrázolás)
- Statikus verem
- Dinamikus verem
- Statikus várakozási sor
- Dinamikus várakozási sor
- Dinamikus várakozási sor strázsákkal
- Dinamikus rendezett lista
- Dinamikus rendezett lista strázsákkal
- Duplán láncolt lista
- Körkörös lista

Témák 2.

- Tökéletesen egyensúlyozott bináris fa (dinamikusan)
- Bináris keresőfa (iteratíván)
- Bináris keresőfa (rekurzívan)
- Piros-fekete keresőfa
- Bináris kupac
- Elsőbbségi sor
- Splay-fa
- Karakterlánc (Rabin-Karp)
- Karakterlánc (KMP)
- Karakterlánc (Boyer-Moore)
- Binárisan indexelt fák
- Intervallumkupacok (jegyzet)
- Binomiális kupacok (Jegyzet)
- Min-Max kupacok (Jegyzet)
- Diszjunkt halmazok (fákkal) (Jegyzet)

Témák 3.

- Hasító tábla nyílt címzés, lineáris kipróbálás
- Hasító tábla nyílt címzés, négyzetes kipróbálás
- Hasító tábla nyílt címzés, bővítéssel
- Hasító tábla listával
- Hasító tábla dupla hasítás
- Quadfák
- Trie (karakterlánc)
- AVL-fa

Projekt elvárások

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Kategória	Tételek	Megjegyzések	Pontszám
Adatszerkezet	Helyesség		30
	Megfelelőség	Fejlécek paraméterezése	
	Hatékonyság		
	Programozási stílus	Metódusok, paraméterek dokumentálása, stílus	
Alkalmazás	Feladat kijelentése	Megoldandó feladat, bemenet, kimenet, példa, megszorítások	30
	Programozási stílus	metódusok, paraméterek dokum., stílus	
	Hatékonyság	Mennyire felel meg az adatszerkezet, működés	
	Tesztelés	Terv, eredmények, szélsőséges esetek tárgyalása	
Prezentáció	AAT		30
	Implementáció		
	Feladat bem		
	Alkalmazás		
	Indoklás	Alkamazás és adatszerkezet kapcsolata	
	Forma		
	Előadásmód		

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Zászlók

Feladat

Adottak egy nemzeti zászló színei. Írjunk alkalmazást, amely

- eldönti, hogy a zászló csak alapszíneket tartalmaz-e vagy sem;
- a kevert színeket alkotószíneire bontja;
- ha "idegen" színt talál, kiír egy megfelelő üzenetet;
- ha a zászló csak kevertszíneket tartalmaz, a program kiír egy megfelelő üzenetet.

Elemzés

- A három alapszín (elsőrendű): **piros, sárga, kék**
- Azok a színek, amelyek nem alapszínek: fehér, fekete és a kevertszínek, amelyek a három alapszín keverékeként jönnek létre: piros + sárga = **narancssárga**, piros + kék = **lila**, sárga + kék = **zöld**

Színek

Egyszerű adattípus

- Típus neve: **szín**.
- Lehetséges **értékek** felsorolása.
 - Az absztrakt adattípusunk értékei színek: **piros**, **sárga**, **kék**, valamint azok a másodrendű színek, amelyeket ezeknek a keveréke alkot: **narancssárga**, **lila**, **zöld**
- Lehetséges **műveletek** felsorolása.

Specifikáció - Egyszerű absztrakt adattípus: **szín**

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Egyszerű absztrakt adattípus: **szín**

Értéktartomány: piros, sárga, kék, zöld, narancssárga, lila

Műveletek:

Beolvas: beolvassa az *sz* színt

előfeltételek: -

utófeltételek: ha a beolvasott *szín* helyes adat, az érték *sz*-ben tárolódik, ha nem helyes, ezt egy hibajelző paraméter értéke tárolja

Kiír: kiírja az *sz* színt

előfeltételek: -

utófeltételek: kiírja az *sz*-ben tárolt színt

Specifikáció - Egyszerű absztrakt adattípus: **szín**

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Elsőrendű-e? Ellenőrzi, hogy az adott *sz* szín elsőrendű szín-e vagy sem

előfeltételek: -

utófeltételek: ha *sz* elsőrendű szín, akkor **true**-t térít, különben **false**-t

Kevert-e? Ellenőrzi, hogy az adott *sz* szín kevert szín-e vagy sem

előfeltételek: -

utófeltételek: ha *sz* kevert szín, akkor **true**-t térít, különben **false**-t

Specifikáció - Egyszerű absztrakt adattípus: **szín**

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Bont: Megállapítja, hogy mely színek alkotják az sz színt

előfeltételek: sz_1 nem elsőrendű szín

utófeltételek: sz_1 és sz_2 azok az elsőrendű színek, amelyeknek keverékéből létrejön az sz szín

Kever: Megállapítja azt az sz kevert színt, amely akkor jön létre, ha az sz_1 színt összekeverjük az sz_2 színnel

előfeltételek: sz_1 és sz_2 elsőrendű színek

utófeltételek: ha sz_1 és sz_2 egyenlők, sz elsőrendű szín, különben kevertszín

Implementálás

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Hogyan ábrázoljuk a *szín* típust?

- A műveletek implementációja legyen minél **egyszerűbb!**
- Az alkalmazás interfésze legyen minél **barátságosabb!**
- A kimenet legyen **egyértelmű** (könnyen érthető/értelmezhető)!

Lehetne tömb:

- Hátrány: csak tömbelemeket "látunk".
- Kódolhatnánk számjegyekkel, de nem látjuk a tulajdonképpeni színt, csak egy egy sorszámot.

Lehetne felsorolás:

- Előny: látjuk a szín nevét.
- Aritmetikai műveleteket is végezhetünk a színekkel.

Saját műveletek felhasználása!

- A **Bont** művelet előfeltétele: a művelet akkor értelmezett ha az *sz* szín nem elsőrendű szín.
- \Rightarrow létezik a művelet, amely ezt a tulajdonságot ellenőrzi: **Elsőrendű-e?**

Megjegyzések

- Ha megengednénk az alkotószínekre bontást alapszínek esetében is, akkor ezt külön kezelniünk kellene, mert a két alkotószín azonos lenne.
- Ha megengednénk a keverést két azonos alapszín esetében is, akkor ezt is külön kezelniünk kellene: a keverés eredménye elsőrendű szín lenne.

Ha módosítjuk a **Bont** specifikációját:

Bont: Megállapítja, hogy mely színek alkotják az sz színt

előfeltételek: -

utófeltételek: ha sz elsőrendű szín, akkor $sz_1 = sz_2 = sz$, különben sz_1 és sz_2 azok az elsőrendű színek, amelyeknek keverékéből létrejön az sz szín

Fontos

- Annak eldöntése, hogy melyik specifikációt fogjuk használni attól függ, hogy hogyan van megszerkesztve az alkalmazás.
- Ez a döntés a programterve ző kizárólagos feladata.
- Megírjuk a programot vigyázva arra, hogy a program szigorúan megfeleljen ezeknek a specifikációknak.

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Betűsor

Specifikáció - Összetett absztrakt típus: **betűsor**

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Összetett absztrakt típus: **betűsor**

Elemek: betűk ('A'-tól 'Z'-ig és 'a'-tól 'z'-ig) + szóköz **Szerkezet:** lineáris (a betűk egymás után vannak elhelyezve) **Értéktartomány:** 0 és 80 közötti hosszúságú betűsorok
Műveletek:

Beolvasás: beolvassa az *s* betűsort

előfeltételek: -

utófeltételek: létrejön az *s* betűsor

Kiírás: kiírja az *s* betűsort

előfeltételek: -

utófeltételek: az *s* betűsor kiíródik a képernyőre

Specifikáció - Összetett absztrakt típus: **betűsor**

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Az első szó kiírása és törlése: leválasztja, és kiírja az s első szavát

előfeltételek: az s betűsor nem üres (hossza nagyobb, mint 0)

utófeltételek: megadja az s betűsor első szavát, kiírja és téríti az új betűsört, amely ugyanaz, mint a művelet végrehajtása előtt, de hiányzik belőle az első szó

Az utolsó szó kiírása és törlése: leválasztja, és kiírja az s utolsó szavát

előfeltételek: az s betűsor nem üres (hossza nagyobb, mint 0)

utófeltételek: megadja az s betűsor utolsó szavát, kiírja és téríti az új betűsört, amely ugyanaz, mint a művelet végrehajtása előtt, de hiányzik belőle az utolsó szó

Betűsor szavakra bontása: szavakra bontja az s betűsört

előfeltételek: a betűk száma az s betűsorban nagyobb mint 0

utófeltételek: megadja az s betűsört alkotó szavakat

Specifikáció - Összetett absztrakt típus: **betűsor**

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Betű ragasztása a betűsor végére: az s betűsorhoz jobbról hozzáragasztja a b betűt

előfeltételek: a betűk száma a bemeneti betűsorban kisebb, mint 80

utófeltételek: az s hossza 1-gyel nő, és a b betű a betűsor utolsó (legjobbra eső) eleme

Betű törlése a betűsor elejéről: az s (betűsor típusú) adatból kiveszi és visszatéríti balról az első betűt

előfeltételek: a betűk száma a bemeneti betűsorban nagyobb, mint 0

utófeltételek: megadja az s betűsor balról első betűjét, és téríti az új betűsort, amelyből hiányzik az első betű

Specifikáció - Összetett absztrakt típus: **betűsor**

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Üres: eldönti, hogy az *s* betűsor üres-e

előfeltételek: -

utófeltételek: ha *s* elemeinek száma 0, a válasz értéke *true*, különben *false*

Tele: eldönti, hogy az *s* betűsor hossza egyenlő-e 80-nal

előfeltételek: -

utófeltételek: ha *s* elemeinek száma 80, a válasz értéke *true*, különben *false*

Fordít: megfordítja az *s* betűsort

előfeltételek: -

utófeltételek: az *s* betűsor elemei fordított sorrendben kerülnek az új *s* betűsorba

Implementálás: **betűsor**

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Ábrázolás

Olyan típusokat használunk fel, amelyeket a programozási nyelv ismer (**előre deklarált típusok**)

```
struct betusor{  
    int n;           // A betűsor aktuális hossza  
    char str[80];    // a betűsor  
};
```

Az ábrázolás **saját típus**.

Implementálás: **betűsor**

Adatszerkezetek

Vekov Géza

Projekt

Zászlók

Betűsor

Implementálás

- Modulárisan tervezzük az adatszerkezet implementációját.
- A "felhasználó" programnak nem szabad látnia az ábrázolást illetve az implementációs részleteket.
- Általában csak a header-állomány (deklarációk) látható az implementációban a "felhasználó" számára.