|  |  |  |
| --- | --- | --- |
| *Софийски университет* |  | *Факултет по математика* |
| „*Св. Климент Охридски“* | *и информатика* |
|  |  |  |

Бакалавърска програма   
„Информационни системи”

Дисциплина „Обектно-ориентирано програмиране”

Проект

“Склад”

**Изготвил: Преподавател:**

Елис Данга Калин Николов

ф.н. 71998

**Съдържание**

Увод 2

Преглед на предметната област 2

Проектиране 3

Реализация 5

Заключение 8

## Увод

Работата в склад е изключително трудоемка. Тя изисква много усилия и организираност за своето изпълнение. За улеснение на този вид дейност, се използват програми, реализиращи информационни системи за обслужване на складове. Именно написването на такава програма е в основата на този проект.

Склада е помещение, предназначено за съхраняване на [стоки](https://bg.wikipedia.org/wiki/%D0%A1%D1%82%D0%BE%D0%BA%D0%B8) и ценности. Следователно програмата ще работи главно с данните за налични продукти, като ги съхранява и обработва във файл. За тази цел програмата се разделя на два основни етапа:

1. Съхраняване на информация за всеки продукт

(име, срок на годност, дата на постъпване в склада, име на производител, мерна единица, налично количество, местоположение в склада и коментар)

1. Извършване на следните операции:
   1. Основни (Команден ред)

(отваряне, затваряне, запазване на промените, записване като, помощ и изход)

* 1. Специфични

(извеждане на информация за наличните продукти в склада, добавяне на нова стока и поверяване за коректност, премахване на вече съществуваща, извеждане на справки за всички промени в даден период и разчистване на склада от продукти с изтекъл срок на годност)

В следващите няколко страници ще се опише подробно реализацията на проекта. Ще се прегледат основните концепции, дефиниции и методи. Ще се представят както архитектурата на програмата и диаграмите на поведение, така и извадка от кода и тестови сценарии. В обобщение ще се разгледа изпълнението на началните цели, бъдещото развитие и усъвършенстване.

## Преглед на предметната област

В проекта са дефинирани 2 основни класа – Product и FileSystem, които са в основата на задачата и изпълняват първия етап, а Project12 изпълнява втория. Основната концепция е работа с файлове. Съдържанието на файла се съхранява в стринга buffer, чрез който генерираме обекти в масива products. Благодарение на класа Product, работата и достъпа до продуктите е улеснен.

Сложността на дадена задача се определя от вида на стъпките до реализирането й. В нашия случай те са три:

* Извличане на информация
* Обработка на дадената информация в паметта
* Записване на промените

В процеса на разработка на системата, различните команди ги разглеждаме като отделни подзадачи или по-точно проблеми. Като за решаването им използваме операциите от втория етап. Нужна ни е структура, която да пази информацията за складираните продукти и за тази цел се използва масива от обекти на класа Product. Пазенето се реализира чрез динамична памет, която след това се изтрива.

Тук използваме функционалните изисквания. Всяка команда изпълнява някаква операция. Например чрез add добавяме нов продукт в масива, чрез print се извежда информация за наличните продукти в склада, чрез clean се разчиства склада от всички стоки с изтекъл срок на годност и т.н. Освен основното действие, при някой операции, има и допълнителни условия. Например при add, едно от допълнителните условия е ако еднакви продукти имат един и същи срок на годност, да се съхраняват на едно и също място.

[1] for (int i = 0; i < number\_of\_product\_in\_buffer; i++) {

if ((products[i].get\_name() == name) && (products[i].get\_expiring\_date() == expiring\_date)) {

location = products[i].get\_location();

}

}

## Проектиране

Както казахме по-нагоре, програмата е съставена от два основни класа.

Класа Product има за член-данни информацията за даден продукт

* име
* срок на годност
* дата на постъпване в склада
* име на производител
* мерна единица(килограми, литри)
* налично количество
* местоположение в склада
* коментар

а като член-функции

* конструктор по подразбиране
* конструктор с параметри(информацията за продукта)
* getters
* setters
* add() - функция за добавяне
* remove() - функция за премахване
* create\_from\_file() - функция която създава продукт от буфера
* toString() - функция която превръща продукт в стринг
* fromString() - ункция което превръща стринг в продукт
* load\_products() - функция която зарежда продукти

Класа FileSystem има една член-данна

* Stream

и две член-функции

* read() – чете от файл
* write() – записва във файл

В главната функция са записани най-важните операции в програмата. В началото са създадени масив products и стринг buffer, съхраняващи съдържанието на файла. Нашата програма е създадена по команден ред, тоест в зависимост от командата която дадем се извършват различни операции.

За да намерим каква команда е въвел потребителя и какви са нейните параметри, използваме следния подход :

[2] int delimiter\_pos = input.find(' ');

command = input.substr(0, delimiter\_pos);

input.erase(0, delimiter\_pos + 1);

if (delimiter\_pos != -1) {

while (!input.empty()) {

delimiter\_pos = (input.find(" ") != -1) ? input.find(" ") : input.size();

args[i] = input.substr(0, delimiter\_pos);

input.erase(0, delimiter\_pos + 1);

i++;

}

}

В променливата command запазваме името на командата, а аргументите в args (ако има такива). Project12.cpp се грижи за разпознаването на командите и тяхното изпълнение. Например операцията remove:

[3] if (command.compare("remove") == 0) {

if (main\_filename.size()) {

if (args[0].size()) {

Product::remove(buffer, args[0]);

Product::load\_products(buffer, products);

std::cout << "Successfuly remove!\n";

}

else {

std::cout << "Please use format \"remove product\_name\"" << std::endl;

}

}

else {

std::cout << "Open file first!" <<

std::endl;

}

}

Тук имаме разпознаване на команда, проверка дали има отворен файл и проверка на синтаксиса на операцията, както и подходящи съобщения за грешка. За извършването на операцията са използвани функции от класа Product. В главната програма можем да видим връзката и значимостта между класовете.

## Реализация, тестване

В класа FileSystem имаме две член-функции :

- write() – записва input във файла, като първо изтрива всичко и след това го презаписва обновено.

[4] this->stream.open(file\_name, std::ios::in | std::ios::out | std::ios::trunc);

- read() – чете output ред по ред и връща съобщение дали е прочетено

В класа Product по-значими функции са:

- load\_products() – зарежда продуктите от буфера в масива, като намира първия ред и с помощта на функцията fromString() записва продукт

[5] while (!buffer.empty()) {

pos = (buffer.find("\n") != -1) ? buffer.find("\n") : buffer.size();

std::string line = buffer.substr(0, pos);

pr\_array[n] = fromString(line);

buffer.erase(0, pos + 1);

n++;

}

- fromString() – записваме продукт в масива от даден стринг, като използва цикъл за да намери първата дума и я записва в масива params[8]. Тъй като имаме определен синтаксис на информацията за продукта, в зависимост индекса на масива отговаря на определен критерии

* [6] while (!input.empty()) {

delimiter\_pos = (input.find(" ") != -1) ? input.find(" ") : input.size();

* params[i] = input.substr(0, delimiter\_pos);
* input.erase(0, delimiter\_pos + 1);
* i++;
* }

Основната информация за наличните продукти се пази в стинг, от който се генерират продукти в масива [6]. Промяната на информацията се прилага първо върху масива и след това се записва в буфера, като се изтрива изцяло и се презаписва наново обновен [5]. Чрез цикъл обхождаме въведената информация и разграничаваме командата от аргументите [2]. В зависимост от командата се изпълняват различни операции [3]. За съхранението на продуктите се използва динамична памет която после се освобождава.

Тестване на програмата:

*>open Test.txt*

*Successfully opened file: Test.txt*

*>print*

*product1 2021-03-03 2020-04-03 svoge count 40.00 1/1/1 ""vkusen""*

*product2 2021-06-09 2020-07-04 milka count 42.00 1/2/1 ""vkusen""*

*product3 2018-07-08 2019-05-03 mlechen count 21.00 2/1/2 ""vkusen""*

*product4 2019-08-03 2018-01-03 svoge count 56.00 2/2/1 ""vkusen""*

*product4 2022-08-03 2020-01-03 svoge count 56.00 3/2/1 ""vkusen""*

*>add*

*Enter name*

*product5*

*Enter expiring date (yyyy-mm-dd)*

*2020-05-03*

*Enter entry\_date (yyyy-mm-dd)*

*2020-06-10*

*Enter producer*

*milde*

*Enter unit (kg, l, count)*

*count*

*Enter quantity*

*12.00*

*Enter location(x/x/x)*

*2/1/3*

*Enter comment*

*"meka"*

*Successfully added!*

*>save*

*File saved!*

*>close*

*Successfully closed file!*

*>open Test.txt*

*Successfully opened file: Test.txt*

*>print*

*product1 2021-03-03 2020-04-03 svoge count 40.00 1/1/1 ""vkusen""*

*product2 2021-06-09 2020-07-04 milka count 42.00 1/2/1 ""vkusen""*

*product3 2018-07-08 2019-05-03 mlechen count 21.00 2/1/2 ""vkusen""*

*product4 2019-08-03 2018-01-03 svoge count 56.00 2/2/1 ""vkusen""*

*product4 2022-08-03 2020-01-03 svoge count 56.00 3/2/1 ""vkusen""*

*product5 2020-05-03 2020-06-10 milde count 12.00 2/1/3 """meka"""*

*>remove*

*Please use format "remove product\_name"*

*>remove product5*

*Successfully remove!*

*>print*

*product1 2021-03-03 2020-04-03 svoge count 40.00 1/1/1 ""vkusen""*

*product2 2021-06-09 2020-07-04 milka count 42.00 1/2/1 ""vkusen""*

*product3 2018-07-08 2019-05-03 mlechen count 21.00 2/1/2 ""vkusen""*

*product4 2019-08-03 2018-01-03 svoge count 56.00 2/2/1 ""vkusen""*

*product4 2022-08-03 2020-01-03 svoge count 56.00 3/2/1 ""vkusen""*

*>exit*

*Exiting the program...*

## Заключение

Програмата е реализирана и работи успешно. Началните цели са постигнати, зададените операции и проверките към тях са изпълнени. Отварянето на повече от един файл едновременно и автоматично генериране на местоположението на продуктите спрямо зададени критерии са малка част от бъдещото развитие и усъвършенстване на програмата.

*Линк към github -* [*https://github.com/elisdanga/Storage*](https://github.com/elisdanga/Storage)