Проект 12: Склад

*Изготвил:* Георги Станчев, I курс, ИС, Фак. № 71959

[Глава 1. Увод 3](#_Toc42075794)

[1.1. Описание и идея на проекта 3](#_Toc42075795)

[1.2. Цел и задачи на разработка 3](#_Toc42075796)

[1.3. Структура на документацията 4](#_Toc42075797)

[Глава 2. Преглед на предметната област 4](#_Toc42075798)

[2.1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани 4](#_Toc42075800)

[2.2. Дефиниране на проблеми и сложност на поставената задача 5](#_Toc42075801)

[2.3. Подходи, методи (евентуално модели и стандарти) за решаване на поставените проблемите 5](#_Toc42075802)

[2.4. Потребителски (функционални) изисквания (права, роли, статуси, диаграми, ...) и качествени (нефункционални) изисквания (скалируемост, поддръжка, ...) 5](#_Toc42075803)

[Глава 3. Проектиране 6](#_Toc42075804)

[3.1. Обща архитектура – ООП дизайн 6](#_Toc42075806)

[Глава 4. Реализация, тестване 7](#_Toc42075807)

[4.1. Реализация на класове (включва важни моменти от реализацията на класовете и малки фрагменти от кода) 7](#_Toc42075809)

[4.2. Управление на паметта и алгоритми. Оптимизации 8](#_Toc42075810)

[4.3. Планиране, описание и създаване на тестови сценарии (създаване на примери) 8](#_Toc42075811)

[Глава 5. Заключение 16](#_Toc42075812)

[5.1. Обобщение на изпълнението на началните цели 16](#_Toc42075814)

[5.2. Насоки за бъдещо развитие и усъвършенстване 16](#_Toc42075815)

[Глава 6. Използвана литература 16](#_Toc42075816)

# Увод

* 1. Описание и идея на проекта

Да се напише програма, реализираща информационна система, обслужваща склад. Програмата съхранява и обработва данните за наличността в склада във файл.

За всеки продукт се съхранява следната информация:

* име (описание на продукта, символен низ с произволна дължина)
* срок на годност
* дата на постъпване в склада
* име на производител
* мерна единица (килограми, литри)
* налично количество
* местоположение (секция/рафт/номер)
  + номерирайте склада си, както прецените, че ще ви е удобно, имайте предвид, че в началото той е празен и различно количество стока е нормално да заема различно по обем място
* коментар (свободен текст)

**Бонуси:**

* по въведен продукт и неговата цена, за брой или съответно килограм, да се пресметнат загубите за даден от потребителя период (изхвърленото количество продукт със развален срок на годност)
  1. Цел и задачи на разработка

Целта на приложението, е да може да оперира с множество файлове и да може да извършва общи операции (open, close, save, save as, help и exit). След като приложението отвори даден файл, то трябва да може да извършва посочените по-долу операции (*Таблица 1*):



|  |  |
| --- | --- |
| print | Извежда информация за наличните продукти в склада |
| add | Добавя нов продукт в склада в диалогов режим, като въвежда всички данни за продукта и ги проверява за коректност. При добавянето на продукта се спазват следните правила:   * ако нов продукт е с различен срок на годност от вече съществуващ едноименен продукт, той да бъде поставен на различно място * ако имате достатъчно място, еднакви продукти с един и същи срок на годност да бъдат поставени на едно и също място * при извеждане на списъка с налични продукти да се изведе общото количество на едноименните продукти, независимо от срока им на годност |
| remove | Изважда продукт от склада в диалогов режим:   * по дадено име и количество изважда съответните продукти от склада и извежда информацията за продукта и къде се е намирал * при наличие на повече от една партида, първо намалява тази със най-скоро изтичащ срок на годност, тогава във информацията за извършеното действие се отбелязва количеството и мястото на всяка от партидите, които сме намалили * в случай на опит за изваждане на повече от наличното да се дава информация на потребителя за наличността на продукта и срока на годност на партидите и възможност да реши дали все пак не иска да извади това което е останало |
| log <from> <to> | Извежда справка за всички промени в наличността в периода от дата <from> до дата <to>, включително зареждания и извеждания на стоки. |
| clean | Разчиства склада от всички стоки, на които е изтекъл или предстои скоро да изтече срока на годност, като извежда информация за разчистените стоки |

* 1. Структура на документацията

Документа е структуриран в пет основни глави, описващи целта и идеята на проекта, както и неговото реализиране.

# Преглед на предметната област

1. 1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани

Програмата е реализирана на Visual Studio 2019 , C++. Основната идея, по която е разработена, е информационна система за склад, която съдържа в себе си информация за наличността на склада заедно със историята на зареждания и извеждания на стоки, като самия склад се състои от 24 на брой секции, 2 x 12 за литри и килограми. Всяка секция се състои от 6 рафта с по 4 отделения с максимум на количеството 40 кг/л, като за един едноименен продукт максимума е количеството, което може да се сложи на два рафта, следователно 4 x 40 x 2 = 320 кг/л. Във връзка с информацията за времето, по което е извършвано всяко действие е заимстван алгоритъм за изчисляване на датата.

* 1. Дефиниране на проблеми и сложност на поставената задача

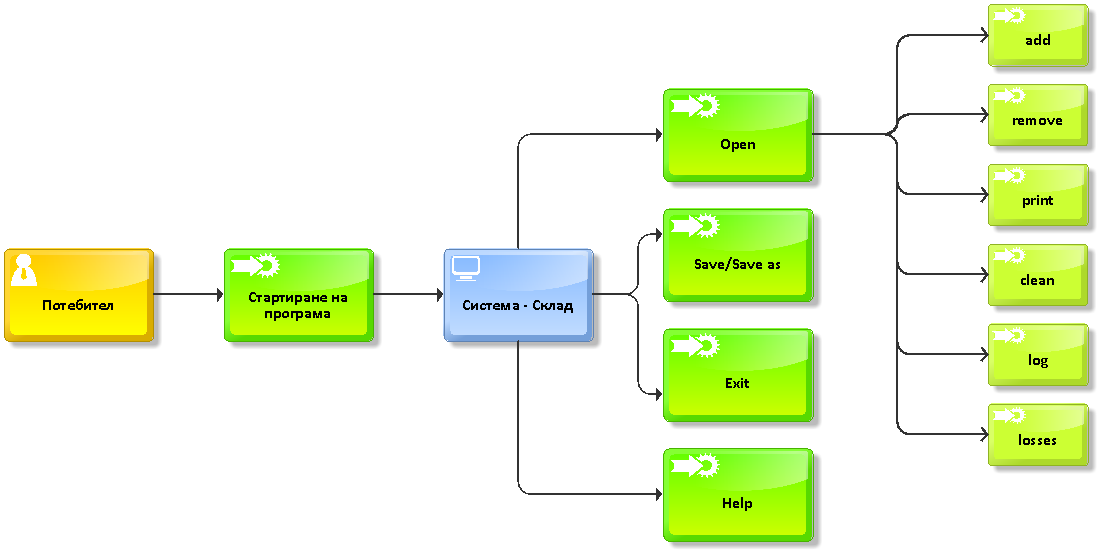
От самото начало бе съставен план, който стриктно се следваше докрай. Като цяло функционалността не представляваше толкова трудност, колкото самата концепция за информационна система на склад, тъй като самата програма трябва да определя местата на продуктите по специфични критерии, а не както принципно се подреждат от работниците в склада, които влагат умисъл в разумната подредба.

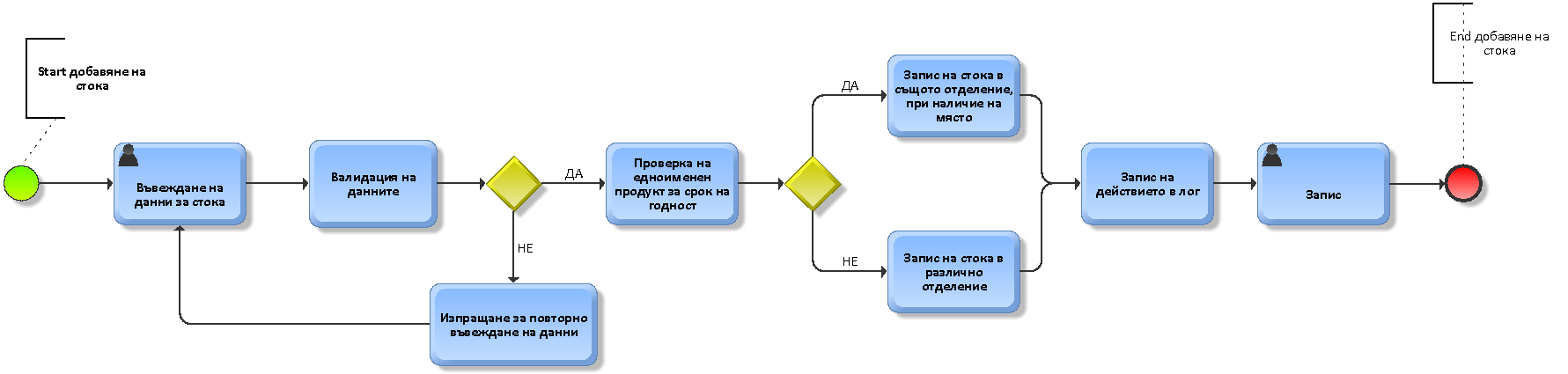
* 1. Подходи, методи (евентуално модели и стандарти) за решаване на поставените проблемите

След като бе направен анализ на задачата и поставените цели, първата ми задача беше да направя основата – имплементация на продукт в клас и създаването на модел на склад, чрез секции, рафтове и отделения в тях. След това започнах да си създавам функционалностите – добавяне и премахване на продукт, разчистване на стоките с изтекъл срок на годност и създаване на записите при всяка промяна в наличността. В крайна сметка бе направена и бонус задачата, която представляваше пресмятане на загубите по даден продукт в определен период.

* 1. Потребителски (функционални) изисквания (права, роли, статуси, диаграми, ...) и качествени (нефункционални) изисквания (скалируемост, поддръжка, ...)

Потребителските (функционални) изисквания към приложението (*Фигура* *1*) се дефинират и детайлизират с цел постигане на високо ниво на разбиране за потребностите на потребителя от функционалността на приложението и дават ясна представа за отделните компоненти, които да се разработят по подходящ метод и алгоритъм.



На следващата схема (*Фигура 2*) е показана детайлно функционалността на операция „add”, което представлява добавяне на продукт в склада:

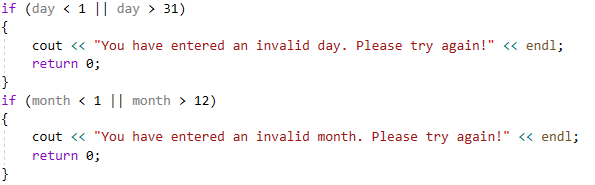


# Проектиране

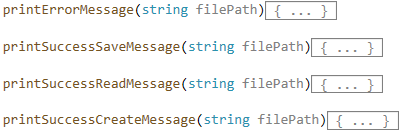
1. 1. Обща архитектура – ООП дизайн

Архитектурата на програмата е внимателно моделирана, вземайки се предвид и най-малките детайли. Създадени са нужните класове за поддържането на функционалността, а изгледа е достатъчно улеснен с цел подпомагане на потребителя как да използва програмата правилно и лесно. (*Фрагмент 1*)



  
Предвидени са всички случаи, в които потребителя може да сгреши или неволно, ако обърка нещо, за да може ако това се случи, програмата да му изведе подходящо съобщение и от съображение на удобство, не се почва отначало въвеждането, а само трябва да въведе компонента, в който е допуснал грешката. (*Фрагмент 2*)

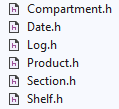
Относно грешките, които могат да възникнат в информационната система при работа с файловете също не остават по назад, тъй като е много важно информацията да се записва винаги точно и на правилното място в тях. (*Фрагмент 3*)

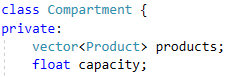




# Реализация, тестване

1. 1. Реализация на класове (включва важни моменти от реализацията на класовете и малки фрагменти от кода)

Създадените класове, както се подразбира от техните имена, са имплементации на компонентите, които задължително един склад трябва да има – секции, рафтове и отделение, като продуктът е сред най-важните, тъй като склада за благополучието на потребителя, не трябва да остава празен. Останалите класове – дата и запис, чрез тях информационната система определя кога е извършена каквато и да е промяна в наличността на склада, записвайки промените в отделен файл по отношение на този, където се записват продуктите. (Фрагмент 4)

****Това е малък фрагмент от класа Отделение представящ член данните му. Използвам последователният контейнер вектор, за да мога вътре да запазвам продуктите, които се добавят в склада и число с плаваща запетая за капацитета на отделението, за да се знае колко е заетото място в отделението. (Фрагмент 5)

* 1. Управление на паметта и алгоритми. Оптимизации

Върху динамичната памет има контрол, както още в изискванията бе споменато, че трябва, като все пак за някои операции има още какво да се желае от гледна точка на оптимална работа по отношение на паметта, която се заема от използваните масиви, вектори или собствени обекти.

* 1. Планиране, описание и създаване на тестови сценарии (създаване на примери)

За постигане на успешното внедряване на системата е необходимо много добре да се планират стъпките и дейностите в етапите на тестване и внедряване. Крайният сорс код ще се окомплектова заедно с помощните материали и публикува в GitHub, като същевременно е спазвано и качването на междинните разработки.

За преглед - [*https://github.com/FantomGSS/Storage*](https://github.com/FantomGSS/Storage)

С цел проверка на постигнатата реализация на приложението с необходимите критерии по задание, е необходимо да се изпълняват минимум следните функционални тестови сценарии (*Таблица 2*) и следователно още по времето на създаване на всяка една функционалност са направени нужните тестове, за да се потвърди тяхното правилно работене. Не винаги се получаваше от първия път, но по този начин излизаха и още нови идеи, по които по-лесно и стабилно да се направи задачата.



|  |  |  |  |
| --- | --- | --- | --- |
| **номер** | **Функция** | **Описание** | **Резултат (да/не)** |
| 1 | print | Извежда информация за наличните продукти в склада. | да |
| 2 | print | При извеждане на списъка с налични продукти да се изведе общото количество на едноименните продукти, независимо от срока им на годност. | да |
| 3 | add | Добавя нов продукт в склада в диалогов режим, като въвежда всички данни за продукта и ги проверява за коректност. | да |
| 4 | add | Проверка за условие: ако нов продукт е с различен срок на годност от вече съществуващ едноименен продукт, той да бъде поставен на различно място. | да |
| 5 | add | Проверка за условие: ако имате достатъчно място, еднакви продукти с един и същи срок на годност да бъдат поставени на едно и също място. | да |
| 6 | remove | Изважда продукт от склада в диалогов режим: по дадено име и количество изважда съответните продукти от склада и извежда информацията за продукта и къде се е намирал Изважда продукт от склада в диалогов режим. | да |
| 7 | remove | При наличие на повече от една партида, първо намалява тази със най-скоро изтичащ срок на годност, тогава във информацията за извършеното действие се отбелязва количеството и мястото на всяка от партидите, които сме намалили. | да |
| 8 | remove | В случай на опит за изваждане на повече от наличното да се дава информация на потребителя за наличността на продукта и срока на годност на партидите и възможност да реши дали все пак не иска да извади това което е останало. | да |
| 9 | log <from> <to> | Извежда справка за всички промени в наличността в периода от дата <from> до дата <to>, включително зареждания и извеждания на стоки. | да |
| 10 | clean | Разчиства склада от всички стоки, на които е изтекъл или предстои скоро да изтече срока на годност, като извежда информация за разчистените стоки | да |
| 11 | losses | По въведен продукт и неговата цена, за брой или съответно килограм, да се пресметнат загубите за даден от потребителя период (изхвърленото количество продукт със развален срок на годност) | да |

По-долу е представено с няколко примера работата на приложението, покривайки основните функционалности, които се изискваха от задачата. Входа от потребителя е означен с долар ($) в примерите.

*Пример 1:*

Enter the operation you want to perform.

The options regarding working with files are: open, close, save, save as, help and exit.

Storage options: add, remove, log, clean, losses.

Command: $ save

To execute this command, you must first open the file!

Command: $ open

You forgot to enter a path!

Command: $ open products.txt

===========================================================

Successfully created products.txt!

===========================================================

Command: $ add

Name: $ sprite

Expiry Date: $ 2020-06-2020

You have entered an invalid day. Please try again!

Expiry Date: $ 2020-06-25

Manufacturer's Name: $ Coca-Cola

Unit of Measure: $ 56

The allowed units of measure are kg and l!

Unit of Measure: $ l

Quantity: $ 56

Comment: $ Fresh and biting!

The product has been successfully added to the storage!

Command: $ add

Name: $ potatoes

Expiry Date: $ 2020-05-24

This expiry date expires today. Are you sure you want to enter it ?

Answer: $ No

In that case, it would be better to enter a new date!

Expiry Date: $ 2020-06-10

Manufacturer's Name: $ Kompir

Unit of Measure: $ kg

Quantity: $ 90

Comment: $ HIGH QUALITY POTATOES FROM SAMOKOV SCROLL DOWN

The product has been successfully added to the storage!

Command: $ save

===========================================================

All changes are saved successfully in the file products.txt!

===========================================================

Command: $ exit

*Пример 2:*

Enter the operation you want to perform.

The options regarding working with files are: open, close, save, save as, help and exit.

Storage options: add, remove, log, clean, losses.

Command: $ open products.txt

===========================================================

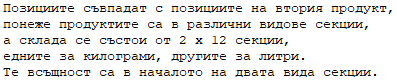
Successfully opened products.txt!

===========================================================

Command: $ print

Name: potatoes

Quantity: 90kg

Position: [1/1/1], [1/1/2], [1/1/3]

Name: sprite

Quantity: 56l

Position: [1/1/1], [1/1/2]

Command: $ remove

Name: $ potatoes

Quantity: $ 37

The quantity of product (40kg) in section 1, shelf 1, compartment 1 of the lot 2020-06-10 was reduced by 37kg

Successful rearrangement completed!

Command: $ print

Name: potatoes

Quantity: 53kg

Position: [1/1/1], [1/1/2]

Name: sprite

Quantity: 56l

Position: [1/1/1], [1/1/2]

Command: $ remove

Name: $ potatoes

Quantity: $ 500

The product cannot be removed because the quantity you entered exceeds available in the storage. You can only remove: 53kg

Lot 1 of the product with a quantity of 53 with an expiry date: 2020-06-10

Do you want to remove the quantity that is available in the storage from this product ?

Answer: $ Yes

The product has been removed successfully!

Command: $ print

Name: sprite

Quantity: 56l

Position: [1/1/1], [1/1/2]

Command: $ save

===========================================================

All changes are saved successfully in the file products.txt!

===========================================================

Command: $ exit

Пример 3:

Enter the operation you want to perform.

The options regarding working with files are: open, close, save, save as, help and exit.

Storage options: add, remove, log, clean, losses.

Command: $ open products.txt

===========================================================

Successfully opened products.txt!

===========================================================

Command: $ log

Enter the start and end dates of the period from which you want to receive information!

From: $ 2020-05-20

To: $ 2020-05-29

2020-05-24 - 56l of sprite were added to the storage.

2020-05-24 - 90kg of potatoes were added to the storage.

2020-05-24 - 37kg of potatoes were removed from the storage.

2020-05-24 - 53kg of potatoes were removed from the storage.

Command: $ add

Name: $ sprite

Expiry Date: $ 2020-05-25

Manufacturer's Name: $ Coca-Cola

Unit of Measure: $ l

Quantity: $ 24

Comment: $ Fresh and biting

The product has been successfully added to the storage!

Command: $ print

Name: sprite

Quantity: 80l

Position: [1/1/1], [1/1/2], [1/1/3]

Command: $ save

===========================================================

All changes are saved successfully in the file products.txt!

===========================================================

Command: $ clear

This command is not recognized, please make sure you have entered the command correctly!

Command: $ clean

The product sprite with quantity 24 at position [1/1/3] has been cleared!

Successful clearing and rearranging!

Command: $ save

===========================================================

All changes are saved successfully in the file products.txt!

===========================================================

Command: $ losses

Name: $ sprite

Price(BGN) per 1kg(l): $ 1.056

The program works with prices limited to the second decimal places!

Price(BGN) per 1kg(l): $ 1.5

Enter the start and end dates of the period from which you want to receive information about the losses!

From: $ 2020-05-20

To: $ 2020-05-29

The losses incurred are equal to the value of: 36 BGN.

Command: $ exit

# Заключение

1. 1. Обобщение на изпълнението на началните цели

Като цяло, проекта беше интересен. Трябваше да се запозная първо с подобни типове програми за складове как работят, за да си направя своя версия и накрая да я реализирам с помощта на досегашните ми знания. Лично аз съм доволен от своята работа, тъй като имах доста идеи и успях да реализирам най-добрите, постигайки стабилна работа и коректно изпълняване на поставените цели.

* 1. Насоки за бъдещо развитие и усъвършенстване

Като възможно усъвършенстване на приложението в бъдеще, е оптимизацията на паметта.

# Използвана литература

*Използван алгоритъм за изчисляване на дата –* [*https://stackoverflow.com/questions/8343676/how-to-get-current-date-and-time*](https://stackoverflow.com/questions/8343676/how-to-get-current-date-and-time)

*Използван продукт за схеми по фиг.1 и 2 – ARIS Express* [*https://www.ariscommunity.com/aris-express*](https://www.ariscommunity.com/aris-express)