Склад

Линк към хранилището в *Github*: https://github.com/KrisCvertanov/Storage

1.Увод

Разглежданият проект реализира информационна система, обслужваща склад. Обслужването се изразява в промени по продуктите на разположение в склада(добавяне/изваждане/разчистване). Всяка промяна се следи и се отчита. Програмата обработва и съхранява данните по наличността в склада във файл.

За реализирането на програмата трябва да бъдат постигнати следните цели и задачи:

- Подходящо представяне на един продукт за програмата, така че да бъдат обхванати неговите по-важни характеристики: име, производител, срок на годност и т.н.
- Изграждането на склад. Неговата главна цел ще е съхранението на множество от продукти, като всеки продукт трябва да има уникално местоположение. Това зависи от самия избор на структура за склада.
- Промените по множеството от продукти да се осъществяват с набор от операции, които програмата ще предлага.
- Правените изменения да могат да се записват във файлове. Трябва да и има опция тези промени да бъдат отхвърлени.

Документацията за проекта е разделена на 3 слоя:

1. Увод
2. Проектиране
3. Реализация и уточнения

2. Проектиране

Избраната структура на склада е следната:

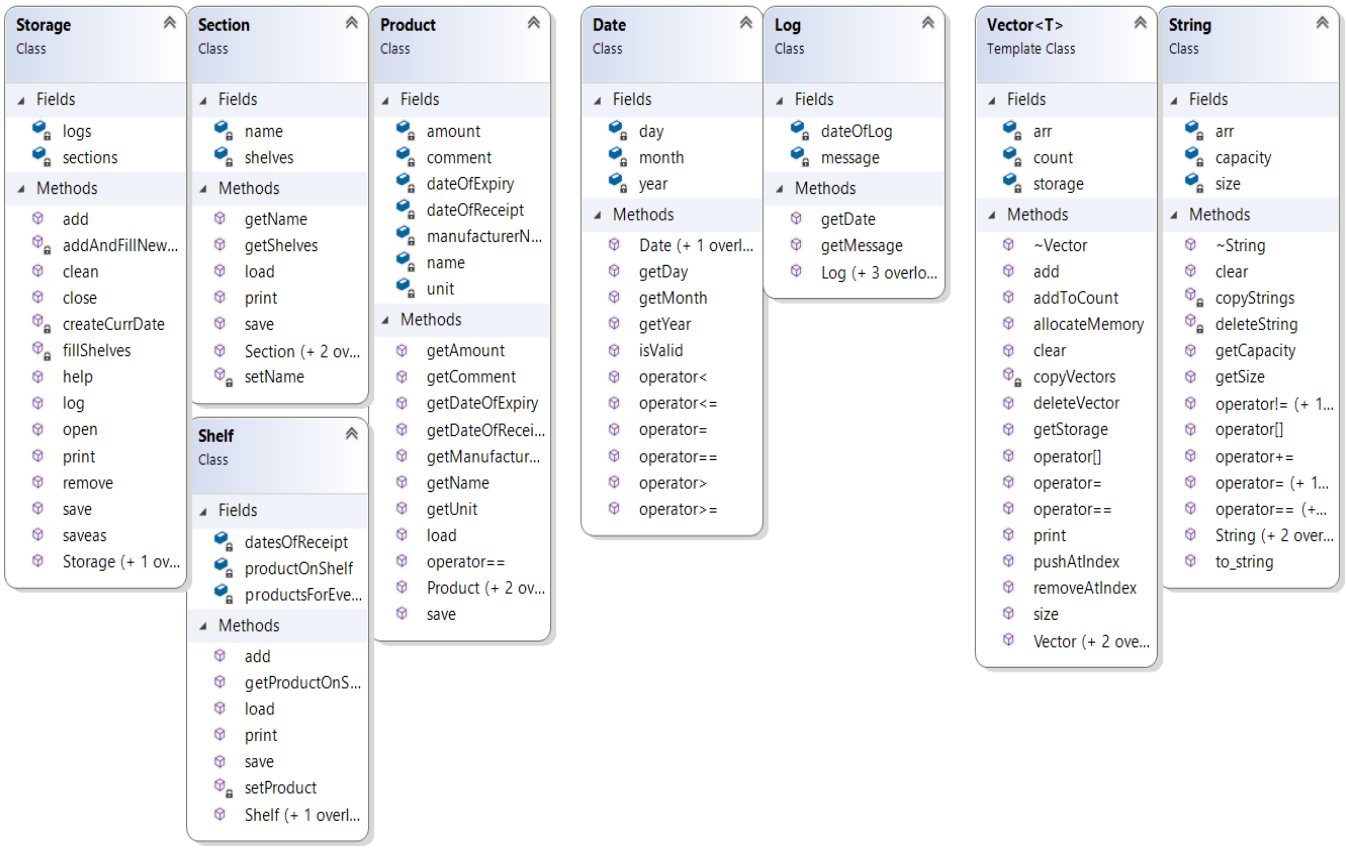
- Складът има множество от секции;
- Секцията има множество(ограничено) от рафтове;
- Рафтът има множество(ограничено) от продукти.

Реализирани са 8 класа:

- Product
- Shelf

- Section
- Storage
  - Date
  - Log
  - String
  - Vector<T>

Класовете, отбелязани с ●, са главни за проекта. Класовете, отбелязани с ○, са помощни, с цел улесняване работата с главните. Ето и характеристиките и йерархията на класовете:



Диаграма 2.1

1. **Клас String**: осъществява преоразмеряващ се динамичен масив от символи. Има следните три полета:
   - char* arr - динамичен масив от символи;
   - int size - размер на масива;
   - int capacity - заделена памет за масива.

Методи:

- String() - конструктор по подразбиране
- String(const char*, cons tint, const int) - конструктор с параметри;
- String(const String&) - копи-конструктор;
- String& operator=(const String&) - оператор равно спрямо друг обект от тип String;
- String& operator=(const char*) - оператор равно спрямо низ;
- ~String() - деструктор;
- void copyStrings(const char*, const int, const int) - функция, достъпна само за класа, копираща информация по зададени параметри. Следва добрите принципи на ООП;
- void deleteString() - изтрива заделената памет за масива. Следва добрите принципи на ООП;
- friend std::istream& operator>>(std::istream&, String&) - оператор за вход. Чрез него се въвежда обект от тип String.
- friend std::ostream& operator<<(std::ostream&, String&) - оператор за изход. Чрез него извежда обект от тип String.
- friend std::ifstream& operator>>(std::ifstream&, String&) - оператор за четене от файлов поток.
- friend std::ofstream& operator<<(std::ofstream&, String&) - оператор за писане във файлов поток.
- char& operator[](const int) - оператор за достъп до даден елемент на обекта
- bool operator==(const String&) const - оператор за сравняване на два обекта от типа, казва дали са еднакви или не.
- bool operator==(const char*) const - оператор за сравняване на обект от тип String и низ, казва дали са еднакви или не.
- bool operator!=(const String&) const - оператор за различаване на два обекта от типа, казва дали са различни или не.
- bool operator!=(const char*) const - оператор за различаване на String и низ, казва дали са различни или не.
- String& operator+=(const String&) - конкатенира два обекта от тип String
- String& to_string(const int) - метод, приемащ цяло неотрицателно число, превръщащо го в обект от тип String
- int getCapacity() const - метод за достъп до заделената памет
- int getSize() const - метод за достъп до големината
- void clear() - изчиства заделената памет за масива

2. **Клас Vector**<T>: осъществява преоразмеряващ се динамичен масив от шаблонен тип. Има следните три полета, аналогични на клас String:
    - T* arr;
    - int storage;
    - int count;

    Методи:

    - Vector() - конструктор по подразбиране;
    - Vector(int) - коснтруктор с параметър int;
    - Vector(const Vector&) - копи – конструктор;
    - Vector& operator=(const Vector&) - оператор=;
    - ~Vector() - деструктор
    - void add(const T&) - добавя елемент към края на вектора.;
    - int getStorage() const - метод за достъп на заделената памет;
    - int size() const - метод за достъп на големината ;
    - void print() const - извежда елементите на вектора;
    - void pushAtIndex(const T&, int) - на зададен индекс се присвоява новата стойност;
    - void removeAtIndex(int) - премахва на зададен индекс елемента. Големината на вектора намалява с 1;
    - void clear() - изчиства заделената памет за масива;
    - void deleteVector() - изтрива заделената памет за масива;
    - void copyVectors(const Vector&) - копира информация от друг вектор;
    - void addToCount() - увеличава големината с 1(достъпен само за класа!);
    - void allocateMemory(int) - заделя подаденото количество памет за вектора
    - bool operator==(const Vector<T>&) const - сравнява дали два вектора са еднакви
    - T& operator[] (int) const - оператор за досъп до даден елемент

Класовете String и Vector<T> са съществени за проекта, понеже по – голямата част от полетата на останалите класове са от такива типове и спестяват работата с динамична памет.

3. **Клас Date**: представя валидна дата за програмата. Има следните полета:
    - int year - година
    - char month[3] - месец
    - char day[3] - ден

    Методи:
    - Date() - конструктор по подразбиране
    - Date(cons tint, const char*, const char*) - конструктор с параметри

- Date& operator=(const Date&) - оператор=
- const int getYear() const - метод за достъп до годината
- const char* getMonth() const - метод за достъп до месеца
- const char* getDay() const - метод за достъп до деня
- bool operator==(const Date&) const - проверява дали две дати съвпадат
- bool operator<(const Date&) const - проверява дали дата е по – стара от друга дата
- bool operator>(const Date&) const - проверява дали дата е по – нова от друга дата
- bool operator<=(const Date&) const - проверява дали дата е по – стара или същата спрямо друга дата
- bool operator>=(const Date&) const - проверява дали дата е по - нова или същата спрямо друга дата
- bool isValid() const - проверява дали една дата е валидна
- friend std::istream& operator>>(std::istream&, Date&) - оператор за вход
- friend std::ostream& operator<<(std::ostream, const Date&) - оператор за изход

4. **Клас Log**: осъществява справка, която отразява промяна по склада. Има следните полета:
   - Date dateOfLog - датата на справката
   - String message - съобщението от справката

   Пример:

   2020-05-23 operation: add | Section: 1 | Shelf: 2 | product: patladjan | amount: 10

   Методи:

   - Log() - конструктор по подразбиране
   - Log(const Date&, const String&) -
   - Log(const Date&, const char*) - конструктори с параметри
   - Log(const char*, const char*) -
   - const Date& getDate() const - метод за достъп до датата на справката
   - const String& getMessage() cosnt - метод за достъп до съобщението от справката
   - friend std::istream& operator>>(std::istream&, Log&) - оператор за вход
   - friend std::ostream& operator<<(std::osteam&, const Log&) - оператор за изход

5. **Клас Product**: представя продукт за програмата. Има следните полета:
   - String name - име на продукта
   - Date dateOfExpiry - дата, до която е годен продукта
   - Date dateOfReceipt - дата на внос в склада

- String manufacturerName - производите
- String unit - мерна единица(кг/л)
- int amount - количество
- Srting comment - кратък коментар

Методи:

- Product() - конструктор по подразбиране
- Product(const String&, const Date&, const Date&, const String&, const String&, const int, const String&) - конструктор с параметри
- Product(const char*, const char*, const char*, const char*, const char*, const int, const char*) - конструктор с параметри(низове)
- const String& getName() const - метод за достъп до името
- const Date& getDateOfExpiry() const - метод за достъп до срока на годност
- const Date& getDateOfReceipt() const - метод за достъп до датата на внос
- const String& getManufacturerName() const - метод за достъп до производителя
- const String& getUnit() const - метод за достъп до мерната единица
- const int getAmount() const - метод за достъп до количеството
- const String& getComment() const - метод за достъп до коментара
- bool operator==(const Product&) const - сравнява два продукта дали съвпадат
- friend std::ifstream& operator>>(std::ifstream&, Product&) - оператор за четене от файлов поток
- friend std::ofstream& operator<<(std::ofstream&, const Product&) - оператор за писане във файлов поток
- friend std::istream& operator>>(std::istream&, Product&) - оператор за вход
- friend std::ostream& operator<<(std::ostream&, const Product&) - оператор за изход
- void save(std::ofstream&) - метод за запазване на характеристиките на продукт във файлов поток
- void load(std::ifstream&) - метод за извличане на характеристиките на продукт от файлов поток

6. **Клас Shelf**: представя рафт за програмата. Има следните полета:
   - Product productOnShelf - представя продукта, който се намира на дадения рафт
   - Vector<Date> datesOfReceipt - масив от датите на внос за продукта на рафта

- Vector<int> productsForEveryDate - масив, който пази за всяка дата на внос какво количество е било заредено на рафта

Методи:

- Shelf() - конструктор по подразбиране
- Shelf(const Product&, const Vector<Date>&, const Vector<int>&) - конструктор с параметри
- void add(int, const Date&) -
- void print() const - извежда информацията за рафта
- const Product& getProductOnShelf() const - метод за достъп до продукта на рафта
- void save(std::ofstream&) - метод за записване на характеристиките на рафта във файл
- void load(std::ifstream&) - метод за извличане на характеристиките на рафт от файл
- void setProduct(cosnt Product&) - мутатор за продукта на рафта(недостъпно за външния свят)

7. **Клас Section**: представя секция за програмата. Има следните полета:
   - String name - име на секцията
   - Vector<Shelf> shelves - множеството от рафтове за секцията

   Методи:

   - Section() - конструктор по подразбиране
   - Section(const String&, const Vector<Shelf>&) - конструктор с параметри
   - Section(const char*, const Vector<Shelf>&) - конструктор с параметри
   - void print() const - извежда информация за секцията
   - const String& getName() const - метод за достъп до името на секцията
   - const Vectror<Shelf> getShelves() const - метод за достъп до рафтовете на секцията
   - void save(std::ofstream&) - метод за записване на характеристиките на секция във файл
   - void load(std::ifstream&) - метод за извличане на характеристиките на секция от файл
8. **Клас Storage**: осъществява представянето за склада. Има следните полета:
   - Vector<Section> sections - множеството от секции в склада
   - Vector<Log> logs - отчита справки по промените в склада

   Методи:

   - Storage() - конструктор по подразбиранею

- Storage(const Vector<Section>&, const Vector<Log>&) - конструктор с параметри
- void print() const - извежда информация по наличността в склада
- void add(Vector<Section>&) - добавя продукт в диалогов режим
- void remove(Vector<Section>&) - изважда продукт в диалогов режим
- void log(const char*, const char*) const - за даден период от време извежда справки за промените по склада
- void clean(Vector<Section>&) - разчиства склада от всички продукти с изтекъл срок
- void open(const char*, Vector<Section>&) - отваря и зарежда информацията от файл с посочено име
- void close(Vector<Section>&) const - затваря текущия файл и изчиства заредената информация
- void save(const char*, Vector<Section>&) - запазва промените по склада в текущо отворения файл
- void saveas(const char*, Vector<Section>&) - запазва промените по склада в нов файл
- void help() const - извежда списък с поддържаните от програмата команди
- void fillShelves(int, const Product&, Vector<Shelf>&) - запълва определено количество рафтове с даден продукт(достъпно само за класа)
- void addAndFillNewSections(int, const Product&, Vector<Section>&, const Date&) - добавя и запълва секции с даден продукт, като се подава и датата на извършване на действието(достъпно само за класа)
- const Date& createCurrDate() const - връща датата, която е текуща за компютъра(достъпно само за класа)

3. Реализация и уточнения

Класовете **String** и **Vector<T>** са единствените, в които има динамично заделяне на памет. Това значително подпомага и улеснява реализацията на останалите класове. При всяко използване на оператора new се прави проверка дали динамичната памет е заделена успешно или не. Пример:

```
arr = new(std::nothrow) char[1];

if(arr == nullptr){
   throw (“No memory for string!”);

}
```

Въвеждането на обект от тип **String** става символ по символ.

Клас **Date:**

- Месеца и деня от датата са представени като низ
- Датата е в посочения в условието формат
- При въвеждането на дата се проверява дали тя е валидна и в правилния формат

Клас **Log:**

- Класът се използва за улеснено създаване на справка и нейното съхранение
- Клас **Storage** е приятел за класа

Клас **Product:**

- В условието на задача се казва, че един продукт има за характеристика местоположение в склада. Тази черта на продукта не е реализирана като поле за класа. Тя еднозначно се определя от мястото(секция -> рафт), на което се намира.
- Не може продукт да бъде внесен след изтичането на срока му на годност. Складът не приема развалени продукти.
- Приема се, че името на производителя може да съдържа всякакви символи
- Името може да съдържа **само малки, главни латински букви и арабски цифри**
- Два продукта се сравняват за еднаквост, ако имат същите **имена, срок на годност, производител и мерна единица**. Причините са свързани с изискваната функционалност.
- Мерната единица може да е само kg или l
- Класовете **Shelf** и **Storage** са приятели за класа

Клас **Shelf**:

- Един рафт не е безкраен. Приема се, че горната граница за броя продукти, които побира, е **50**.
- Всички продукти на рафта имат единствен представител: член – данната productOnShelf. Безсмислено е да се заделя памет за 50 продукта за един рафт, при положение че те ще имат минимални разлики. Ползват се всички характеристики на член – данната, с изключение на датата на внос. Не пречи на един рафт да има продукти, внесени на различни дати.

- Векторът datesOfReceipt пази именно датите на внос за продуктите на рафта. Иначе бихме загубили това като характеристика.
- Векторът productsForEveryDate следи за всяка от датите на внос какво количество е добавено към рафта. Това е нужно, понеже премахването на продукти трябва и да се отразява на вектора с дати, иначе се губи като характеристика. При операция **remove** се изваждат от тези продукти (**гледано за рафта**), внесени най – отдавна.
- Продукти с различни **срокове на годност, производители или мерни единици** се намират на различни рафтове
- Ако се премахнат всички продукти от рафт, той също се премахва
- Класът **Storage** е приятел за класа

Клас **Section**:

- Секциите не са безкрайни. Приема се за горна граница на броя рафтове за секция да е **50**.
- Една секция не стартира директно с 50 рафта. Всеки рафт се добавя при нужда от място.
- Всяка секция има име: името на продукта, който се съхранява в нея. Продукти с различни имена стоят в различни секции.
- Естествено, възможно е да има повече от една секция с едно и също име, понеже побират фиксирано количество продукти.
- Ако се премахнат всички рафтове от секция, тя също се премахва
- Клас **Storage** е приятел за класа

Клас **Storage:**

- Също като рафтовете и секциите, складът не е безкраен, но може да бъде разширяван. Ако в даден момент мястото в склада свърши, е възможно наемането или създаването на някаква постройка в близост до склада, служеща за разширение. В този смисъл, няма горна граница за секциите в склада.
- Понеже трябва да има опция направените изменения по склада да се отхвърлят, в главната функция се създават временен вектор от секции. Всички команди, правещи промени по наличността, се изпълняват върху него. При команда open, прочетените данни се присвояват на оригиналните секции.
- Направени и записани промени по склада са достъпни чак след повторно ползване на командата open!
- Промените по наличността се следят от стартирането на програмата до приключването ѝ(т.е. не се съхраняват във файл напр.)

- При командата **add**, за дата на добавяне се приема вносната дата на продукта.
- При командите **remove** и clean, се избира между две опции за дата на премахване/изчистване на продуктите : 1. Въведена от ползващия програмата дата или 2. Текущата за компютъра дата

Общи подробности:

- Между всеки две думи в команда има **точно един** интервал.
- Всяка команда се въвежда по посочения в условието начин. Не може да започва или завършва с интервал.
- Приема се че в името на файловете няма интервали
- При **технически** или **логически** неправилно използване на командите се извежда съответното съобщение за грешка.
- Тестове има в хранилището в *Github*