Система за управление на БД със самолети за авиобаза

Автор: Мария Дачева

Проектът представлява база данни за самолети и поддържа проста реализация на REPL(read-evaluate-print-loop). Информацията за самолетите се съхранява в постоянната памет(текстов файл) като всеки запис има вида Id, Plane, Type, Flights.

1. Архитектура

1.1 Plane

Класът Plane описва отделен запис от базата. Основните му характеристики са:

- Private:
  - unsigned long long id - id на дадения самолет
  - string plane - име на самолета
  - string type - тип на самолета
  - unsigned long long flights - брой извършени полети на самолета
- Public:
  - Plane() - конструктор по подразбиране
  - Plane(unsigned long long, const string&, const string&, unsigned long long) - конструктор, приемащ атрибутите Id - уникално число, Plane - име на самолета, Type - тип на самолета и Flights - брой извършени полети. В себе си конструкторът извиква функциите setID(unsigned long long), setPlane(const string&), setType(const string&), setFlights(unsigned long long)
  - unsigned long long getID() const - връща стойността на полето id
  - void setID(unsigned long long) - задава стойност на полето id като проверява дали подаденият параметър е валиден
  - void setPlane(const string&) - задава стойност на полето plane като проверява дали подаденият параметър е валиден
  - void setType(const string&) - задава стойност на полето type като проверява дали подаденият параметър е валиден
  - void setFlights(unsigned long long) - задава стойност на полето flights като проверява дали подаденият параметър е валиден
  - bool operator>(const Plane&) - предефиниран е оператор > за класа, така че да зависи от полето id

- friend ostream& operator<<(ostream&, const Plane&) - приятелска функция, предефинираща оператор << за класа Plane

- friend istream& operator>>(istream&, Plane&) - приятелска функция, предефинираща оператор >> за класа Plane

1.2 Node

Структурата Node е основният елемент на класа BST(Binary Search Tree). Използва се рекурсивна структура със стойност value от тип Plane и указатели left и right, отново от тип Node*. Стойността се инициализира чрез конструктор, приемащ параметър от тип const Plane& за полето value, а left и right указателите се задават с nullptr.

1.3 BST(Binary Search Tree)

Класът представлява custom реализация на двоично дърво за търсене

- Private:
  - Node* root - указател към корена на дървото

  - void deleteNode(Node*) - премахва подадения връх като рекурсивно рекурсивно изтрива наследниците му

  - void insertNode(Node *&, const Plane&) - рекурсивно добавя връх към дървото със стойност от тип Plane. Ако дървото няма върхове, то се добавя като корен. В противен случай добавя като наследник на листо в зависимост от големината на id на подадения самолет. По-големите стойности се добавят в дясно, а по-малките - в ляво.

  - Plane& searchNode*, unsigned long long) - рекурсивно проверява дали дървото съдържа самолет с подадения като параметър id. Връща стойността Plane на намерения връх, ако той съществува, в противен случай връща самолет със стойности по подразбиране.

  - void inOrder(Node *) - обхожда дървото тип ляв наследник, корен, десен наследник
- Public:
  - BST() - конструктор по подразбиране, извикващ в себе си функцията reset()

- BST(const BST&) = delete - забранява копиращ конструктор за дадения клас
- BST& operator=(const BST&) = delete - забранява предефинирането на оператора = за дадения клас
- ~BST - деструктор, извикващ в себе си функцията clear()
- void reset() - задава стойност nullptr на корена на дървото - root
- void clear() - изтрива дървото като извиква в себе си метода deleteNode(Node *) с параметър коренът на дървото
- void insert(const Plane&) - добавя нов връх към дървото като използва метода insertNode(Node *&, const Plane&) с параметри коренът на дървото и подадения самолет
- Plane& search(unsigned long long) - търси в дървото връх със стойност Plane с id подадения параметър като използва метода searchNode(Node *) с параметри коренът на дървото и подаденото id
- void print() - принтира стойностите на дървото като използва метода inOrder(Node*) с параметър коренът на дървото

1.4 PlanesDataBase

Класът PlanesDataBase описва основните операции поддържани от базата.

- Private:
  - vector<Plane> planesData - списък от самолети в базата данни
  - BST planesDataOptimized - дърво от самолети, което се използва след изпълнение на Optimize командата
  - string fileName - име на файла, който съдържа данните
  - bool isOptimized - показва дали е използвана Optimize командата
  - unsigned int countOfLines() - пресмята колко записа от тип Plane има във файла

- ➢ void readFromFile() - прочита данните за самолетите от текстовия файл

- ➢ void optimizedSearch(unsigned long long id) - изпълнява оптимизарана Search команда като използва дървото planesDataOptimized вместо вектора planesData

- ➢ void removeOptimized() - премахва оптимизирането на базата като занулява planesDataOptimized и isOptimized

- ➢ int searchByID(unsigned long long id) const - връща индекса във вектора planesData на самолета с подаденото id, ако такъв съществува, в противен случай връща -1

❖ Public:

- ➢ PlanesDataBase(const string& fileName) - конструктор, който задава стойности на fileName и isOptimized. Също така извлича данни от файла, използвайки метода readFromFile()

- ➢ void search(unsigned long long id) - търси самолет с подаденото id. Ако е изпълнена Optimizied командата, то ползва метода optimiziedSearch. Ако самолетът не бъде намерен, то уведомява потребителя

- ➢ void changeAttribute(unsigned long long id, const string& attribute, const string& newInfo) - обновява стойността на подадения атрибут на самолет с подаденото id. Ако такъв самолет не е намерен - уведомява потребителя

- ➢ void optimize() - оптимизира базата данни като добавя стойностите от вектора planesData в дървото planesDataOptimized. Задава стойност true на isOptimized

- ➢ void show(unsigned int offset, unsigned int limit) - принтира limit на брой самолети като започва от offset разстояние от първия елемент в списъка. При невалидни параметри като offset да е извън рамките на вектора planesData или offset + limit да е повече от големината на вектора, то програмата уведомява потребителя

- ➢ void addPlane(unsigned long long id, const string& plane, const string& type, unsigned long long flights) - създава нов самолет, проверявайки дали подаденото id е уникално. Ако то не е уведомява потребителя. Премахва оптимизирането

- ➢ void deletePlane(unsigned long long id) - изтрива самолет по подадено id като намира позицията на самолета чрез метода searchByID(unisgned long long). Ако такъв не бъде намерен - уведомява потребителя

- ➢ void writeToFile() - запазва стойностите от planesData във файла

2. Четене на входните данни

- ❖ create Id Plane Type Flights създава запис за самолет с въведените от потребителя данни за самолет (Id Plane Type Flights). Използва помощната функция create(PlanesDataBase& db, unsigned long long, const string&, const string&, unsigned long long), която в себе си извиква addPlane върху db

- ❖ delete Id - изтрива запис за самолет с въведения от потребителя Id. Използва се помощната функция deleteById(PlaneDataBase& db, unsigned long long), която в себе си извиква функцията deletePlane на класа PlaneDataBase върху db

- ❖ update Id attribute attributeValue e - обновява записаната информация за въведения от потребителя атрибут attribute(Id или Plane или Type или Flights) за запис за самолет с въведения от потребителя Id. Използва се помощната функция update(PlanesDataBase& db, unsigned long long, const string&, const string&), която в себе си извиква функцията changeAttribute на класа PlaneDataBase върху db

- ❖ show offset limit - отпечатва на стандартния изход limit на брой записи за самолети започвайки от записа на разстояние offset от първия запис. Използва се помощната функция show(PlanesDataBase& db, unsigned int, unsinged int), която в себе си извиква функцията show на класа PlaneDataBase върху db

- ❖ optimize - изпълнението на тази команда започва построяване на индекс за атрибута Id. Сложността по време за построяване на индекс трябва да бъде O(n.log(n)). Използва се помощната функция optimize(PlanesDataBase& db), която в себе си извиква функцията optimize на класа PlaneDataBase върху db

- ❖ search Id - отпечатва на стандартния изход пълния запис (Id Plane Type Flights) за самолетът с въведения от потребителя Id. Използва се

помощната функция search(PlanesDataBase& db, unsigned long long), която в себе си извиква функцията search на класа PlaneDataBase върху db

- exit - спира изпълнението на програмата като извиква и функция writeToFile на класа PlaneDataBase и записва информацията във файла

3. Схема на проекта