Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина: Технология разработки программного обеспечения

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему

**Разработка программы учета выпускаемой**

**предприятием продукции**

Студент:

гр.­­­­­ 113802 Скачко А. Г.

Руководитель:

к.т.н. Кабариха В.А.

Минск 2022

# Содержание

Содержание 2

1 Требования к программе 3

1.1 Исходные требования к курсово работе 3

1.2 Функциональные требования 3

1.3 Требования к програмнной реализации 7

2 Конструирование программы 9

2.1 Разработка структуры программы 9

2.2 Выбор способа организации данных 9

2.3 Разработка перечня пользовательских функций программы 11

3 Разработка алгоритмов работы программы 18

3.1 Алгоритм функции main 18

3.2 Алгоритм функции editProduct 18

3.3 Алгоритм функции writeToConsole 18

4 Описание работы программы 19

4.1 Вход в программу 17

4.3 Модуль администратора 20

4.4 Модуль пользователя 30

4.5 Исключительные ситуации 38

Приложение А (обязательное) Модульная структура программы 46

Приложение Б (обязательное) Алгоритм функции main 47

Приложение В (обязательное) Алгоритм функции editProduct 48

Приложение Г (обязательное) Алгоритм функции writeToConsole 49

Приложение Д (обязательное) Листинг кода с комментариями 50

# Требования к программе

Разработать программу учета выпускаемой предприятием продукции.

Сведения о выпущенной продукции включают: дату; номер цеха; наименование продукции; количество выпущенных единиц; ФИО ответсвенного по цеху в данный день

Индивидуальное задание: для заданного цеха необходимо вывести количество выпущенных изделий по каждому наименованию за требуемый период времени (требуемый период времени вводится с клавиатуры).

Реализовать авторизацию для входа в систему, функционал администратора и функционал пользователя, как минимум три вида поиска, как минимум три вида сортировки.

## 1.1 Исходные требования к курсовой работе

1. Язык программирования С++.
2. Среда разработки Microsoft Visual Studio версии 2010 и выше.
3. Вид приложения – консольное.
4. Парадигма программирования – процедурная.
5. Способ организации данных – структуры.
6. Способ хранения данных – текстовые файлы.
7. Каждая логически завершенная задача программы должна быть реализована в виде функции.
8. К защите курсовой работы предоставляются: консольное приложение и пояснительная записка.
9. Текст пояснительной записки оформляется в соответствии со стандартом предприятия «СТП 01–2017».

## 1.2 Функциональные требования к курсовой работе

В начале программы осуществляется чтение информации из файлов в массивы, далее все операции с данными производятся посредством массивов, однако при каждом изменении информации дополнительно происходит запись информации из массива в файл.

Первым этапом работы программы является авторизация – предоставление прав доступа. В рамках данного этапа необходимо считать данные из файла, содержащего учетные записи пользователей следующего вида:

* login;
* hash;
* salt;
* role;
* access.

После ввода пользователем своих персональных данных (логина и пароля) и сверки с данными, находящимися в файле учетных записей, необходимо предусмотреть возможность входа в качестве администратора (role = 1) или в качестве пользователя (role = 0).

Если файла с учетными записями не существует, то необходимо его программно создать и сделать учетную запись главного администратора.

Регистрация новых пользователей осуществляется самим пользователем путем ввода желаемых логина и пароля, и ожидания подтверждения администратором новой учетной записи или главным администратором при редактировании учетной записи. Для реализации этого способа в структуре учетных записей аккаунтов предусмотрено поле access. По умолчанию access = 0 при попытке зарегистрироваться; далее администратор при подтверждении заявки меняет значение на access = 1 и тем самым подтверждает новую учетную запись; пользователь может осуществить вход в систему. Также реализована проверка новых учетных записей на уникальность логина.

По соображениям безопасности пароли учетных записей за хешированы. Выполнено двойное хеширование пароля с солью (соль – случайная строка, специально сгенерированная для данной учетной записи). И соль, и результат применения хеш-функции к паролю с солью храниться в файле с учетными записями.

Вторым этапом работы программы является собственно работа с данными, которая становится доступной только после прохождения авторизации. Данные хранятся в отдельном файле продукции.

Для работы с данными должны быть предусмотрены два функциональных модуля: модуль главного администратора и модуль пользователя.

Допускается создавать несколько учетных записей администраторов, только в том случаем, если администратор существующей учетной записи меняет role = 1 вручную.

Модуль администратора включает следующие подмодули (с указанием функциональных возможностей):

1. Работа с учетными записями:

* Просмотр всех учетных записей;
* Редактирование учетных записей:

1. Изменение логина;
2. Изменение роли;
3. Одобрение доступа пользователя;

* Удаление учётной записи;

1. Обработка данных продукции:

* Просмотр информации об всей продукции (выполняется автоматически);
* Поиск:

1. Поиск по году, когда продукт был произведён;
2. Поиск по названию продукции;
3. Поиск по количеству выпущенных единиц;
4. Поиск по имени ответсвенного по цеху в этот день;

* Сортировка:

1. Сортировка по году, когда продукция была произведена;
2. Сортировка по номеру цеха;
3. сортировка по количеству выпущенных единиц;

3. Вывод количество выпущенных изделий по определённому наименовани

Модуль пользователя включает подмодуль работы с данными со следующими функциональными возможностями:

1. Просмотр информации обо продукции;
2. Вывод количество выпущенных изделий по определённому наименованию
3. Поиск:

* поиск по году, кода продукт был произведён;
* поиск по названию продкции;
* поиск по колчеству выпущенных единиц;
* поиск по имени ответсвенного по цеху в этот день;

1. Сортировка:

* сортировка по году, когда продукция была произведена;
* сортировка по номеру цеха;
* сортировка по количеству выпущенных единиц;

1. Изменение собственного пароля;

Для реализации перечисленных модулей/подмодулей необходимо создавать меню с соответствующими пунктами.

Предусмотреть:

1. Обработку исключительных ситуаций:
   * файла с данными для чтения не существует;
   * пользователь с таким именем уже существует;
   * ничего не найдено по результатам поиска;
   * номер удаляемой записи, выбор действия, номер редактирования записи и т.д. выходит за пределы массива.
   * введенные данные пользователем нелогичны (отрицательная цена, несуществующий возраст и т.д.);
2. Возможность возврата назад (навигация);
3. Обратная связь с пользователем, вывод сообщения об успешности удаления/добавления/редактирования записи.

## 1.3 Требования к программной реализации

1. Все переменные и константы должны иметь осмысленные имена в рамках тематики варианта к курсовой работе. Переменным рекомендуется присваивать имена, состоящие из букв нижнего регистра; для формирования составного имени используется нижнее подчеркивание (например, number\_of\_students) или “верблюжья аннотация”. Константам рекомендуется присваивать имена, состоящие из букв верхнего регистра (например, SIZE\_ARR\_OF\_ACCOUNTS).
2. Имена функций должны быть осмысленными и строится по принципу «глагол + существительное». Если функция выполняет какую-либо проверку и возвращает результат типа bool, то ее название должно начинаться с глагола is (например, isFileExist, isUnicLogin).
3. Не допускается использование оператора прерывания goto.
4. Код не должен содержать неименованных числовых констант (так называемых «магических» чисел), неименованных строковых констант (например, имен файлов и др.). Подобного рода информацию следует выносить в глобальные переменные с атрибутом const. По правилам хорошего стиля программирования тексты всех информационных сообщений, выводимых пользователю в ответ на его действия, также оформляются как константы.
5. Код необходимо комментировать (как минимум в части нетривиальной логики).
6. Код не должен дублироваться – для этого существуют методы и функции.
7. Одна функция решает только одну задачу (например, не допускается в одной функции считывать данные из файла и выводить их на консоль – это две разные функции). При этом внутри функции возможен вызов других функций.
8. Выполнение операций чтения/записи/ в файл должно быть сведено к минимуму (т.е. после однократной выгрузки данных из файла в массив дальнейшая работа ведется с этим массивом, а не происходит многократное считывание данных из файла в каждой функции).
9. Следует избегать длинных функций и глубокой вложенности: текст функции должен умещаться на один экран, а вложенность блоков и операторов должна быть не более трёх.
10. Следует избегать длинных функций: текст функции должен умещаться на один экран (размер текста не должен превышать 25-50 строк).
11. Минимизировать область видимости переменной, сделав ее как можно более локальной.
12. Следует выносить код логически независимых модулей в отдельные .cpp файлы и подключать их с помощью заголовочных .h файлов.

# Конструирование программы

Реализация программы будет осуществляться на языке С++ в IDE-среде Microsoft Visual Studio 2022. Программа будет компилироваться и использоваться в операционных системах семейства Microsoft Windows версии 7 и выше.

## 2.1 Разработка структуры программы

Согласно требованиям к программе, необходимо наличие исполняемой программы, которая работает с файлом пользователей workingWithUsers и файлом продуктов workWithData. Следовательно, с точки зрения верхней архитектуры программы, можно выделить два основных модуля: модуль работы с массивом продуктов arrayOfProduct и модуль работы с массивом учетных записей arrayOfUsers.

В приложении А показана модульная структура программы.

Функция авторизации пользователя заключается в проверке существования в файле учетных записей введённого логина и соответствующего ему пароля, а также наличие доступа для данной учетной записи. Также авторизация пользователя подразумевает получение его роли из файла и предоставление ему соответствующего функционала (функционала пользователя, или функционала администратора, или функционала главного администратора).

После успешной авторизации пользователя создаётся пользовательская сессия с соответствующими привилегиями, согласно роли пользователя. Сессия администратора имеет доступ к модулю управления файлом аккаунтов, и файлов продукта, а сессия пользователя, в свою очередь, имеет доступ только к модулю управления массивом продукта на складе (только обработка информации).

## 2.2 Выбор способа организации данных

Для представления в программе объекта аккаунтов вводится структура account, содержащая nickname, saltedHashPassword, salt, access и role аккаунта.

* saltedHashPassword - осуществляет преобразование массива входных данных произвольной длины в выходную битовую строку установленной длины, выполненную определённым алгоритмом.
* salt - строка данных (16 символов), которая передаётся хеш-функции вместе с входным массивом данных для вычисления хэша.
* nickname - персональное имя для входа в программу
* access - доступ аккаунта, если “0” – у данного аккаунта доступа нет, он не может войти в программу, если “1” – у данного аккаунта есть доступ и при правильной авторизации (ввода логина и пароля) он может войти в систему.
* role - роль аккаунта. В зависимости от роли выбирается определенный функционал для данного пользователя: 1) “0” - имеет доступ только к модулю управления массивом продуктов (только обработка информации); 2) “1” - администратор имеет доступ к модулю управления файлом учетных записей arrayOfUsers и к обработке информации из массива продуктов arrayOfProduct.

Для представления в программе данных о продукте вводится структура ProductInfo, содержащая dayWhenProductCreate, workShopNumber, productName, numberOfProductsProductd, responsiblePerson и подструктуру Date, содержащую day, month, year:

* dayWhenProductCreate – день, месяц и год создания продукта.
* workshopNumber – номер цеха.
* productName – название продукта.
* numberOfProductsProduced – количество продукта
* responsiblePerson – имя ответсвенного по цеху в этот день
* подструктура Date – дата создание продукта: day – день создания, month – месяц создания, year – год создания.

Программа обслуживает данные, хранимые в двух файлах: файле учетных записей account и файле продукта productFile. На рисунке 1 показано хранение учетных записей в файле account. На рисунке 2 показано хранение данных продуктов в файле productFile.

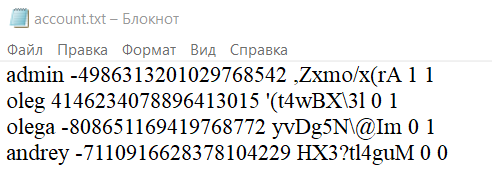


Рисунок 1 –Хранение учетных записей в файле workWithUsers

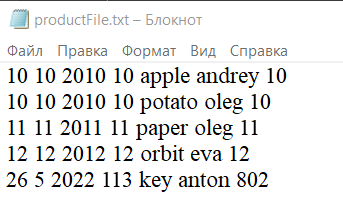


Рисунок 2 – Хранение данных продуктов в файле workWithData

В качестве способа объединения входных данных используются динамически создаваемые массивы:

* для структуры Users используется массив arrayOfUsers, а количество учетных записей в массиве обозначается переменной sizeArrayOfUsers;
* для структуры Product используется массив arrayOfProduct, а количество продукта в массиве обозначается переменной sizeArrayOfData;

Для хранения текстовых данных выбрана кодировка ANSI.

## 2.3 Разработка перечня пользовательских функций программы

**Выход из программы –** функция выхода из программы, при выборе этой функции.

**Регистрация –** функция регистрации, пользователь вводит желаемый логин и пароль и при корректно введенных данных, по умолчанию присваивается введенный логин, хешируется пароль, предоставляется роль пользователя без доступа. Далее происходит ожидание подтверждения заявки от администратора или главного администратора. Заявка вноситься в массив workWithUsers и храниться в файле с учетными записями, до тех пор, пока ее не удалит администратор. В случае подтверждения администратором заявки или редактирования учетной записи администратором, пользователь получает доступ для пользования программой в функционале пользователя.

**Авторизация –** функция авторизации заключается в проверке существования в файле аккаунтов введённого логина и соответствующего ему пароля (хеша). Также авторизация подразумевает получение роли из файла и при наличии доступа (access = “1”) у данной учетной записи предоставляется ему соответствующий функционал (функционал пользователя, функционал администратора). В случае некорректно введённых данных, либо же данных не имеющих доступа, вход в программу не возможен, пользователю предоставляется попытка повторного ввода либо же возврат в главное меню.

**Модуль пользователя** (предоставляется меню возможного функционала для пользователя)**:**

1. **Просмотр информации об всех продуктах –** пользователь может просмотреть данные обо всех пациентах (дата производства, номер цеха, название продукта, количество продукта, имя ответвенного человека в этот день).
2. **Поиск -** предоставляется меню поиска (**поиск по дню производства продукта, номеру цеха, имени продукта, количеству продукта, имени ответсвенного в день производства продукта),** выбирается вид поиска, вводятся данные, которые необходимо найти, в случае успешного поиска выводиться найденная информация, в случае некорректно введённых данных либо же если данные не найдены, то предлагается выбор на повторный поиск или возврат в главное меню пользователя.
3. **Сортировка –** предоставляется меню сортировки (**сортировка по дню производства продукта, номеру цеха, имени продукта, количеству продукта, имени ответсвенного в день производства продукта),** выбирается по какому признаку сортировать данные, затем предоставляется выбор вида сортировки от А до Я, либо от Я до А, либо же вернуться назад в главное меню сортировки, в зависимости от выбора выводиться соответствующая информация.
4. **Выход из функционала пользователя –** при подтверждении действия происходит выход из аккаунта пользователя и возврат в главное меню.

**Модуль администратора** (предоставляется меню возможного функционала для администратора)**:**

* **Работа с данными продукта** (предоставляется меню возможного функционала для работы с данными продукта):

1. **Просмотр –** администратор может просмотреть данные обо всех продуктах (дата производства, номер цеха, название продукта, количество продукта, имя ответвенного человека в этот день).
2. **Добавление –** администратор может добавлять новые продукты вручную. Вводятся все необходимые данные и проверяются на корректный ввод. В случае корректно введенных данных, поступает запрос на подтверждение действия. В случае подтверждения действия, размер массива patients увеличивается, данные присваиваются в структуру и записываются в файл и массив workWithProduct, в случае отказа на добавление размер массива остается прежним и данные не добавляются в массив.
3. **Удаление –** администратор может удалять существующих продукты. Выводятся все продукты из массива workWithProduct, он выбирает какого именно продукт удалить. Поступает запрос на подтверждения действия, размер массива уменьшается, и данный продукт удаляется из массива и файла productFile, в случае отказа на удаление размер массива остается прежним, и выбранный пациент не удаляется.
4. **Редактирование –** выводятся все продукта из массива workWithProduct. Он выбирает данные какого продукта ему нужно отредактировать, выводиться вся информация о выбранном продукте и меню выбора редактирования:

* **Редактирование по дате производства, номеру цеха, названию продукта, количеству продукта, ответвенного за производсто продукта–** администратор вводит новые данные. При корректном вводе, предоставляется запрос на подтверждения действия. В случае подтверждения действия данные перезаписываются выбранному продукта в массив workWithProduct и файл productFile, в случае отказа данные не изменяются.

1. **Выход из работы с данными об продукте –** происходит выход из работы с данными об продукте и перенаправление в главное меню администратора.

* **Работа с учетными записями** (предоставляется меню возможного функционала для работы с учетными записями):

1. **Просмотр всех учетных записей –** выводятся данные (логин, доступ, роль) всех учетных записей в массиве workWithUsers.
2. **Подтверждение –** выводятся все учетные записи без доступа (access = “0”). Администратор выбирает какую заявку одобрить, предоставляется запрос на подтверждения действия. В случае подтверждения данной учетной записи присваивается доступ (access = “1”) и изменения перезаписываются в массив и файл workWithUsers и пользователю данной учетной записи предоставляется возможность входа в программу, в случае отказа выбранная учетная запись не активируется.
3. **Блокировка –** выводятся все учетные записи с доступом (access = “1”). Администратор выбирает какую заявку заблокировать и предоставляется запрос на подтверждение действия. В случае подтверждения данная учетная запись блокируется (access = “0”) и изменения перезаписываются в массив и файл account, и пользователь с этой учетной записью не может войти в программу, в случае отказа выбранная учетная запись не блокируется.
4. **Выход из работы с учетными записями –** происходит выход из работы с учетными записями и перенаправление в главное меню администратора

**– Обработка информации о продукта** (предоставляется меню возможного функционала для обработки информации о пациентах)**:**

1. **Просмотр информации об всех продуктах –** администратор может просмотреть данные обо всех пациентах (дата производства, номер цеха, название продукта, количество продукта, имя ответвенного человека в этот день).
2. **Поиск -** предоставляется меню поиска (**поиск по дню производства продукта, номеру цеха, имени продукта, количеству продукта, имени ответсвенного в день производства продукта),** выбирается вид поиска, вводятся данные, которые необходимо найти, в случае успешного поиска выводиться найденная информация, в случае некорректно введённых данных либо же если данные не найдены, то предлагается выбор на повторный поиск или возврат в главное меню пользователя.
3. **Сортировка –** предоставляется меню сортировки (**сортировка по дню производства продукта, номеру цеха, имени продукта, количеству продукта, имени ответсвенного в день производства продукта),** выбирается по какому признаку сортировать данные, затем предоставляется выбор вида сортировки от А до Я, либо от Я до А, либо же вернуться назад в главное меню сортировки, в зависимости от выбора выводиться соответствующая информация.

**– Выход из функционала администратора –** при подтверждении действия происходит выход из аккаунта и возврат в главное меню.

# Разработка алгоритмов работы программы

## 3.1 Алгоритм функции main

Функция main является точкой входа в программу, вызывает функцию авторизаии. В приложении Б показан алгоритм данной функции.

## Алгоритм функции editProduct

Функция editProduct является функцией редактирования информации об продукте. В приложении В показан алгоритм данной функции.

## 3.3 Алгоритм функции

Функция deletePatient является функцией удаления пациента из массива и файла. Алгоритм данной функции показан в приложении Г.

# описание работы программы

## 4.1 Вход в программу

При запуске программа пытается найти и открыть файлы с учетными записями и данными о продукте. Если файл найден, программа начинает выводить Главное меню.

* **Главное меню входа** - после успешного входа в программу, программа запускает диалоговое взаимодействие с пользователем путём отображения пронумерованных пунктов меню и запросом у пользователя ввода желаемого номера (рис.8).

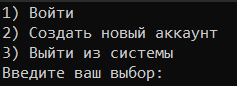


Рисунок 8 – Главное меню входа

* **Войти** - пользователю предоставляется возможность ввести логин и пароль для входа в программу. При корректном вводе и при правильном вводе пароля и логина, пользователь заходит в свой аккаунт, с функционалом соответствующей роли (рис.9).



Рисунок 9 – Авторизация

* **Регистрация -** пользователю предоставляется возможность зарегистрироваться в программу. Он вводит логин, пароль и повторяет пароль для подтверждения. При корректном вводе данных и при вводе уникального логина, пользователь записывается в файл учетных записей и автоматически подается заявка на активацию аккаунта (рис.10).

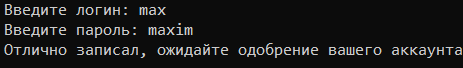


Рисунок 10 – Регистрация

* **Выход из программы -** при выборе выхода из программы поступает запрос на подтверждения действия и осуществляется выход из программы (рис.11).



Рисунок 11 – Выход из программы

## 4.2 Модуль администратора

После успешного входа в функционал администратора программа запускает диалоговое взаимодействие с администратором путём отображения пронумерованных пунктов меню и запросом у администратора ввода желаемого номера (рис.12).

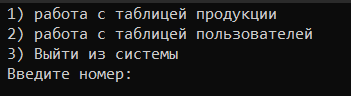


Рисунок 12 – Главное меню администратора

* **Работа с данными продукта -** программа запускает диалоговое взаимодействие с администратором путём отображения пронумерованных пунктов меню «Работы с данными продукта» и запросом у администратора ввода желаемого номера (рис.13).

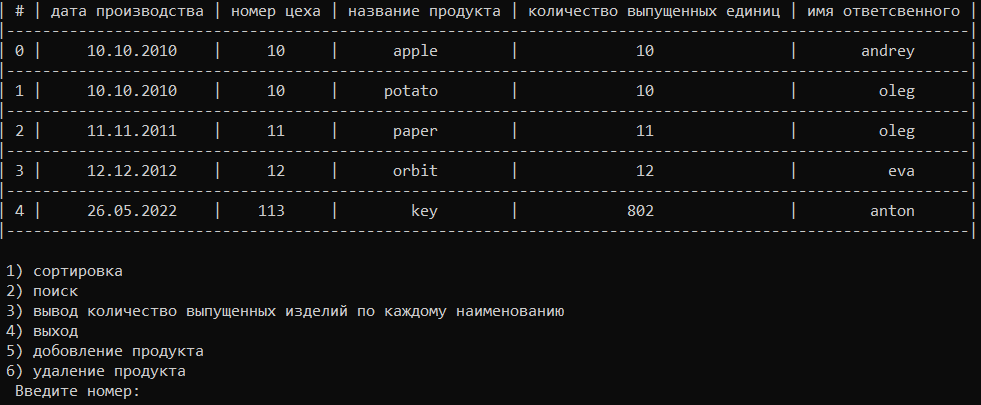


Рисунок 13 – Меню «Работы с данными пациентов» для администратора

1. **Просмотр всей продукции** – выводится информация о всех пациентах (рис.14).

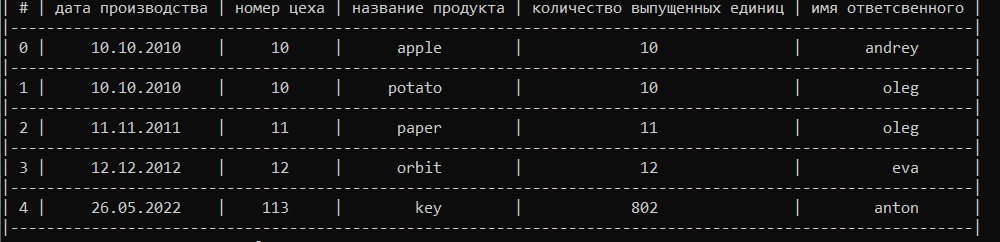


Рисунок 14 – Вывод всей продукции

1. **Добавить продукт –** нужно заполнить корректно все данные, а затем поступает запрос на подтверждение действия (рис.15).

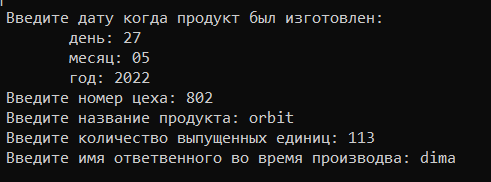
****

Рисунок 15 – Добавление пациента

1. **Удалить продукт -** выводится информация о всех продуктах, и администратор выбирает номер пациента, которого следует удалить (рис.16).

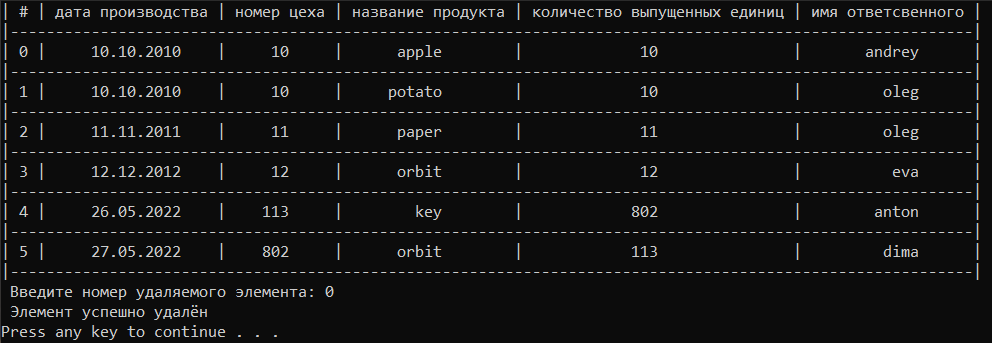


Рисунок 16 – Удаление продукта

Затем поступает запрос на подтверждения удаления данного п(рис.17).

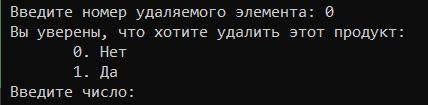


Рисунок 17 – Подтверждение удаления данного продукта

**4) Редактирование продукта -** выводится информация о всех проду

тах, и администратор выбирает номер пациента, которого хочет отредактировать (рис. 18).

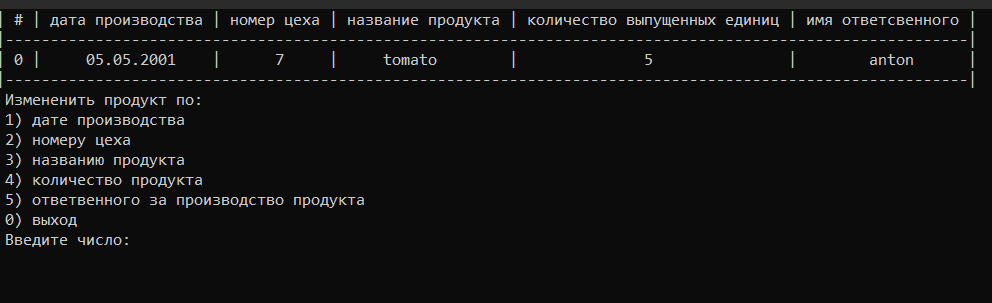


Рисунок 18 - Информация о всех продукта для редактирования

Затем выводятся данные продукта которого будем редактировать, меню редактирования для пациента и производиться ввод критерия редактирования (рис.18).

* **Работа с учетными записями -** программа запускает диалоговое взаимодействие с администратором путём отображения пронумерованных пунктов меню «Работы с учетными записями» и запросом у администратора ввода желаемого номера (рис.19).

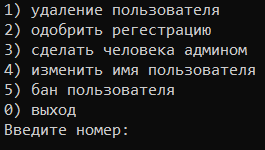


Рисунок 19 – Меню «Работы с учетными записями» для администратора

1. **Просмотр аккаунтов -** выводится информация о всех учетных записях (логин, доступ, роль, хешированный пароль) (рис.20).

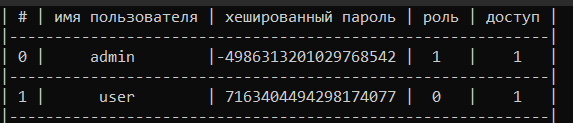


Рисунок 20– Просмотр информации о всех учетных записях

1. **Активация аккаунта –** выводятся все аккаунты без доступа входа в программу, и администратор выбирает какую заявку одобрить (дать доступ учетной записи)(рис.21).

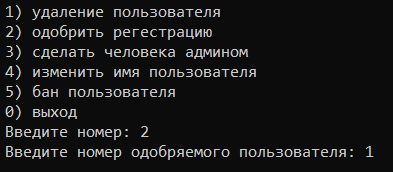


Рисунок 21 – Активация аккаунта

1. **Блокировка аккаунта -** выводятся все аккаунты с доступом для входа в программу, и администратор выбирает какой аккаунт заблокировать (без доступа), затем поступает запрос на подтверждения действия для блокировки данного аккаунта (рис.22).

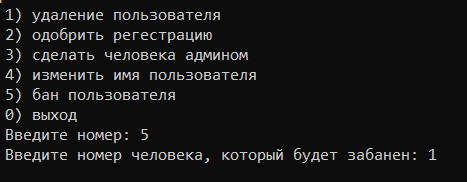


Рисунок 22 – Блокировка аккаунта

* **Обработка данных продукта –** снимки экрана обработки данных о продукта соответствуют снимкам в модуле главного администратора.
* **Выход из функционала администратора -** снимки экрана выхода из функционала администратора соответствуют снимкам в модуле главного администратора.

## 4.3 Модуль пользователя

После успешного входа в функционал пользователя программа запускает диалоговое взаимодействие с пользователем путём отображения пронумерованных пунктов меню и запросом у пользователя ввода желаемого номера (рис. 23).

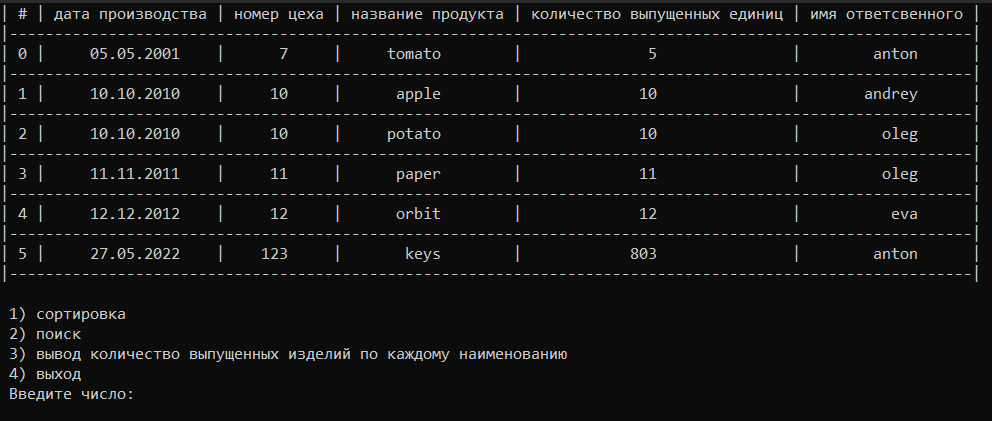
****

Рисунок 23 – Главное меню пользователя

* **Просмотр всей продукции –** снимки экрана просмотра всей продукции совпадают со снимками экрана в модуле главного администратора.
* **Поиск данных –** снимки экрана поиска данных совпадают со снимками экрана в модуле главного администратора.
* **Сортировка данных –** снимки экрана сортировки данных совпадают со снимками экрана в модуле главного администратора.
* **Выход из функционала пользователя –** снимки экрана выхода из функционала пользователя совпадают со снимками экрана в модуле главного администратора.

## 4.4 Исключительные ситуации

Обработка исключительных ситуаций, как правило, занимает существенную часть кода программы. Это неизбежно, потому что программа без обработки ошибок и пограничных ситуаций выглядит для пользователя некачественной и может приводить к множеству негативных последствий: повреждению данных, неверной трактовке результатов работы программы, нагрузкой на службу поддержки или разработчика и т.п. В связи с этим, в программу встроено достаточно много проверок данных и результатов выполнения функций, обработок программных исключений, сопровождающихся уведомлением пользователя о проблеме и, при возможности, логичной реакцией на ситуацию либо прерыванием работы программы, если программа не смогла обработать исключительную ситуацию.

На рисунке 24 показана первая, простейшая исключительная ситуация: введён неверный логин пользователя, либо пароль, либо у данного аккаунта нет доступа для входа в программу. Программа сообщает об этом пользователю и предлагает повторить попытку либо вернуться в главное меню авторизации.



Рисунок 24– Неверный ввод логина или пароля при входе

Во всех меню при вводе несуществующего пункта меню программа выведет соответствующее сообщение и попросит повторить ввод либо вернуться назад. На рисунке 25 показан пример отображения ошибки при вводе несуществующего пункта меню.

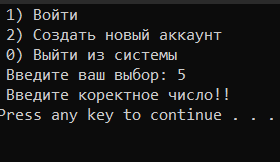


Рисунок 25 – Неверный ввод пункта меню

При вводе пользователем несуществующего номера учетной записи или номера продукта для таких операций, как редактирование, удаление, блокировка, активация и т.д., программа выведет соответствующее сообщение. На рисунке 26 показан пример отображения ошибки при вводе несуществующего номера пациента.

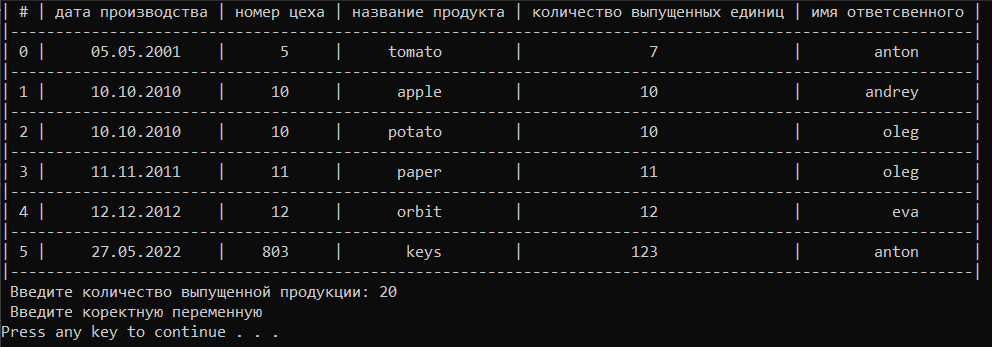


Рисунок 26 – Ошибка при вводе несуществующего номера продукта или аккаунта

На рисунке 27 показан пример вывода сообщения при попытке создания новой учетной записи (регистрации) с логином, который уже существует (занят), программа выведет соответствующее сообщение об ошибке.



Рисунок 27 – Занятый логин

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Модульная структура программы (к пункту 2.1)**



Рисунок 3 – Модульная структура программы

**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Алгоритм функции main (к пункту 3.1)**



Рисунок 4 – Блок-схема алгоритма функции main

**ПРИЛОЖЕНИЕ В**

**(обязательное)**

**Алгоритм функции surnamePatintsEdit (к пункту 3.2)**



Рисунок 5 – Блок-схема алгоритма функции surnamePatientsEdit

**ПРИЛОЖЕНИЕ Г**

**(обязательное)**

**Алгоритм функции deletePatient (к пункту 3.3)**



Рисунок 6 – Блок-схема алгоритма функции deletePatient

**ПРИЛОЖЕНИЕ Д**

**(обязательное)**

**Листинг кода с комментариями**

**Файл Main.cpp**

\*\*/

// подключаем заголовочные файлы

#include "outputToConsole.h"

#include "workingWithUsers.h"

#include "workWithData.h"

#include <iostream>

using namespace std;

int main() {

// подключение русского языка

setlocale(LC\_ALL, "Russian");

int sizeArrayOfData = 1, sizeArrayOfUsers = 1;

ProductInfo\* arrayOfData = new ProductInfo[sizeArrayOfData];

readProductFile(arrayOfData, sizeArrayOfData);

Users\* arrayOfUsers = new Users[sizeArrayOfUsers];

readUserFile(arrayOfUsers, sizeArrayOfUsers);

// вывод

writeToConsole(arrayOfData, arrayOfUsers, sizeArrayOfData, sizeArrayOfUsers);

return 0;

}

**Основыне функции из файла outputToConsole.cpp**

void createNewAccount(Users\*& arrayOfUsers, int &sizeArrayOfUsers) {

system("cls");

string password, username;

int countOfMistake = 3;

arrayOfUsers = resizeUserArray(arrayOfUsers, sizeArrayOfUsers + 1, sizeArrayOfUsers);

while (countOfMistake != 0) {

cout << " Введите логин: "; cin >> username;

// проверка на правильность ввода имени

if (isUsernameCorrect(username, arrayOfUsers, sizeArrayOfUsers - 1)) {

cout << " этот логин занят, введите новый!!\n";

cout << " У вас осталось ввести " << countOfMistake << "раза\n";

countOfMistake--;

if (countOfMistake == 0) {

cout << " У вас не осталось попыток";

return;

}

}

else {

break;

}

}

cout << " Введите пароль: "; cin >> password;

// добавление в массив

getSalt(arrayOfUsers, password, sizeArrayOfUsers - 1);

password += arrayOfUsers[sizeArrayOfUsers - 1].salt;

arrayOfUsers[sizeArrayOfUsers - 1].saltedHashPassword = hashing(password);

arrayOfUsers[sizeArrayOfUsers - 1].role = 0;

arrayOfUsers[sizeArrayOfUsers - 1].access = 0;

arrayOfUsers[sizeArrayOfUsers - 1].nickname = username;

cout << " Отлично записал, ожидайте одобрение вашего аккаунта \n";

}

// функционал админа

void adminOutput(ProductInfo\*& arrayOfProduct, Users\*& arrayOfUsers, int& sizeArrayOfData, int& sizeArrayOfUsers) {

int adminChoice, choiceOfProduct;

while (true) {

system("cls");

cout << " 1) работа с таблицей продукции \n 2) работа с таблицей пользователей \n"

<< "0) Выйти из системы \n Введите номер: ";

enum ADMIN\_CHOICE {

EXIT,

WORK\_WITH\_PRODUCT,

WORK\_WITH\_USERS

};

cin >> adminChoice;

switch (adminChoice) {

case WORK\_WITH\_PRODUCT:

while (true) {

system("cls");

productTableOutput(arrayOfProduct, sizeArrayOfData);

cout << "\n 1) сортировка \n 2) поиск \n 3) вывод количество выпущенных изделий по каждому наименованию \n"

<< "4) выход";

cout << "\n 5) добовление продукта\n 6) удаление продукта \n 7) редактировать данные\n";

cout << " Введите номер: ";

cin >> choiceOfProduct;

if (choiceOfProduct < 4) {

userOutput(arrayOfProduct, sizeArrayOfData, choiceOfProduct);

}

else if(choiceOfProduct == 4) {

cout << " Прощайте\n";

break;

}

else {

displayAdditionalFeatures(arrayOfProduct, sizeArrayOfData, choiceOfProduct);

}

}

break;

case WORK\_WITH\_USERS:

system("cls");

outputWorkdWithUsers(arrayOfUsers, sizeArrayOfUsers);

break;

case EXIT:

system("cls");

cout << "Прощайте\n";

return;

default:

cout << "Введите коректное число";

break;

}

}

}

// что должно выводить у обычного пошльзователя

void userOutput(ProductInfo\*& arrayOfProduct, int sizeArrayOfData, int choice) {

// sortingChoice и searchingChoice переменные для выбора метода сортировки и поиска соответсвенно

int sizeArrayOfProduct;

string nameOfProduct, nameOfResponsible;

enum CHOICE {

EXIT,

SORTING,

SEARCH,

INDIVIDUAL\_TASK

};

switch (choice) {

case SORTING:

outputSorting(arrayOfProduct, sizeArrayOfData);

break;

case SEARCH:

outputSearch(arrayOfProduct, sizeArrayOfData);

break;

case INDIVIDUAL\_TASK:

outputIndividualTask(arrayOfProduct, sizeArrayOfData);

break;

default:

system("cls");

cout << " Введите коректное число!";

}

}

**Основные функции из файла workWithUsers.cpp**

#include "outputToConsole.h"

string EnterPassword() {

string pass;

char symbol;

const char BACK\_SPACE = 0;

const char ENTER = 13;

while (true) {

// принимаем функции с консоли

symbol = \_getch();

if (symbol == ENTER) {

break;

}

else if (symbol == BACK\_SPACE) {

if (pass.size() > 0) {

pass.pop\_back();

cout << "\b \b";

}

}

else if (symbol > 31) {

cout << "\*";

pass += symbol;

}

else {

cout << "\b \b";

}

}

cout << "\n";

return pass;

}

string validation(Users\* arrayOfUsers, int sizeArrayOfUsers) {

int attempts = 3; // количество попыток у пользователя

string username, password;

while (true) {

system("cls");

cout << " Введите логин: "; cin >> username;

cout << " Введите пароль: ";

password = EnterPassword();

if (isPasswordCorrect(password, username, arrayOfUsers, sizeArrayOfUsers) &&

isUsernameCorrect(username, arrayOfUsers, sizeArrayOfUsers)) {

return username;

}

cout << " Неверный пароль или логин, повторыите попытку, у вас осталось попыток " << attempts << endl;

system("pause");

--attempts;

if (attempts == 0) {

cout << " Ваш лимит исчерпан";

break;

}

}

}

// если день или месяц является единичным числом, то к этому числу добавляется 0

string dayWithZero(int date) {

string resultString = "0";

if (date < 10) {

resultString += to\_string(date);

return resultString;

}

return to\_string(date);

}

// вывод таблицы продуктов склада

void productTableOutput(ProductInfo\* arrayOfProduct, int sizeArrayOfData) {

cout << "|" << setw(2) << "#" << setw(2) << "|" << setw(18) << "дата производства" << setw(2) << "|"

<< setw(11) << "номер цеха" << setw(2) << "|" << setw(18) << "название продукта" << setw(2) << "|"

<< setw(29) << "количество выпущенных единиц" << setw(2) << "|" << setw(18) << "имя ответсвенного" << setw(3) << "|\n";

for (int i = 0; i < sizeArrayOfData; i++) {

cout << "|-----------------------------------------------------------------------------------------------------------|\n";

cout << "|" << setw(2) << i << setw(2) << "|" << setw(7) << dayWithZero(arrayOfProduct[i].dayWhenProductCreate.day) << "." <<

dayWithZero(arrayOfProduct[i].dayWhenProductCreate.month) << "." << dayWithZero(arrayOfProduct[i].dayWhenProductCreate.year) << setw(5) << "|"

<< setw(7) << arrayOfProduct[i].workShopNumber << setw(6) << "|" << setw(11) << arrayOfProduct[i].productName << setw(9) << "|"

<< setw(15) << arrayOfProduct[i].numberOfProductsProduced << setw(16) << "|" << setw(13) << arrayOfProduct[i].responsiblePerson << setw(8) << "|\n";

}

cout << "|-----------------------------------------------------------------------------------------------------------|\n";

}

// вывод таблицы пользователей

void userTableOutput(Users\* arrayOfUsers, int sizeArrayOfUsers) {

cout << "|" << setw(2) << "#" << setw(2) << "|" << setw(17) << "имя пользователя" << setw(2) << "|"

<< setw(20) << "хешированный пароль" << setw(2) << "|"

<< setw(5) << "роль" << setw(2) << "|" << setw(7) << "доступ" << setw(3) << "|\n";

for (int i = 0; i < sizeArrayOfUsers; i++) {

cout << "|------------------------------------------------------------|\n";

cout << "|" << setw(2) << i << setw(2) << "|" << setw(10) << arrayOfUsers[i].nickname << setw(9) << "|"

<< setw(20) << arrayOfUsers[i].saltedHashPassword << setw(2) << "|"

<< setw(3) << arrayOfUsers[i].role << setw(4) << "|" << setw(5) << arrayOfUsers[i].access << setw(5) << "|\n";

}

cout << "|------------------------------------------------------------|";

}

void outputSorting(ProductInfo\* arrayOfProduct, int sizeArrayOfData) {

system("cls");

productTableOutput(arrayOfProduct, sizeArrayOfData);

cout << " по какому методу будем соритровать:\n 1) по дню когда он был произведён \n"

<< " 2) сортировка по номеру цеха \n 3) по количеству выпущенных единиц \n 4) выход\n Введите число: ";

int sortingChoice;

cin >> sortingChoice;

if (sortingChoice == 4) {

return;

}

sorting(arrayOfProduct, sizeArrayOfData, sortingChoice);

system("cls");

productTableOutput(arrayOfProduct, sizeArrayOfData);

system("pause");

}

void outputSearch(ProductInfo\* arrayOfProduct, int sizeArrayOfData) {

system("cls");

int searchingChoice, date, element, count, numberOfWorkshop, sizeOfTempArray = 1;

ProductInfo\* searchingArray = new ProductInfo[sizeOfTempArray];

string nameOfResponsible, nameOfProduct;

productTableOutput(arrayOfProduct, sizeArrayOfData);

cout << " по какому методу будем искать:\n 1) по году когда он был произведён \n"

<< " 2) поиск по номеру цеха \n 3) по названию продукта \n 4) по количеству выпущенных единиц"

<< "\n 5) по имени ответсвенного \n 0) Выход \n Введите число: ";

cin >> searchingChoice;

system("cls");

productTableOutput(arrayOfProduct, sizeArrayOfData);

enum SEARCHING\_CHOICE {

EXIT,

YEAR\_OF\_CREATION,

WORKSHOP\_NUMBER,

PRODUCT\_NAME,

AMOUNT\_OF\_PRODUCT,

RESPONSIBLE\_BERSON

};

switch (searchingChoice) {

case YEAR\_OF\_CREATION:

cout << " Введите год: ";

cin >> date;

searchingArray = searchByDayWhenProductCreate(arrayOfProduct, sizeArrayOfData, sizeOfTempArray, date);

break;

case WORKSHOP\_NUMBER:

cout << " Введите номер цеха: ";

cin >> element;

searchingArray = searchByWorkShopNumber(arrayOfProduct, sizeArrayOfData, sizeOfTempArray, element);

break;

case PRODUCT\_NAME:

cout << " Введите название продукта: ";

cin >> nameOfProduct;

searchingArray = searchByProductName(arrayOfProduct, sizeArrayOfData, sizeOfTempArray, nameOfProduct);

break;

case AMOUNT\_OF\_PRODUCT:

cout << " Введите количество выпущенной продукции: ";

cin >> count;

searchingArray = searchByNumberOfProductsProduced(arrayOfProduct, sizeArrayOfData, sizeOfTempArray, count);

break;

case RESPONSIBLE\_BERSON:

cout << " Введите ответсвенного в день выпуска продукции: ";

cin >> nameOfResponsible;

searchingArray = searchByResponsiblePerson(arrayOfProduct, sizeArrayOfData, sizeOfTempArray, nameOfResponsible);

break;

case EXIT:

return;

default:

cout << " такого числа нет попробуй снова";

}

if (sizeOfTempArray >= 1) {

productTableOutput(searchingArray, sizeOfTempArray);

system("pause");

system("cls");

}

else {

cout << " Введите коректную переменную \n";

system("pause");

return;

}

}

void outputIndividualTask(ProductInfo\* arrayOfProduct, int sizeArrayOfData) {

system("cls");

productTableOutput(arrayOfProduct, sizeArrayOfData);

Date beginning, end;

int numberOfWorkshop;

cout << " Введите номер цеха: ";

cin >> numberOfWorkshop;

cout << " Введите период времени \n начало: \n день: ";

cin >> beginning.day;

cout << " месяц: ";

cin >> beginning.month;

cout << " год:";

cin >> beginning.year;

cout << " Введите период времени \n конец: \n день: ";

cin >> end.day;

cout << " месяц: ";

cin >> end.month;

cout << " год:";

cin >> end.year;

system("cls");

int bufSize;

auto buf = OutputOfNumberOfManufacturedProducts(arrayOfProduct, beginning, end, numberOfWorkshop, sizeArrayOfData, bufSize);

if (bufSize == 0) {

cout << " Вы ввели неправильную информацию \n ";

system("pause");

return;

}

productTableOutput(buf, bufSize);

system("pause");

system("cls");

}

// что должно выводить у обычного пошльзователя

void userOutput(ProductInfo\*& arrayOfProduct, int sizeArrayOfData, int choice) {

// sortingChoice и searchingChoice переменные для выбора метода сортировки и поиска соответсвенно

int sizeArrayOfProduct;

string nameOfProduct, nameOfResponsible;

enum CHOICE {

EXIT,

SORTING,

SEARCH,

INDIVIDUAL\_TASK

};

switch (choice) {

case SORTING:

outputSorting(arrayOfProduct, sizeArrayOfData);

break;

case SEARCH:

outputSearch(arrayOfProduct, sizeArrayOfData);

break;

case INDIVIDUAL\_TASK:

outputIndividualTask(arrayOfProduct, sizeArrayOfData);

break;

default:

system("cls");

cout << " Введите коректное число!";

}

}

void outputWorkdWithUsers(Users\*& arrayOfUsers, int& sizeArrayOfUsers) {

int choiceOfUserTable, numberOfUser;

string newName;

while (true) {

system("cls");

userTableOutput(arrayOfUsers, sizeArrayOfUsers);

cout << "\n 1) удаление пользователя\n 2) одобрить регестрацию \n 3) сделать человека админом \n"

<< " 4) изменить имя пользователя\n 5) бан пользователя \n 0) выход \n"

<< " Введите номер: ";

cin >> choiceOfUserTable;

enum CHOICE\_OF\_USER\_TABLE {

EXIT,

DELETE\_USERS,

APPROVE\_USERS,

MAKE\_USER\_ADMIN,

CHANGE\_USERNAME,

BAN

};

switch (choiceOfUserTable) {

case DELETE\_USERS:

cout << "\n Введите номер удаляемого пользователя: ";

cin >> numberOfUser;

deleteUser(arrayOfUsers, sizeArrayOfUsers, numberOfUser);

break;

case APPROVE\_USERS:

cout << " Введите номер одобряемого пользователя: ";

cin >> numberOfUser;

approveUser(arrayOfUsers, sizeArrayOfUsers, numberOfUser);

break;

case MAKE\_USER\_ADMIN:

cout << " Введите номер человека, который станет тоже админом: ";

cin >> numberOfUser;

makeUserAdmin(arrayOfUsers, sizeArrayOfUsers, numberOfUser);

break;

case CHANGE\_USERNAME:

cout << " Введите номер человека, у коготого изменим имя: ";

cin >> numberOfUser;

cout << " Введите новоё имя: ";

cin >> newName;

changeUsername(arrayOfUsers, sizeArrayOfUsers, numberOfUser, newName);

break;

case BAN:

cout << " Введите номер человека, который будет забанен: ";

cin >> numberOfUser;

ban(arrayOfUsers, numberOfUser);

break;

case EXIT:

cout << " Прощайте! ";

system("cls");

return;

default:

system("cls");

cout << " Введите коректное число!";

break;

}

}

}

void outputEditProduct(ProductInfo\* arrayOfProduct, int sizeArrayOfData) {

const string CHANGE = "изменить";

int editChoice, indexOfData;

productTableOutput(arrayOfProduct, sizeArrayOfData);

cout << " Введите номер изменяемого элемента: ";

cin >> indexOfData;

system("cls");

const int ONE\_ELEMENT = 1;

// выводит один нужный нам эелемент.

productTableOutput(arrayOfProduct + indexOfData, ONE\_ELEMENT);

cout << " Измененить продукт по: \n 1) дате производства\n 2) номеру цеха\n 3) названию продукта\n"

<< " 4) количество продукта\n 5) ответвенного за производство продукта \n 0) выход\n";

cout << " Введите число: ";

cin >> editChoice;

if (isVerification(CHANGE)) {

editProduct(arrayOfProduct, indexOfData, editChoice);

}

else {

return;

}

}

void displayAdditionalFeatures(ProductInfo\*& arrayOfProduct, int& sizeArrayOfData, int choiceOfProduct) {

int deletingElement;

const string DELETE = "удалить";

const string ADD = "добавить";

switch (choiceOfProduct) {

case 5:

system("cls");

if (isVerification(ADD)) {

productTableOutput(arrayOfProduct, sizeArrayOfData);

addInArray(arrayOfProduct, sizeArrayOfData);

cout << " Элемент успешно добавлен\n";

system("pause");

}

else {

cout << "Мы ничего не будем добовлять\n";

}

break;

case 6:

system("cls");

productTableOutput(arrayOfProduct, sizeArrayOfData);

cout << " Введите номер удаляемого элемента: ";

cin >> deletingElement;

if (isVerification(DELETE)) {

deleteElement(arrayOfProduct, sizeArrayOfData, deletingElement);

cout << " Элемент успешно удалён\n";

system("pause");

}

else cout << " Мы ничего не будем удалять";

break;

case 7:

system("cls");

outputEditProduct(arrayOfProduct, sizeArrayOfData);

productTableOutput(arrayOfProduct, sizeArrayOfData);

}

}

// вывод админа

void adminOutput(ProductInfo\*& arrayOfProduct, Users\*& arrayOfUsers, int& sizeArrayOfData, int& sizeArrayOfUsers) {

int adminChoice, choiceOfProduct;

while (true) {

system("cls");

cout << " 1) работа с таблицей продукции \n 2) работа с таблицей пользователей \n"

<< "0) Выйти из системы \n Введите номер: ";

enum ADMIN\_CHOICE {

EXIT,

WORK\_WITH\_PRODUCT,

WORK\_WITH\_USERS

};

cin >> adminChoice;

switch (adminChoice) {

case WORK\_WITH\_PRODUCT:

while (true) {

system("cls");

productTableOutput(arrayOfProduct, sizeArrayOfData);

cout << "\n 1) сортировка \n 2) поиск \n 3) вывод количество выпущенных изделий по каждому наименованию \n"

<< "4) выход";

cout << "\n 5) добовление продукта\n 6) удаление продукта \n 7) редактировать данные\n";

cout << " Введите номер: ";

cin >> choiceOfProduct;

if (choiceOfProduct < 4) {

userOutput(arrayOfProduct, sizeArrayOfData, choiceOfProduct);

}

else if(choiceOfProduct == 4) {

cout << " Прощайте\n";

break;

}

else {

displayAdditionalFeatures(arrayOfProduct, sizeArrayOfData, choiceOfProduct);

}

}

break;

case WORK\_WITH\_USERS:

system("cls");

outputWorkdWithUsers(arrayOfUsers, sizeArrayOfUsers);

break;

case EXIT:

system("cls");

cout << "Прощайте\n";

return;

default:

cout << "Введите коректное число";

break;

}

}

}

bool isVerification(string doingSomething) {

bool choice;

cout << " Вы уверены, что хотите " << doingSomething << " этот продукт:\n ";

cout << " 0. Нет\n 1. Да\n Введите число: ";

cin >> choice;

return choice;

}

void createNewAccount(Users\*& arrayOfUsers, int &sizeArrayOfUsers) {

system("cls");

string password, username;

int countOfMistake = 3;

arrayOfUsers = resizeUserArray(arrayOfUsers, sizeArrayOfUsers + 1, sizeArrayOfUsers);

while (countOfMistake != 0) {

cout << " Введите логин: "; cin >> username;

// проверка на правильность ввода имени

if (isUsernameCorrect(username, arrayOfUsers, sizeArrayOfUsers - 1)) {

cout << " этот логин занят, введите новый!!\n";

cout << " У вас осталось ввести " << countOfMistake << " раза\n";

countOfMistake--;

if (countOfMistake == 0) {

cout << " У вас не осталось попыток";

return;

}

}

else {

break;

}

}

cout << " Введите пароль: "; cin >> password;

// добавление в массив

getSalt(arrayOfUsers, password, sizeArrayOfUsers - 1);

password += arrayOfUsers[sizeArrayOfUsers - 1].salt;

arrayOfUsers[sizeArrayOfUsers - 1].saltedHashPassword = hashing(password);

arrayOfUsers[sizeArrayOfUsers - 1].role = 0;

arrayOfUsers[sizeArrayOfUsers - 1].access = 0;

arrayOfUsers[sizeArrayOfUsers - 1].nickname = username;

cout << " Отлично записал, ожидайте одобрение вашего аккаунта \n";

}

// старт работы с консолью

void writeToConsole(ProductInfo\* arrayOfProduct, Users\*& arrayOfUsers, int sizeArrayOfData, int sizeArrayOfUsers) {

setlocale(LC\_ALL, "Russian");

string username;

int choice, choiceOfUsers;

while(true){

system("cls");

cout << " 1) Войти \n 2) Создать новый аккаунт \n 0) Выйти из системы \n";

cout << " Введите ваш выбор: ";

cin >> choice;

enum CHOICE {

EXIT,

LOGIN,

CREATE\_NEW\_ACCOUNT

};

switch (choice) {

case LOGIN:

system("cls");

username = validation(arrayOfUsers, sizeArrayOfUsers);

// часть админа

if (isItAdmin(username, arrayOfUsers, sizeArrayOfUsers)) {

adminOutput(arrayOfProduct, arrayOfUsers, sizeArrayOfData, sizeArrayOfUsers);

}

// часть пользователя

else {

// проверяем допущен ли пользователь к использованию файла с продукцией

if (isAccess(username, arrayOfUsers, sizeArrayOfUsers)) {

system("cls");

productTableOutput(arrayOfProduct, sizeArrayOfData);

cout << "\n 1) сортировка \n 2) поиск \n 3) вывод количество выпущенных изделий по каждому наименованию \n 4) выход\n Введите число: ";

cin >> choiceOfUsers;

if (choiceOfUsers == 4) {

cout << " Прощайте!\n ";

break;

}

userOutput(arrayOfProduct, sizeArrayOfData, choiceOfUsers);

}

// вывод исключительной ситуации

else {

system("cls");

cout << " У вас ещё нет доступа\n";

system("pause");

break;

}

}

break;

// создание нового пользователя

case CREATE\_NEW\_ACCOUNT:

system("cls");

createNewAccount(arrayOfUsers, sizeArrayOfUsers);

system("pause");

break;

// выход из программы

case EXIT:

system("cls");

cout << " Прощайте! \n";

writeInUsersFile(arrayOfUsers, sizeArrayOfUsers);

writeInFile(arrayOfProduct, sizeArrayOfData);

system("pause");

return;

default:

cout << " Введите коректное число!!\n";

system("pause");

break;

}

}

}

**Основные функции из файла workWithData.cpp**

#include "workWithData.h"

ProductInfo\* resizeArray(int& oldSize, int newSize, ProductInfo\*& arrayOfProduct) {

if (oldSize == newSize) {

return arrayOfProduct;

}

ProductInfo\* newArray = new ProductInfo[newSize];

oldSize = newSize < oldSize ? newSize : oldSize;

for (int i = 0; i < oldSize; i++) {

newArray[i] = arrayOfProduct[i];

}

oldSize = newSize;

delete[] arrayOfProduct;

return newArray;

}

// чтение файла с продукцией

void readProductFile(ProductInfo\*& arrayOfProduct, int& size) {

const string NAME\_OF\_PRODUCT\_FILE = "productFile.txt";

ifstream file(NAME\_OF\_PRODUCT\_FILE);

if (!file.is\_open()) {

cout << " У нас проблемы базой данных продукта" << endl;

}

else {

int numberOfLines = 0;

while (!file.eof()) {

if (numberOfLines >= size) {

arrayOfProduct = resizeArray(size, size + 1, arrayOfProduct);

}

file >> arrayOfProduct[numberOfLines].dayWhenProductCreate.day;

file >> arrayOfProduct[numberOfLines].dayWhenProductCreate.month;

file >> arrayOfProduct[numberOfLines].dayWhenProductCreate.year;

file >> arrayOfProduct[numberOfLines].workShopNumber;

file >> arrayOfProduct[numberOfLines].productName;

file >> arrayOfProduct[numberOfLines].responsiblePerson;

file >> arrayOfProduct[numberOfLines].numberOfProductsProduced;

numberOfLines++;

}

--size;

}

file.close();

}

void addInArray(ProductInfo\*& arrayOfProduct, int& size) {

arrayOfProduct = resizeArray(size, size + 1, arrayOfProduct);

cout << " Введите дату когда продукт был изготовлен: \n";

cout << " день: "; cin >> arrayOfProduct[size - 1].dayWhenProductCreate.day;

cout << " месяц: "; cin >> arrayOfProduct[size - 1].dayWhenProductCreate.month;

cout << " год: "; cin >> arrayOfProduct[size - 1].dayWhenProductCreate.year;

cout << " Введите номер цеха: "; cin >> arrayOfProduct[size - 1].workShopNumber;

cout << " Введите название продукта: "; cin >> arrayOfProduct[size - 1].productName;

cout << " Введите количество выпущенных единиц: "; cin >> arrayOfProduct[size - 1].numberOfProductsProduced;

cout << " Введите имя ответвенного во время производва: "; cin >> arrayOfProduct[size - 1].responsiblePerson;

}

bool compByDayWhenProductCreate(ProductInfo& begin, ProductInfo& end) {

return begin.dayWhenProductCreate.day < begin.dayWhenProductCreate.day;

}

bool compByWorkShopNumber(ProductInfo& begin, ProductInfo& end) {

return begin.workShopNumber < end.workShopNumber;

}

bool compByNumberOfProductsProduced(ProductInfo& begin, ProductInfo& end) {

return begin.numberOfProductsProduced < end.numberOfProductsProduced;

}

void sorting(ProductInfo\* arrayOfProduct, int size, int choice) {

/\* 1 - sorting by date (int)

2 - sorting by workShopNumber (int)

3 - sorting by numberOfProductsProduced (int)

4 - sorting by productName (string) x

5 - sorting by responsiblePerson (string) x

\*/

switch (choice) {

// TO DO сделать более нормальную сортировку с учётом месяца и дня

case 1:

sort(arrayOfProduct, arrayOfProduct + size, compByDayWhenProductCreate);

break;

case 2:

sort(arrayOfProduct, arrayOfProduct + size, compByWorkShopNumber);

break;

case 3:

sort(arrayOfProduct, arrayOfProduct + size, compByNumberOfProductsProduced);

break;

}

}

// поиск по году когда он произведён

ProductInfo\* searchByDayWhenProductCreate(ProductInfo\* arrayOfProduct, int size, int& sizeOfTempArray, int date) {

ProductInfo\* tempArray = new ProductInfo[sizeOfTempArray];

for (int numberOfProduct = 0; numberOfProduct < size; numberOfProduct++) {

if (arrayOfProduct[numberOfProduct].dayWhenProductCreate.year == date) {

tempArray = resizeArray(sizeOfTempArray, sizeOfTempArray + 1, tempArray);

tempArray[sizeOfTempArray - 2] = arrayOfProduct[numberOfProduct];

}

}

tempArray = resizeArray(sizeOfTempArray, sizeOfTempArray - 1, tempArray);

return tempArray;

}

// поиск по номеру цеха

ProductInfo\* searchByWorkShopNumber(ProductInfo\* arrayOfProduct, int size, int& sizeOfTempArray, int element) {

ProductInfo\* tempArray = new ProductInfo[sizeOfTempArray];

for (int numberOfProduct = 0; numberOfProduct < size; numberOfProduct++) {

if (arrayOfProduct[numberOfProduct].workShopNumber == element) {

tempArray = resizeArray(sizeOfTempArray, sizeOfTempArray + 1, tempArray);

tempArray[sizeOfTempArray - 2] = arrayOfProduct[numberOfProduct];

}

}

tempArray = resizeArray(sizeOfTempArray, sizeOfTempArray - 1, tempArray);

return tempArray;

}

// поиск по имени продукта

ProductInfo\* searchByProductName(ProductInfo\* arrayOfProduct, int size, int& sizeOfTempArray, string nameOfProduct) {

ProductInfo\* tempArray = new ProductInfo[sizeOfTempArray];

for (int numberOfProduct = 0; numberOfProduct < size; numberOfProduct++) {

if (arrayOfProduct[numberOfProduct].productName == nameOfProduct) {

tempArray = resizeArray(sizeOfTempArray, sizeOfTempArray + 1, tempArray);

tempArray[sizeOfTempArray - 2] = arrayOfProduct[numberOfProduct];

}

}

tempArray = resizeArray(sizeOfTempArray, sizeOfTempArray - 1, tempArray);

return tempArray;

}

// поиск по количеству выпущенных единиц

ProductInfo\* searchByNumberOfProductsProduced(ProductInfo\* arrayOfProduct, int size, int& sizeOfTempArray, int count) {

ProductInfo\* tempArray = new ProductInfo[sizeOfTempArray];

for (int numberOfProduct = 0; numberOfProduct < size; numberOfProduct++) {

if (arrayOfProduct[numberOfProduct].numberOfProductsProduced == count) {

tempArray = resizeArray(sizeOfTempArray, sizeOfTempArray + 1, tempArray);

tempArray[sizeOfTempArray - 2] = arrayOfProduct[numberOfProduct];

}

}

tempArray = resizeArray(sizeOfTempArray, sizeOfTempArray - 1, tempArray);

return tempArray;

}

// поиск по ответсвенному в это дату выпуска продукции

ProductInfo\* searchByResponsiblePerson(ProductInfo\* arrayOfProduct, int size, int& sizeOfTempArray, string person) {

ProductInfo\* tempArray = new ProductInfo[sizeOfTempArray];

for (int numberOfProduct = 0; numberOfProduct < size; numberOfProduct++) {

if (arrayOfProduct[numberOfProduct].responsiblePerson == person) {

tempArray = resizeArray(sizeOfTempArray, sizeOfTempArray + 1, tempArray);

tempArray[sizeOfTempArray - 2] = arrayOfProduct[numberOfProduct];

}

}

tempArray = resizeArray(sizeOfTempArray, sizeOfTempArray - 1, tempArray);

return tempArray;

}

// будет выводиться таблица элементов с нумерацией и пользователь выбирает

// deletingElement который будет удалять

void deleteElement(ProductInfo\*& arrayOfProduct, int& size, int deletingElement) {

int indexOfBuf = 0;

ProductInfo\* buf = new ProductInfo[size - 1];

for (int i = 0; i < size - 1; i++) {

if (i != deletingElement) {

buf[indexOfBuf] = arrayOfProduct[i];

indexOfBuf++;

}

}

arrayOfProduct = buf;

size --;

delete[] buf;

}

// TO DO better !!!!!!!

Date getCurrentDate() {

tm localTime;

time\_t now = time(NULL);

localtime\_s(&localTime, &now);

Date currentDate;

currentDate.day = localTime.tm\_mday;

currentDate.month = 1 + localTime.tm\_mon;

currentDate.year = 1900 + localTime.tm\_year;

return currentDate;

}

int convertToDay(Date day) {

int numberOfDays;

numberOfDays = day.day + day.month \* 30 + day.year \* 365;

return numberOfDays;

}

ProductInfo\* OutputOfNumberOfManufacturedProducts(ProductInfo\* arrayOfProduct, Date beginning, Date end, int numberOfWorkshop, int size, int &sizeOfManufacturedProduct) {

sizeOfManufacturedProduct = 1;

ProductInfo\* manufacturedProducts = new ProductInfo[sizeOfManufacturedProduct];

for (int i = 0; i < size; i++) {

if (arrayOfProduct[i].workShopNumber == numberOfWorkshop) {

if (convertToDay(arrayOfProduct[i].dayWhenProductCreate) >= convertToDay(beginning) &&

convertToDay(arrayOfProduct[i].dayWhenProductCreate) <= convertToDay(end)) {

manufacturedProducts[sizeOfManufacturedProduct - 1] = arrayOfProduct[i];

manufacturedProducts = resizeArray(sizeOfManufacturedProduct, sizeOfManufacturedProduct + 1, manufacturedProducts);

}

}

}

sizeOfManufacturedProduct--;

return manufacturedProducts;

}

void writeInFile(ProductInfo\* arrayOfProduct, int size) {

ofstream file("productFile.txt");

for (int i = 0; i < size; i++) {

file << arrayOfProduct[i].dayWhenProductCreate.day << " ";

file << arrayOfProduct[i].dayWhenProductCreate.month << " ";

file << arrayOfProduct[i].dayWhenProductCreate.year << " ";

file << arrayOfProduct[i].numberOfProductsProduced << " ";

file << arrayOfProduct[i].productName << " ";

file << arrayOfProduct[i].responsiblePerson << " ";

file << arrayOfProduct[i].workShopNumber << endl;

}

file.close();

delete[] arrayOfProduct;

}

void editProduct(ProductInfo\*& arrayOfProduct, int indexOfData, int choice) {

// indexOfData - номер изменяемого продукта

// choice - что именно мы изменяем

/\*

1 - изменение дня производства

2 - изменение номера цеха

3 - изменение названия продукта

4 - изменение количества продукта

5 - изменение ответвенного за производство продукта

0 - выход

\*/

enum CHOICE

{

EXIT,

ENTER\_DATA,

ENTER\_WORKSHOP\_NUMBER,

ENTER\_PRODUCT\_NAME,

ENTER\_NUMBER\_OF\_PRODUCT\_PRODUCED,

ENTER\_RESPONSIBLE\_PERSON,

};

switch (choice) {

case ENTER\_DATA:

cout << " Введите новую дату:\n"

<< " день: ";

cin >> arrayOfProduct[indexOfData].dayWhenProductCreate.day;

cout << " месяц: ";

cin >> arrayOfProduct[indexOfData].dayWhenProductCreate.month;

cout << " год: ";

cin >> arrayOfProduct[indexOfData].dayWhenProductCreate.year;

break;

case ENTER\_WORKSHOP\_NUMBER:

cout << " Введите новый номер цеха: ";

cin >> arrayOfProduct[indexOfData].workShopNumber;

break;

case ENTER\_PRODUCT\_NAME:

cout << " Введите новое название продукта: ";

cin >> arrayOfProduct[indexOfData].productName;

break;

case ENTER\_NUMBER\_OF\_PRODUCT\_PRODUCED:

cout << " Введите новое количество продукта: ";

cin >> arrayOfProduct[indexOfData].numberOfProductsProduced;

break;

case ENTER\_RESPONSIBLE\_PERSON:

cout << " Введите новое имя ответсвенного за этот цех";

cin >> arrayOfProduct[indexOfData].responsiblePerson;

break;

case EXIT:

return;

default:

cout << " Введите корректное число!!\n";

break;

}

}