A programozás alapjai 2.

Házi Feladat Dokumentáció

**NBA csapatépítő**

Készítette: Baranyai Ármin,UR8BDG

baranyaiarmin2003@gmail.com

Készítés féléve: 2022/23/2

Tartalomjegyzék

[Dokumentációval kapcsolatos teendők 3](#_Toc481761418)

[Felhasználói dokumentáció 4](#_Toc481761419)

[Osztályok statikus leírása 4](#_Toc481761420)

[Osztály1 (Példa: Triangle) 4](#_Toc481761421)

[Felelőssége 4](#_Toc481761422)

[Ősosztályok 4](#_Toc481761423)

[Attribútumok 4](#_Toc481761424)

[Metódusok 4](#_Toc481761425)

[UML osztálydiagramm 5](#_Toc481761426)

[Összegzés 6](#_Toc481761427)

[Mit sikerült és mit nem sikerült megvalósítani a specifikációból? 6](#_Toc481761428)

[Mit tanultál a megvalósítás során? 6](#_Toc481761429)

[Továbbfejlesztési lehetőségek 6](#_Toc481761430)

[Képernyőképek a futó alkalmazásról 7](#_Toc481761431)

# Felhasználói dokumentáció

A program egy NBA csapat NBA rájátszásban (NBA playoffs) való játszatásáról szól. A program először megkérdezi a felhasználótól, hogy össze akar-e egy csapatot rakni, vagy választani szeretne a létező csapatok közül (yes/no bemenetet vár, de működik y/n vagy YES/NO – val is, ha nem megfelelő a bemenet a program újra kérdezi a felhasználót)

Innen ketté válik a program, ha az előző kérdésre nemmel válaszoltunk akkor választanunk kell egy csapatot (pl. San Antonio Spurs) (itt mint majd a program is írja a help bemenetre kilistázza nekünk a választható csapatokat).

Igen válasz esetén megkérdezi, minek szeretnénk elnevezni a csapatunkat (itt igazából akármit megadhatunk pl. Shanghai Sharks ), majd játékos neveket kér. 5 játékost kell a csapatunkba rakni, minden pozícióból egyet, a help inputra kilistázza nekünk a program a lehetséges help kommandokat a különböző posztokhoz (a program egész neveket vár, és csak akkor fogadja el a bemenetet, ha pontosan (tehát megfelelően nagy-kisbetűkkel írva) adjuk meg neki a játékosneveket

pghelp – Point Guard (pl. Stephen Curry)

sghelp – Shooting Guard (pl. Kyrie Irving)

sfhelp – Small Forward (pl. Jaylen Brown)

pfhelp – Power Forward (pl. LeBron James)

chelp – Center (pl. Nikola Jokic)

A program figyel arra is, hogy egy pozícióból ne lehessen több játékos a csapatban.

miután megépítettük a csapatunkat/kiválasztottunk egy csapatot, a program NBA rájátszásba rakja elkészített/választott csapatunkat NBA rájátszáshoz megfelelően először 16 (nyolcaddöntőben) (15 + a saját csapatunk)csapat indul, ebből egy random kiválasztott csapattal kell versenyeznünk (itt kapunk egy százalékot, amennyi esélyünk van megnyerni az adott meccset, ez a csapatban lévő játékosok lineups.com szerinti rating-től függ) a program megkéri minden meccs után enter-t üss, hogy ne egyszerre játszódjon le a 7 meccs. 4 meccs megnyerése után (attól függően, hogy mi nyertük a 4 meccset vagy az ellenfél) továbbjutunk a negyeddöntőbe, vagy kiesünk. A program ismét lejátssza a meccseket, és vagy bekerülünk az elődöntőbe, vagy kiesünk. Ha továbbjutottunk ugyanez történik, és vagy döntőbe kerülünk, vagy kiesünk . A döntőt megnyerve gratulál neknk a program, és kilistázza a csapat tagjait. (ha a döntőben vesztünk szintén csak kiesünk, és a program kiírja ki ellen vesztettünk).

# Osztályok statikus leírása

## Player (csak a heterogén kollekció miatt kell ez az osztály)

### Felelőssége

Lesz a játékosnak neve, csapata, magassága, életkora, és lineups.com szerinti rating-je

Pár getter függvény található, és pár virtuális függvény.

### Ősosztályok

-

### Attribútumok

#### Privát

string name, teamName;

int height, age, rating; (előbb leírtak szerint működik)

#### Védett

-

#### Publikus

-

### Metódusok

#### Privát

-

#### Védett

-

#### Publikus

* konstruktorok
* getter függvények
* toString() kiírató függvény

## PointGuard, ShootingGuard, SmallForward, PowerForward, Center

* Ezek az osztályok mind publikusan származnak le a Player osztályból

egyetlen külöbség, a string position, tehát a pozíciójuk

## Team

csapatok adatait tartalmazó osztály

private:

* string name; - csapatnév
* vector<Player> team; - a Playereket tartalmazó vector
* vector<string> positions – a pozíciókat tartalmazó vector
* static const int MaxPlayerCount = 5; - hány játékos lehet összesen a csapatban
* int playerCount; - hány játékos van a csapatban
* int teamRating; - csapat összesített rating-je

public:

* szimpla getterek, konstruktorok
* add függvény, amellyel játékost lehet a csapathoz adni(ez figyel a pozíciókra, és a méretre is)
* toString() csapat adat kiírató függvény

# UML\_osztálydiagrammA képen képernyőkép, csatlakozó látható Automatikusan generált leírás

(minden leszármazás public, csak nehezen láthazó)

# Összegzés

## Mit sikerült és mit nem sikerült megvalósítani a specifikációból?

Mindent sikerült megvalósítani a specifikációból (az API használatán kívül), új dolgok is vannak benne, ami az előző specifikációban nem volt (pl. nem 7 meccset terveztem, csak 1-et)

## Mit tanultál a megvalósítás során?

Megtanultam mennyivel felhasználó barátabb a c++ a c-hez képest, (pl. a stringek char tömbök helyett, vagy a vectorok saját statikus tömbök helyett) mérföldekkel használhatóbbá teszik a programnyelvet véleményem szerint.

## Továbbfejlesztési lehetőségek

Eredetileg egy publikus adatbázisból, vagy egy API segítségével akartam kigyűjteni a játékos adatokat, azonban ilyet c++-hoz nem találtam, még próbáltam webscrapelni, azonban a lineups.com megkéri a felhasználóit, hogy ne scrapeljék az oldalt. Ha valamilyen új API-t találnék, akkor mindig up-to-date lenne a program. (esetleg még legendás, de már nyugdíjba vonult játékosokat, vagy cseréket lehetne még bevezetni)

# Képernyőképek a futó alkalmazásról

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás