

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

## ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Факультет «ИиВТ»

(наименование факультета)

Кафедра «ПОВТиАС»

(наименование кафедры)

Зав. кафедрой «ПОВТиАС»

В. В. Долгов

(подпись) (И.О.Ф.)

« » 201\_г.

## ОТЧЕТ

по производственной практике

вид практики

на ООО „ВебАнт“

наименование базы практики

Обучающийся Данила Дмитриевич Бачурин

подпись, дата И.О.Ф.

Обозначение отчета ПП.49.0000.00 Группа ВПР21

Направление 09.03.04 Программная инженерия

код наименование направления подготовки

Профиль Программное обеспечение вычислительной техники и администрирование систем

Руководитель практики:

от предприятия В. И. Даронь

должность подпись, дата имя, отчество, фамилия

от кафедры А. И. Шмыгаль

должность подпись, дата имя, отчество, фамилия

Оценка

дата подпись преподавателя

Ростов-на-Дону

2019 г.



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

## ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Факультет «ИиВТ»

(наименование факультета)

Кафедра «ПОВТиАС»

(наименование кафедры)

## ЗАДАНИЕ

на производственную практику

вид практики

на

наименование базы практики

в период с « 06» июля 2019 г. по « 19» июля 2019 г.

Обучающийся

Данила Дмитриевич Бачурин

И.О.Ф.

Обозначение отчета ПП.49.0000.00

Группа ВПР21 \_

Срок представления отчета на кафедру «19» июля 2019 г. Содержание индивидуального задания

Разработка программного модуля «датапикер» и модуля текстового представления отсчета

времени.

Руководитель практики от кафедры

подпись, дата

А. И. Шмыгаль

И.О.Ф.

Задание принял к исполнению Д. Д. Бачурин

подпись, дата И.О.Ф.



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

## ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Факультет «ИиВТ»

(наименование факультета)

Кафедра «ПОВТиАС»

(наименование кафедры)

Зав. Кафедрой «ПОВТиАС»

В. В. Долгов

(подпись) (И.О.Ф.)

«\_ » 201\_г.

# **Рабочий график (план) проведения практики**

|  |  |  |
| --- | --- | --- |
| **№** | **Мероприятие** | **Срок выполнения** |
| 1 | Инструктаж по технике безопасности на предприятии | 8.07.2019 |
| 2 | Аналитический обзор поставленной задачи | 09.07.2019 – 10.07.2019 |
| 3 | Алгоритмическое конструирование | 11.07.2019 – 12.07.2019 |
| 4 | Программное конструирование | 15.07.2019 – 16.07.2019 |
| 5 | Тестирование приложения | 17. 07.2019 – 19.07.2019 |

Руководитель практики:

от предприятия В. И. Даронь

должность подпись, дата имя, отчество, фамилия

от кафедры А. И. Шмыгаль

должность подпись, дата имя, отчество, фамилия

Ростов-на-Дону

2019 г.

ДНЕВНИК ПРОХОЖДЕНИЯ ПРАКТИКИ

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Место работы | Выполняемые работы | Оценка руководителя |
| 8.07.2019 | ООО “ВебАнт” | Инструктаж по технике безопасности на предприятии |  |
| 09.07.2019 – 10.07.2019 | ООО “ВебАнт” | Аналитический обзор поставленной задачи |  |
| 11.07.2019 – 12.07.2019 | ООО “ВебАнт” | Алгоритмическое конструирование |  |
| 15.07.2019 – 16.07.2019 | ООО “ВебАнт” | Программное конструирование |  |
| 17. 07.2019 – 19.07.2019 | ООО “ВебАнт” | Тестирование приложения |  |

ОТЗЫВ - ХАРАКТЕРИСТИКА

Обучающийся Бачурин Данила Дмитриевич

фамилия, имя, отчество

2 курса группы ВПР21 кафедра «ПОВТиАС»

Вид практики производственная

Наименование места практики

наименование предприятия, структурного подразделения

Обучающийся выполнил задания программы практики

Дополнительно ознакомился/изучил

Заслуживает оценки

Руководитель практики от предприятия

В. И. Даронь

« » 20­\_\_г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc14438553)

[1.1 Обзор предметной области 8](#_Toc14438554)

[1.2 Постановка задачи 11](#_Toc14438555)

[2 Алгоритмическое конструирование 12](#_Toc14438556)

[2.1 Проектирование базы данных 12](#_Toc14438557)

[2.2 Алгоритм работы программы 14](#_Toc14438568)

[2.3 Выводы 16](#_Toc14438569)

[3 Программное конструирование 17](#_Toc14438570)

[3.1 Выбор среды разработки 17](#_Toc14438571)

[3.2 Выбор среды программирования 17](#_Toc14438574)

[3.3 Основные модули программного средства 18](#_Toc14438575)

[3.4 Выводы 19](#_Toc14438576)

[4 Тестирование 20](#_Toc14438577)

[ЗАКЛЮЧЕНИЕ 34](#_Toc14438578)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc14438579)

[ПРИЛОЖЕНИЕ А 36](#_Toc14438580)

ВВЕДЕНИЕ

Хранение информации – одна из важнейших функций компьютера. Одним из распространенных средств такого хранения являются базы данных.

База данных (БД) – совокупность взаимосвязанных, хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений.

Создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляется централизованно с помощью специального программного инструментария – системы управления базами данных.

Система управления базами данных (СУБД) – это комплекс программных и языковых средств, необходимых для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации.

В данном отчёте будет рассмотрен алгоритм построения базы данных для прогнозирования потребности в спец. одежде при планировании будущих закупок.

Целью программы является, на основании данных, введённых пользователем в базу данных, спрогнозировать потребность в количестве товара при будущей закупке.

Задачей программы является последовательный анализ входных данных на корректность, занесение их в базу данных, и, как следствие – вывод отчета для закупки.

В первой главе приведен обзор предметной области поставленной задачи, а также сформулировано ее условие.

Во второй главе представлены алгоритмы, используемые для решения поставленной задачи, нормальные формы и даталогическая модель.

В третьей главе дано обоснование выбора языка программирования, представлены основные модули и функции программы.

В четвертой главе представлена работа созданного программного средства.

**1 Аналитический обзор поставленной задачи**

В данной главе будет рассмотрена актуальность темы работы и основные задачи программного средства. Также будет сформулирована постановка задачи.

1.1 Обзор предметной области

Подсчет товара для закупок необходим для многих предприятий. Поэтому так важно автоматизировать работу, упростить расчеты и сэкономить время работника, производящего подсчеты. Удобнее всего это делать с помощью баз данных.

Создание баз данных включает в себя несколько этапов проектирования:

I этап. Постановка задачи.

На этом этапе формируется задание по созданию БД. В нем подробно описывается состав базы, назначение и цели ее создания.

II этап. Анализ объекта.

На этом этапе рассматривается, из каких объектов может состоять БД, каковы свойства этих объектов. После разбиения БД на отдельные объекты необходимо рассмотреть свойства каждого из этих объектов, или, другими словами, установить, какими параметрами описывается каждый объект. Все эти сведения можно располагать в виде отдельных записей и таблиц. Далее необходимо рассмотреть тип данных каждой отдельной единицы записи.

III этап. Синтез модели.

На этом этапе по проведенному выше анализу необходимо выбрать определенную модель БД. После такого анализа выбирают ту модель, которая сможет максимально обеспечить реализацию поставленной задачи. После выбора модели необходимо нарисовать ее схему с указанием связей между таблицами или узлами.

IV этап. Выбор способов представления информации и программного инструментария.

После создания модели необходимо, в зависимости от выбранного программного продукта, определить форму представления информации.

V этап. Синтез компьютерной модели объекта.

В процессе создания компьютерной модели можно выделить некоторые стадии, типичные для любой СУБД.

Стадия 1. Запуск СУБД, создание нового файла базы данных или открытие созданной ранее базы.

Стадия 2. Создание исходной таблицы или таблиц.

Создавая исходную таблицу, необходимо указать имя и тип каждого поля. Имена полей не должны повторяться внутри одной таблицы. В процессе работы с БД можно дополнять таблицу новыми полями. Созданную таблицу необходимо сохранить, дав ей имя, уникальное в пределах создаваемой базы.

Стадия 3. Создание экранных форм.

Первоначально необходимо указать таблицу, на базе которой будет создаваться форма. Ее можно создавать при помощи мастера форм, указав, какой вид она должна иметь, или самостоятельно – с помощью Конструктора. После создания форму необходимо сохранить. Созданную форму можно редактировать, изменяя местоположение, размеры и формат полей.

Стадия 4. Заполнение БД.

Процесс заполнения БД может проводиться в двух видах: в виде таблицы и в виде формы. Числовые и текстовые поля можно заполнять в виде таблицы, а поля типа МЕМО и OLE – в виде формы.

VI этап. Работа с созданной базой данных.

Работа с БД включает в себя следующие действия:

• поиск необходимых сведений;

• сортировка данных;

• отбор данных;

• изменение и дополнение данных.

Принципы построения систем управления баз данных следуют из требований, которым должна удовлетворять организация баз данных:

• производительность и готовность. Запросы от пользователя базой данных удовлетворяются с такой скоростью, которая требуется для использования данных;

• минимальные затраты. Низкая стоимость хранения и использования данных, минимизация затрат на внесение изменений;

• простота и легкость использования. Пользователи могут легко узнать и понять, какие данные имеются в их распоряжении. Доступ к данным должен быть простым, исключающим возможные ошибки со стороны пользователя;

• простота внесения изменений. База данных может увеличиваться и изменяться без нарушения имеющихся способов использования данных;

• возможность поиска. Пользователь базы данных может обращаться с самыми различными запросами по поводу хранимых в ней данных;

• целостность. Современные базы данных могут содержать данные, используемые многими пользователями. Очень важно, чтобы в процессе работы элементы данных и связи между ними не нарушались. Кроме того, аппаратные ошибки и различного рода случайные сбои не должны приводить к необратимым потерям данных. Значит, система управления данными должна содержать механизм восстановления данных;

• безопасность и секретность. Под безопасностью данных понимают защиту данных от случайного или преднамеренного доступа к ним лиц, не имеющих на это права, от неавторизированной модификации (изменения) данных или их разрушения. Секретность определяется как право отдельных лиц или организаций решать, когда, как какое количество информации может быть передано другим лицам или организациям.

СУБД – это программное обеспечение (ПО), которое позволяет создавать БД, обновлять и дополнять информацию, обеспечивать гибкий доступ к информации. СУБД создает на экране компьютера определенную среду для работы пользователя (интерфейс), и имеет определенные режимы работы и систему команд.

В рамках данной практической работы было создано программное средство с базами данных, удовлетворяющее поставленной задаче в следующем пункте.

1.2 Постановка задачи

Разработать автоматизированную систему для прогнозирования потребности в спец. одежде при планировании будущих закупок. Предусмотреть корректность ввода данных пользователем, вывод соответствующих таблиц и отчета о закупке.

2 Алгоритмическое конструирование

Вся необходимая для работы программы информация предоставляется пользователем. Ему предоставлено меню, с помощью которого он может выбирать путь дальнейших действий: заполнение таблиц, вывод таблиц, удаление из таблиц и подсчет закупки.

В пункте «Заполнить таблицу» пользователь может выбрать таблицу для заполнения либо вернуться в главное меню.

В пункте «Вывести таблицу» пользователь может выбрать таблицу для вывода либо вернуться в главное меню.

В пункте «Удалить из таблицы» пользователь может выбрать таблицу, в которой хочет удалить данные, либо вернуться в главное меню.

В пункте «Подсчет закупки» пользователь вводит дату закупки. Программа подсчитывает необходимое количество спец. одежды и выводит на экран отчет.

Весь алгоритм решения поставленной задачи представлен в виде словесного описания в следующем пункте.

2.1 Проектирование базы данных

Нормальная форма — требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц).

Метод нормальных форм (НФ) состоит в сборе информации о объектах решения задачи в рамках одного отношения и последующей декомпозиции этого отношения на несколько взаимосвязанных отношений на основе процедур нормализации отношений.

Цель нормализации: исключить избыточное дублирование данных, которое является причиной аномалий, возникших при добавлении, редактировании и удалении кортежей (строк таблицы).

Процесс проектирования БД с использованием метода НФ является итерационным и заключается в последовательном переводе отношения из 1НФ в НФ более высокого порядка по определенным правилам. Каждая следующая НФ ограничивается определенным типом функциональных зависимостей и устранением соответствующих аномалий при выполнении операций над отношениями БД, а также сохранении свойств предшествующих НФ.

Первая нормальная форма: отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице. В рассматриваемой схеме данных данное требование предусмотрено.

Вторая нормальная форма: отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа(ПК). Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость. В рассматриваемой схеме данных данное требование предусмотрено.

Третья нормальная форма: отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы. В рассматриваемой схеме данных данное требование предусмотрено.

На рисунке 2.1 предоставлена даталогическая модель базы данных.

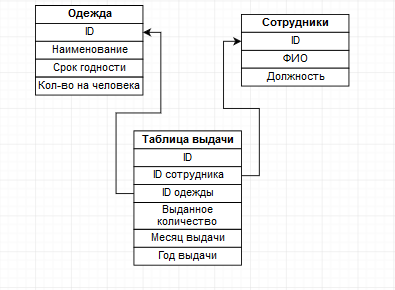


Рисунок 2.1 Даталогическая модель базы данных

2.2 Алгоритм работы программы

Шаг 1. Создание базы данных. Указание названия таблиц и столбцов.

Шаг 2. Указание названия программы, размера окна.

Шаг 3. Создание главного меню с четырьмя пунктами: «Заполнить таблицу», «Вывести таблицу», «Удалить из таблицы» и «Подсчет закупки».

Шаг 4. В пункте «Заполнить таблицу» создать 4 подпункта: «Спец. одежда», «Сотрудники», «Таблица выдачи», «Удалить из таблицы» и «Главное меню».

В подпункте «Спец. одежда» предложить пользователю ввод данных в графы «Наименование одежды», «Срок использования в месяцах» и «Положенное количество спец. одежды на сотрудника в год». Реализовать проверку введенных данных. Если данные введены корректно – внести их в базу данных и уведомить об этом пользователя, иначе вывести сообщение «Данные введены некорректно».

В подпункте «Сотрудники» предложить пользователю ввод данных в графы «ФИО» и «Должность». Реализовать проверку введенных данных. Если данные введены корректно – внести их в базу данных и уведомить об этом пользователя, иначе вывести сообщение «Данные введены некорректно».

В подпункте «Таблица выдачи» предложить пользователю ввод данных в графы «ФИО сотрудника, которому выдана спец. одежда», «Наименование выданной спец. одежды», «Количество выданной спец. одежды» и «Дата выдачи». Реализовать проверку введенных данных. Если данные введены корректно – внести их в базу данных и уведомить об этом пользователя, иначе вывести сообщение «Данные введены некорректно».

Шаг 5. В пункте «Вывести таблицу» создать четыре подпункта: «Спец. одежда», «Сотрудники», «Таблица выдачи» и «Главное меню».

В подпункте «Спец. одежда» вывести из базы данных информацию о введенных на шаге 4 данных в таблицу «Спец. одежда».

В подпункте «Сотрудники» вывести из базы данных информацию о введенных на шаге 4 данных в таблицу «Сотрудники».

В подпункте «Таблица выдачи» вывести из базы данных информацию о введенных на шаге 4 данных в таблицу «Таблица выдачи».

Шаг 6. В пункте «Удалить из таблицы» создать три подпункта «Спец. одежда», «Сотрудники» и «Главное меню».

В подпункте «Спец. одежда» предложить пользователю ввод данных в графу «Наименование одежды, которую нужно удалить из базы данных». Реализовать проверку введенных данных. Если данные введены корректно – удалить их из базы данных и уведомить об этом пользователя, иначе вывести сообщение «Данные введены некорректно».

В подпункте «Сотрудники» предложить пользователю ввод данных в графу «ФИО сотрудника, которого нужно удалить из базы данных». Реализовать проверку введенных данных. Если данные введены корректно – удалить их из базы данных и уведомить об этом пользователя, иначе вывести сообщение «Данные введены некорректно».

Шаг 7. В пункте «Подсчет закупки» предложить пользователю ввод данных в графы «Месяц закупки» и «Год закупки». Реализовать проверку введенных данных. Если данные введены корректно – вывести отчет о количестве необходимого товара, иначе вывести сообщение «Данные введены некорректно».

2.3 Выводы

Разработан алгоритм, который позволяет решить поставленную задачу наиболее оптимальным образом.

3 Программное конструирование

На основе выбранных алгоритмов решения поставленной задачи, выполнено программное конструирование, предполагающее выбор соответствующих требованиям сред разработки и программирования.

3.1 Выбор среды разработки

Для реализации разработанного алгоритма был выбран язык программирования Python.

Python — высокоуровневый язык программирования с динамической типизацией, поддерживающий объектно-ориентированный, функциональный и императивный стили программирования. Это язык общего назначения, на котором можно одинаково успешно разрабатывать системные приложения с графическим интерфейсом, базы данных, утилиты командной строки, научные приложения, игры, приложения для веб и много другое.

Python отличается удобным синтаксисом, что позволяет довольно быстро и доступно его изучать.

3.2 Выбор среды программирования

Jet Brains PyCharm – профессиональная среда разработки, которая предоставляет первоклассную поддержку для многих языков программирования, таких как Python, JavaScript, CoffeeScript, TypeScript, CSS и других.

PyCharm имеет разумное обновление кода с безопасным удалением и переименованием, методом извлечения, вводной переменной, встроенной переменной или методом, и другими видами рефакторинга. Большая коллекция инструментов PyCharm включает в себя интегрированный отладчик и запуск тестирования, а так же:

* профайлер Python;
* встроенный терминал;
* интеграцию с большими VCS и встроенными инструментами [баз данных](https://python-scripts.com/database);
* возможность удаленной разработки с удаленными интерпретаторами;
* интегрированный терминал ssh;
* интеграция с Docker и Vagrant.

PyCharm прекрасно поддерживает работу с базами данных, на него можно рассчитывать при редактировании кода SQL, выполнении запросов, просмотре данных и внесении изменений в схемы.

3.3 Основные модули программного средства

Программа состоит из двух файлов. Файл «CREATE\_DB.py» является файлом разработчика для создания пустой базы данных. Он подключен к библиотеке sqlite3 и содержит функции, при вызове которых создается новая база данных.

Файл «DataBase.py» содержит исходный код программы. Он подключен к библиотеке sqlite3, для работы с базами данных, и библиотеке tkinter, для работы с интерфейсом. Файл содержит несколько основных модулей: запись в таблицы, вывод таблиц, удаление из таблиц и подсчет закупки. Запись в таблицы содержит функции для записи в таблицы «Спец. одежда» (функция «cw» для ввода данных пользователем в базу данных «Спец. одежда» и функция «done\_cw» для записи введенных данных в базу данных «Спец.одежда»), «Сотрудники» (функция «ww» для ввода данных пользователем в базу данных «Сотрудники» и функция «done\_ww» для записи введенных данных в базу данных «Сотрудники») и «Таблица выдачи» (функция «dw» для ввода данных пользователем в базу данных «Таблица выдачи» и функция «done\_dw» для записи введенных данных в базу данных «Таблица выдачи»). Вывод таблиц содержит функции для вывода на экран таблиц «Спец. одежда» («pc» – функция вывода таблицы «Спец. одежда»), «Сотрудники» («pw» – функция вывода таблицы «Сотрудники») и «Таблица выдачи» («pd» – функция вывода таблицы «Таблица выдачи»). Удаление из таблиц содержит функции для удаления данных из таблиц «Спец. одежда» (функция «del\_c» для ввода пользователем данных для удаления из таблицы «Спец. и одежда» и функция «del\_cl», производящая удаление из таблицы «Спец. одежда») и «Сотрудники» (функция «del\_w» для ввода пользователем данных для удаления из таблицы «Сотрудники» и функция «del\_wr», производящая удаление из таблицы «Сотрудники»). Подсчет закупки содержит функции для подсчета необходимой спец. одежды (функция «tz» для ввода пользователем даты закупки и функция «tz\_done» для подсчета закупки).

Исходный код файла «CREATE\_DB.py» находится в листинге А.1 (см. приложение А).

Исходный код файла «DataBase.py» находится в листинге А.2 (см. приложение А).

3.4 Выводы

Реализована программа, позволяющая решить поставленную задачу на основании введенных пользователем данных.

4 Тестирование

Главное меню представлено на рисунке 4.1.

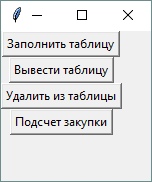


Рисунок 4.1 – Главное меню

Нажатие пользователем кнопки «Заполнить таблицу» продемонстрированно на рисунке 4.2.

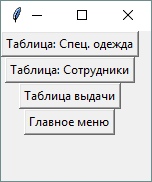


Рисунок 4.2 – Меню выбора таблиц для заполнения

Нажатие пользователем кнопки «Спец. одежда» продемонстрированно на рисунке 4.3.

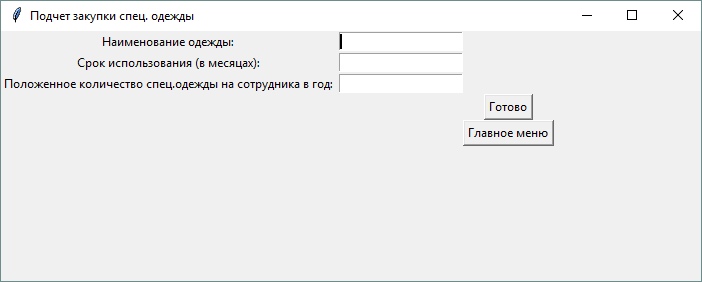


Рисунок 4.3 – Форма для заполнения таблицы «Спец. одежда»

В случае, когда пользователь вводит некорректные данные и нажимает на кнопку «Готово», программа уведомляет об ошибке, что продемонстрированно на рисунке 4.4.

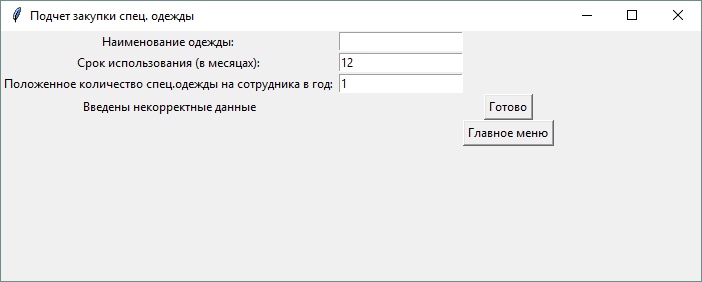


Рисунок 4.4 – Некорректный ввод данных в таблицу «Спец. одежда»

В случае, когда пользователь вводит корректные данные и нажимает на кнопку «Готово», программа уведомляет об успешной записи в базу данных, что продемонстрированно на рисунке 4.5.

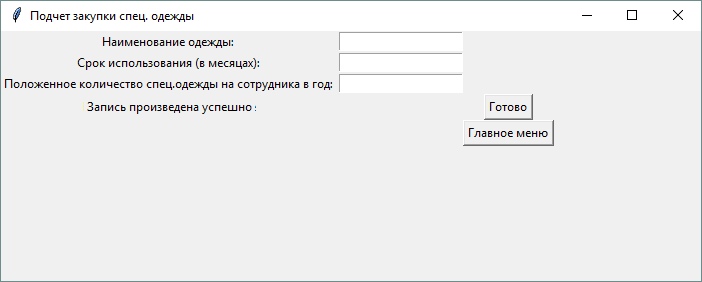


Рисунок 4.5 – Успешная запись данных в таблицу «Спец. одежда»

Нажатие пользователем кнопки «Сотрудники», находящейся в подпункте «Заполнить таблицу» (рисунок 4.2), продемонстрированно на рисунке 4.6.

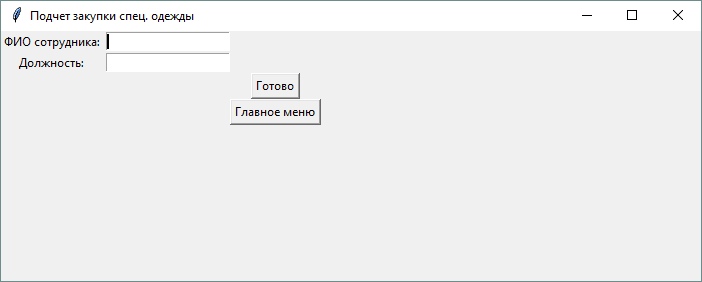


Рисунок 4.6 – Форма для заполнения таблицы «Сотрудники»

В случае, когда пользователь вводит некорректные данные и нажимает на кнопку «Готово», программа уведомляет об ошибке, что продемонстрированно на рисунке 4.7.

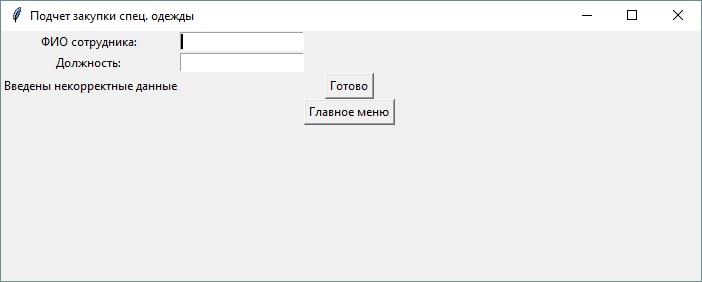


Рисунок 4.7 – Некорректный ввод данных в таблицу «Сотрудники»

В случае, когда пользователь вводит корректные данные и нажимает на кнопку «Готово», программа уведомляет об успешной записи в базу данных, что продемонстрированно на рисунке 4.8.

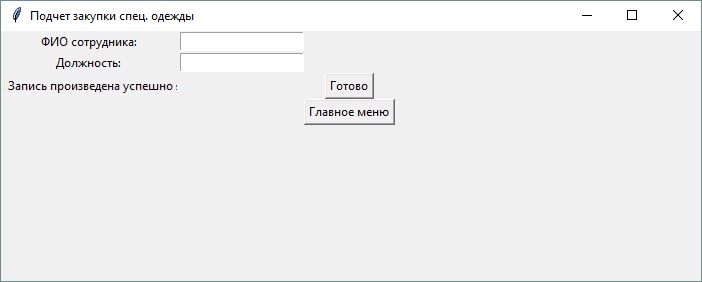


Рисунок 4.8 – Успешная запись данных в таблицу «Сотрудники»

Нажатие пользователем кнопки «Таблица выдачи», находящейся в подпункте «Заполнить таблицу» (рисунок 4.2), продемонстрированно на рисунке 4.9.

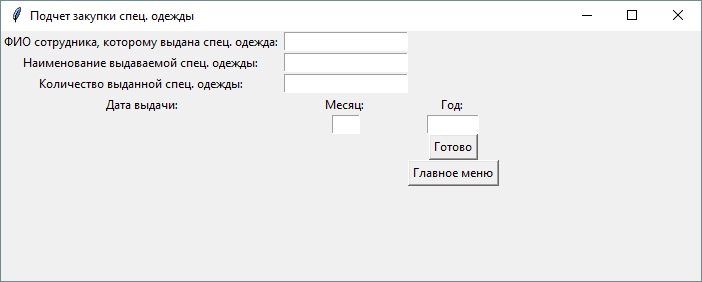


Рисунок 4.9 – Форма для заполнения таблицы «Таблица выдачи»

В случае, когда пользователь вводит некорректные данные и нажимает на кнопку «Готово», программа уведомляет об ошибке, что продемонстрированно на рисунке 4.10.

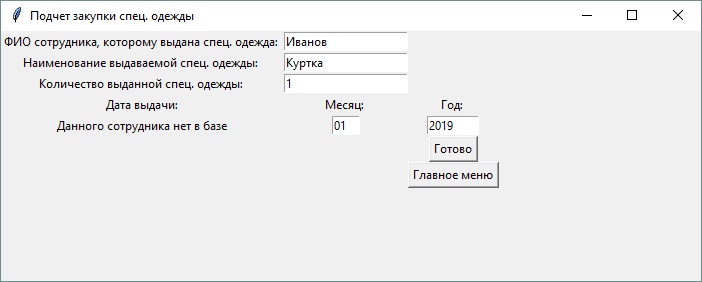


Рисунок 4.10 – Некорректный ввод данных в таблицу «Таблица выдачи»

В случае, когда пользователь вводит корректные данные и нажимает на кнопку «Готово», программа уведомляет об успешной записи в базу данных, что продемонстрированно на рисунке 4.11.

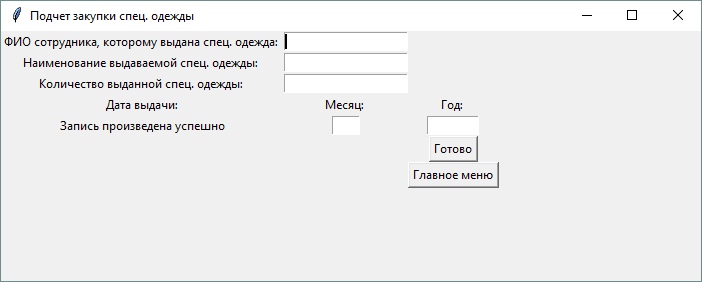


Рисунок 4.11 – Успешная запись данных в таблицу «Таблица выдачи»

Нажатие пользователем кнопки «Вывести таблицу», находящейся в главном меню (рисунок 4.1), продемонстрированно на рисунке 4.12.

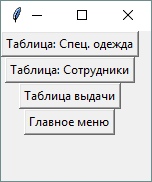


Рисунок 4.12 – Меню выбора таблиц для вывода

Нажатие пользователем кнопки «Спец. одежда» продемонстрированно на рисунке 4.13.

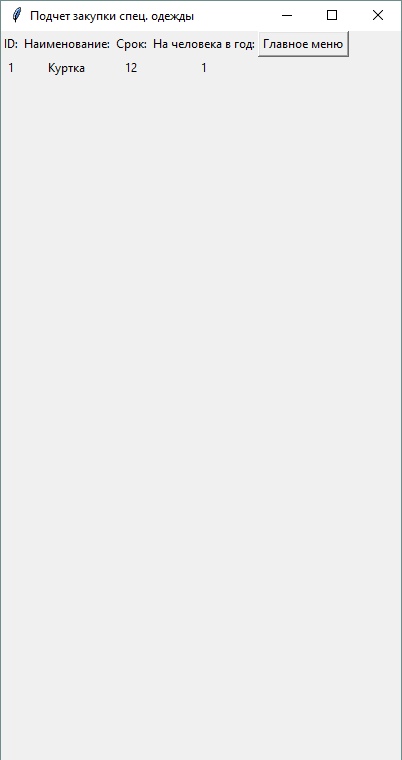


Рисунок 4.13 – Вывод таблицы «Спец. одежда»

Нажатие пользователем кнопки «Сотрудники», находящейся в подпункте «Вывести таблицу» (рисунок 4.12), продемонстрированно на рисунке 4.14.

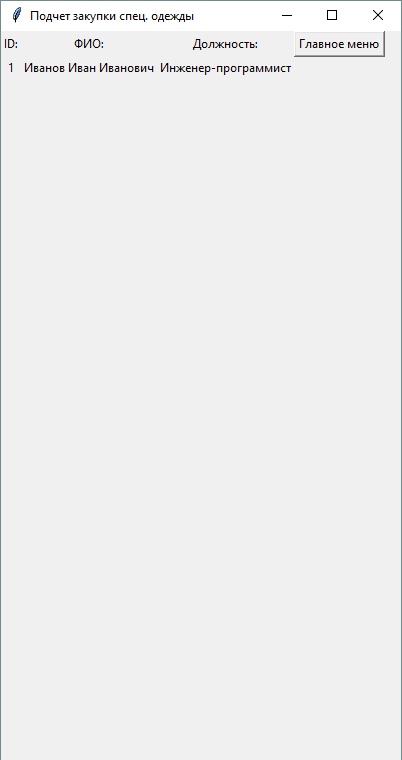


Рисунок 4.14 – Вывод таблицы «Сотрудники»

Нажатие пользователем кнопки «Таблица выдачи», находящейся в подпункте «Вывести таблицу» (рисунок 4.12), продемонстрированно на рисунке 4.15.

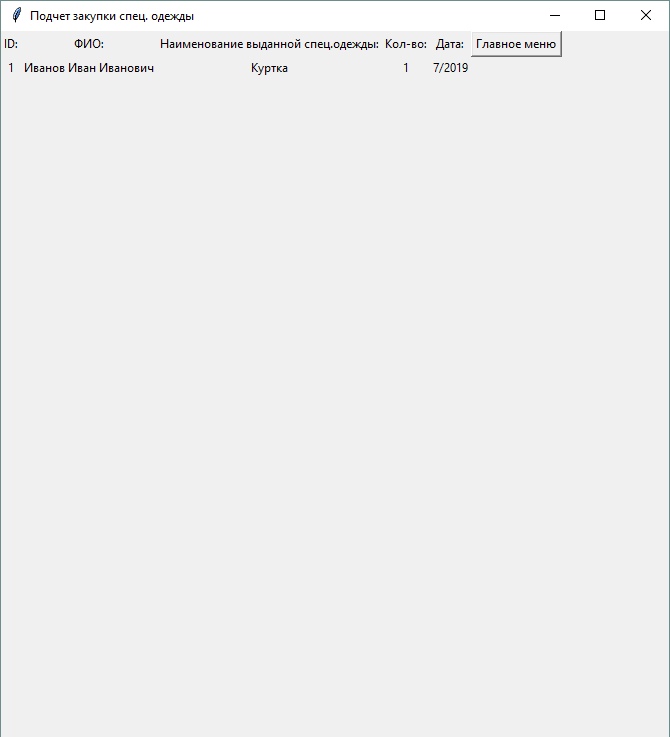


Рисунок 4.15 – Вывод таблицы «Таблица выдачи»

Нажатие пользователем кнопки «Удалить из таблицы», находящейся в главном меню (рисунок 4.1), продемонстрированно на рисунке 4.16.

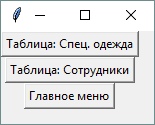


Рисунок 4.16 – Меню выбора таблиц для удаления

Нажатие пользователем кнопки «Спец. одежда» продемонстрированно на рисунке 4.17.

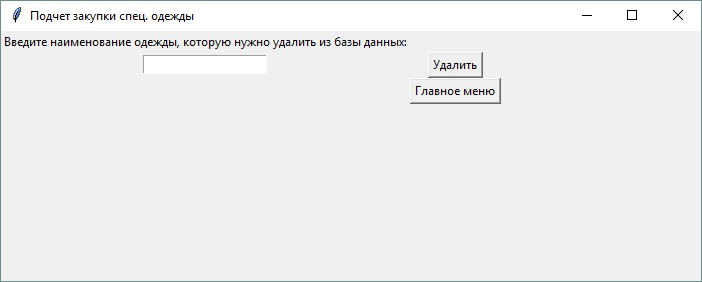


Рисунок 4.17 – Форма для удаления из таблицы «Спец. одежда»

В случае, когда пользователь вводит некорректные данные и нажимает на кнопку «Удалить», программа уведомляет об ошибке, что продемонстрированно на рисунке 4.18.

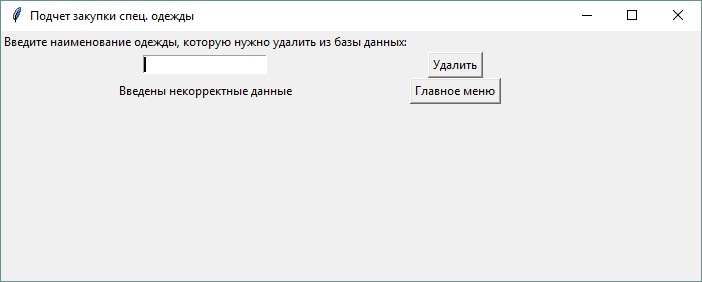


Рисунок 4.18 – Некорректный ввод данных в форму для удаления из таблицы «Спец. одежда»

В случае, когда пользователь вводит корректные данные и нажимает на кнопку «Удалить», программа уведомляет об успешном удалении из базы данных, что продемонстрированно на рисунке 4.19.

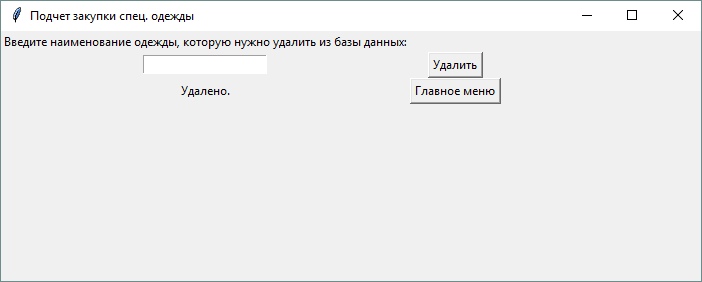


Рисунок 4.19 –Успешное удаление из таблицы «Спец. одежда»

Нажатие пользователем кнопки «Сотрудники», находящейся в подпункте «Удалить из таблицы» (рисунок 4.16), продемонстрированно на рисунке 4.20.

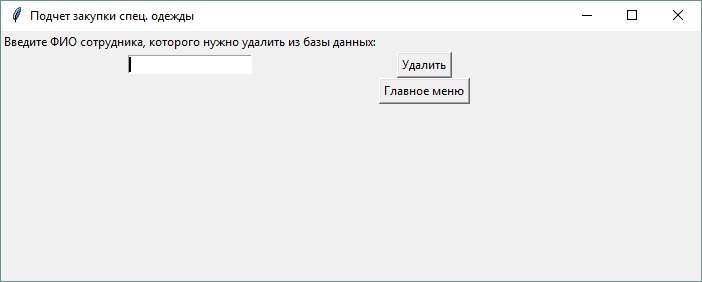


Рисунок 4.20 – Форма для удаления из таблицы «Сотрудники»

В случае, когда пользователь вводит некорректные данные и нажимает на кнопку «Удалить», программа уведомляет об ошибке, что продемонстрированно на рисунке 4.21.

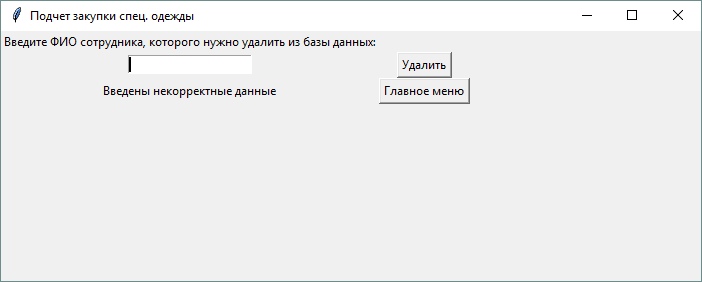


Рисунок 4.21 – Некорректный ввод данных в форму для удаления из таблицы «Сотрудники»

В случае, когда пользователь вводит корректные данные и нажимает на кнопку «Удалить», программа уведомляет об успешном удалении из базы данных, что продемонстрированно на рисунке 4.22.

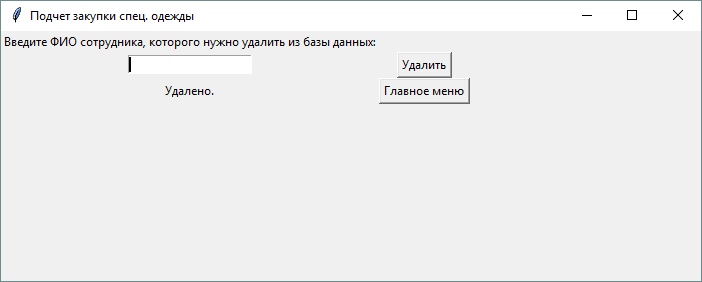


Рисунок 4.22 –Успешное удаление из таблицы «Сотрудники»

Нажатие пользователем кнопки «Подсчет закупки», находящейся в главном меню (рисунок 4.1), продемонстрированно на рисунке 4.23.

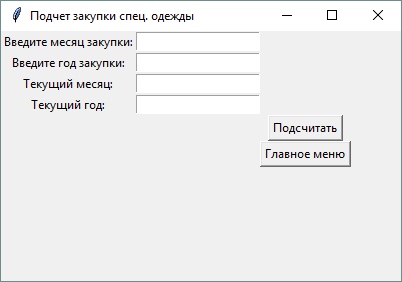


Рисунок 4.23 – Форма заполнения для подсчета закупки

В случае, когда пользователь вводит некорректные данные и нажимает на кнопку «Подсчитать», программа уведомляет об ошибке, что продемонстрированно на рисунке 4.24.

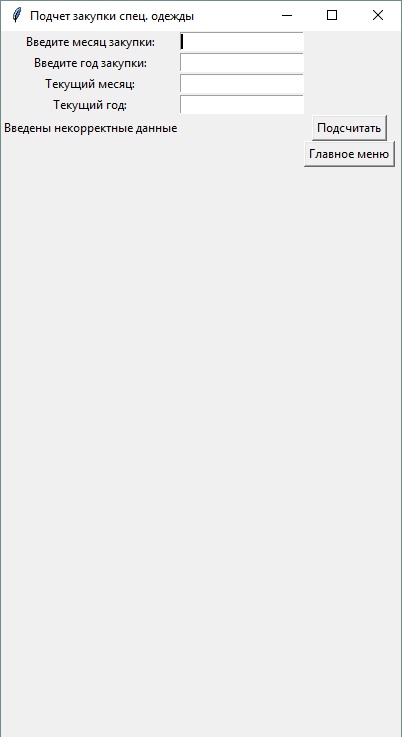


Рисунок 4.24 – Некорректный ввод при подсчете закупки

В случае, когда пользователь вводит корректные данные и нажимает на кнопку «Подсчитать», программа выводит подсчет закупки, что продемонстрированно на рисунке 4.25.

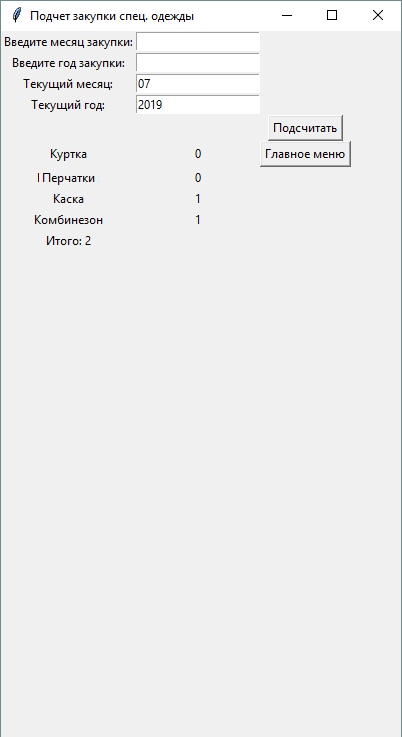


Рисунок 4.25 – Подсчет закупки

Результаты проведенного тестирования показали, что разработанное программное средство в полной мере выполняет поставленную задачу, а также корректно реагирует на любые действия пользователя, обрабатывая все возможные исключения.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была создана программа, решающая задачу подсчета закупки спец. одежды.

В ходе выполнения работы был построен аналитически обоснованный алгоритм решения поставленной задачи и разработано соответствующее программное средство.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алексеев Е.Г., Богатырев С.Д. «Информатика. Мультимедийный электронный учебник»:

URL: <http://inf.e-alekseev.ru/text/Bd.html> (дата обращения 9.07.2019 г.)

1. IT-блог о веб-технологиях, серверах, протоколах, базах данных, СУБД, SQL, компьютерных сетях, языках программирования и создание сайтов.

URL: <https://zametkinapolyah.ru/zametki-o-mysql> (дата обращения 15.07.2019 г.)

1. PyCharm

URL: <https://jetbrains.ru/products/pycharm/> (дата обращения 15.07.2019 г.)

1. Python 3 для начинающих

URL: <https://pythonworld.ru/> (дата обращения 15.07.2019 г.)

1. METANIT.COM

URL: <https://metanit.com/python/tutorial/9.1.php> (дата обращения 16.07.2019 г.)

1. habr.com

URL: <https://habr.com/ru/post/254773/> (дата обращения 15.07.2019 г.)

ПРИЛОЖЕНИЕ А

**Исходный код программного средства**

Листинг А.1 - Исходный код файла «CREATE\_DB.py»

import sqlite3  
conn = sqlite3.connect('PR.db')  
c = conn.cursor()  
  
c.execute(  
 'CREATE TABLE clothes (id INTEGER PRIMARY KEY AUTOINCREMENT not null,'  
 'clothname TEXT NOT NULL, '  
 'selflife INTEGER NOT NULL,'  
 'norma INTEGER NOT NULL)'  
)  
conn.commit()  
  
c.execute(  
 'CREATE TABLE worker (id INTEGER PRIMARY KEY AUTOINCREMENT not null,'  
 'surname TEXT NOT NULL,'  
 'post TEXT NOT NULL)'  
)  
conn.commit()  
  
c.execute(  
 'CREATE TABLE data (Id INTEGER PRIMARY KEY AUTOINCREMENT not null, '  
 'id\_worker INTEGER NOT NULL,'  
 'id\_clothes integer not null,'  
 'num INTEGER NOT NULL,'  
 'month INTEGER NOT NULL,'  
 'year integer not null,'  
 'foreign KEY (id\_clothes) REFERENCES clothes(id),'  
 'foreign key (id\_worker) references worker(id))'  
)  
conn.commit()  
  
c.close()  
conn.close()

Листинг А.2 - Исходный код файла «DataBase.py»

from tkinter import \*  
import sqlite3  
conn = sqlite3.connect('DataBase.db')  
c = conn.cursor()  
  
#очистка экрана и переход в главное меню  
def clear():  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 gl()  
#Запись введенных данных в бд "Спец.одежда"  
def done\_cw(txt, txt1, txt2):  
 clothname = txt.get()  
 selflife = txt1.get()  
 norma = txt2.get()  
 if (clothname == "") or (selflife == "") or (norma == ""):  
 error = Label(window, text='Введены некорректные данные')  
 error.grid(column=1, row=5)  
 return  
 elif (not selflife.isdigit()) or (not norma.isdigit()):  
 error = Label(window, text='Введены некорректные данные')  
 error.grid(column=1, row=5)  
 return  
 elif (int(selflife) <= 0) or (int(norma) <= 0):  
 error = Label(window, text='Введены некорректные данные')  
 error.grid(column=1, row=5)  
 return  
 c.execute(  
 '''INSERT INTO clothes(clothname, selflife, norma) VALUES (?,?,?)''', (clothname, selflife, norma)  
 )  
 conn.commit()  
 txt.delete(0,'end')  
 txt1.delete(0,'end')  
 txt2.delete(0,'end')  
 txt.focus()  
 res = Label(window, text='Запись произведена успешно')  
 res.grid(column=1, row=5)

#Ввод данных пользователем в бд "Спец. одежда"  
def cw():  
 window.geometry('700x250')  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 lbl = Label(window, text = 'Наименование одежды: ')  
 lbl.grid(column = 1, row = 2)  
 txt = Entry(window, width = 20)  
 txt.grid(column = 2, row = 2)  
 txt.focus()  
 lbl1 = Label(window, text='Срок использования (в месяцах): ')  
 lbl1.grid(column=1, row=3)  
 txt1 = Entry(window, width=20)  
 txt1.grid(column=2, row=3)  
 lbl2 = Label(window, text='Положенное количество спец.одежды на сотрудника в год: ')  
 lbl2.grid(column=1, row=4)  
 txt2 = Entry(window, width=20)  
 txt2.grid(column=2, row=4)  
 btn5 = Button(window, text = 'Готово',

command = lambda: done\_cw(txt, txt1, txt2))  
 btn5.grid(column = 3, row = 5)  
 btn6 = Button(window, text = 'Главное меню', command = clear)  
 btn6.grid(column = 3, row = 6)

#Запись введенных данных в бд "Сотрудники"  
def done\_ww(txt,txt1):  
 surname = txt.get()  
 post = txt1.get()  
 if (surname == "") or (post == ""):  
 error = Label(window, text='Введены некорректные данные')  
 error.grid(column=1, row=5)  
 return  
 c.execute(  
 '''INSERT INTO worker(surname, post) VALUES (?,?)''', (surname, post))  
 conn.commit()  
 txt.delete(0, 'end')  
 txt1.delete(0, 'end')  
 txt.focus()  
 res = Label(window, text='Запись произведена успешно')  
 res.grid(column=1, row=5)

#Ввод данных пользователем в бд "Сотрудники"  
def ww():  
 window.geometry('700x250')  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 lbl = Label(window, text = 'ФИО сотрудника: ')  
 lbl.grid(column = 1, row = 1)  
 txt = Entry(window, width = 20)  
 txt.grid(column = 2, row = 1)  
 txt.focus()  
 lbl1 = Label(window, text = 'Должность: ')  
 lbl1.grid(column = 1, row = 2)  
 txt1 = Entry(window, width = 20)  
 txt1.grid(column = 2, row = 2)  
 btn5 = Button(window, text='Готово', command=lambda: done\_ww(txt, txt1))  
 btn5.grid(column=3, row=5)  
 btn6 = Button(window, text='Главное меню', command=clear)  
 btn6.grid(column=3, row=6)

#Запись введенных данных в бд "Таблица выдачи"  
def done\_dw(txt, txt1,txt2,txt4,txt5):  
 id\_worker = 0  
 id\_clothes = 0  
 w = txt.get()  
 cl = txt1.get()  
 n = txt2.get()  
 m = txt4.get()  
 y = txt5.get()  
 if (w == "") or (cl == "") or (n == "") or (m == "") or (y == ""):  
 error = Label(window, text='Введены некорректные данные')  
 error.grid(column=1, row=5)  
 return  
 elif (not n.isdigit()) or (not m.isdigit()) or (not y.isdigit()):  
 error = Label(window, text='Введены некорректные данные')  
 error.grid(column=1, row=5)  
 return  
 elif (int(n) <= 0) or (int(m) <= 0 and int(m)>12) or (int(y) <= 2000):  
 error = Label(window, text='Введены некорректные данные')  
 error.grid(column=1, row=5)  
 return  
 c.execute("SELECT \* FROM worker")  
 while True:  
 row = c.fetchone()  
 if row == None:  
 break  
 if (row[1] == w ): id\_worker = int(row[0])  
 if id\_worker == 0:  
 err = Label(window, text ='Данного сотрудника нет в базе')  
 err.grid(column=1,row=5)  
 return  
 c.execute("SELECT \* FROM clothes")  
 while True:  
 row = c.fetchone()  
 if row == None:  
 break  
 if (row[1] == cl): id\_clothes = int(row[0])  
 if id\_clothes == 0:  
 err = Label(window, text='Данной спец. одежды нет в базе' err.grid(column=1, row=5)  
 return  
 c.execute(  
 '''INSERT INTO data(id\_worker, id\_clothes, num, month, year) VALUES (?,?,?,?,?)''',  
 (id\_worker, id\_clothes, n, m, y)  
 )  
 conn.commit()  
 txt.delete(0, 'end')  
 txt1.delete(0, 'end')  
 txt2.delete(0, 'end')  
 txt4.delete(0, 'end')  
 txt5.delete(0, 'end')  
 txt.focus()  
 res = Label(window, text='Запись произведена успешно')  
 res.grid(column=1, row=5)

#Ввод данных пользователем в бд "Таблица выдачи"  
def dw():  
 window.geometry('700x250')  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 lbl = Label(window, text='ФИО сотрудника, которому выдана спец. одежда: ')  
 lbl.grid(column=1, row=1)  
 txt = Entry(window, width=20)  
 txt.grid(column=2, row=1)  
 txt.focus()  
 lbl1 = Label(window, text='Наименование выдаваемой спец. одежды: ')  
 lbl1.grid(column=1, row=2)  
 txt1 = Entry(window, width=20)  
 txt1.grid(column=2, row=2)  
 lbl2 = Label(window, text='Количество выданной спец. одежды: ')  
 lbl2.grid(column=1, row=3)  
 txt2 = Entry(window, width=20)  
 txt2.grid(column=2, row=3)  
 lbl3 = Label(window, text='Дата выдачи:')  
 lbl3.grid(column=1, row=4)  
 lbl4 = Label(window, text='Месяц: ')  
 lbl4.grid(column=2, row=4)  
 txt4 = Entry(window, width=4)  
 txt4.grid(column=2, row=5)  
 lbl5 = Label(window, text='Год: ')  
 lbl5.grid(column=3, row=4)  
 txt5 = Entry(window, width=8)  
 txt5.grid(column=3, row=5)  
 btn5 = Button(window, text='Готово', command=lambda: done\_dw(txt, txt1,txt2,txt4,txt5))  
 btn5.grid(column=3, row=6)  
 btn6 = Button(window, text='Главное меню', command= clear)  
 btn6.grid(column=3, row=7)

#Меню выбора таблицы для записи  
def wr():  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 btn4 = Button(window, text = 'Таблица: Спец. одежда', command = cw)  
 btn4.grid(column=1, row=1)  
 btn5 = Button(window, text = 'Таблица: Сотрудники', command = ww)  
 btn5.grid(column = 1, row = 2)  
 btn6 = Button(window, text = 'Таблица выдачи', command = dw)  
 btn6.grid(column=1,row=3)  
 btn7 = Button(window, text='Главное меню', command = clear)  
 btn7.grid(column=1, row=4)

#Вывод таблицы "Спец. одежда"  
def pc():  
 window.geometry('400x950')  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 lbl = Label(window,text='ID:')  
 lbl.grid(column = 1, row = 1)  
 lbl2 = Label(window,text='Наименование:')  
 lbl2.grid(column=2, row=1)  
 lbl3 = Label(window,text='Срок:')  
 lbl3.grid(column=3, row=1)  
 lbl4 = Label(window,text = 'На человека в год:')  
 lbl4.grid(column=4, row=1)  
 a = 1  
 b = 2  
 c.execute("SELECT \* FROM clothes")  
 while True:  
 row = c.fetchone()  
 if row == None:  
 break  
 lbl5 = Label(window, text = str(row[0]))  
 lbl5.grid(column = a, row = b)  
 lbl6 = Label(window, text=str(row[1]))  
 lbl6.grid(column=a+1, row=b)  
 lbl7 = Label(window, text=str(row[2]))  
 lbl7.grid(column=a+2, row=b)  
 lbl8 = Label(window, text=str(row[3]))  
 lbl8.grid(column=a+3, row=b)  
 b+=1  
 btn7 = Button(window, text='Главное меню', command=clear)  
 btn7.grid(column=a+4, row=1)

#Вывод таблицы "Сотрудники"  
def pw():  
 window.geometry('400x950')  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 lbl = Label(window, text='ID:')  
 lbl.grid(column=1, row=1)  
 lbl2 = Label(window, text='ФИО:')  
 lbl2.grid(column=2, row=1)  
 lbl3 = Label(window, text='Должность:')  
 lbl3.grid(column=3, row=1)  
 a = 1  
 b = 2  
 c.execute("SELECT \* FROM worker")  
 while True:  
 row = c.fetchone()  
 if row == None:  
 break  
 lbl5 = Label(window, text=str(row[0]))  
 lbl5.grid(column=a, row=b)  
 lbl6 = Label(window, text=str(row[1]))  
 lbl6.grid(column=a + 1, row=b)  
 lbl7 = Label(window, text=str(row[2]))  
 lbl7.grid(column=a + 2, row=b)  
 b += 1  
 btn7 = Button(window, text='Главное меню', command=clear)  
 btn7.grid(column=a + 4, row=1)

#Вывод таблицы "Таблица выдачи"  
def pd():  
 window.geometry('400x950')  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 lbl = Label(window, text='ID:')  
 lbl.grid(column=1, row=1)  
 lbl2 = Label(window, text='ФИО:')  
 lbl2.grid(column=2, row=1)  
 lbl3 = Label(window, text='Наименование выданной спец.одежды:')  
 lbl3.grid(column=3, row=1)  
 lbl4 = Label(window, text='Кол-во:')  
 lbl4.grid(column=4, row=1)  
 lbl5 = Label(window, text='Дата:')  
 lbl5.grid(column=5, row=1)  
 a = 1  
 b = 2  
 c.execute("SELECT \* FROM data")  
 while True:  
 row = c.fetchone()  
 if row == None:  
 break  
 w = conn.cursor()  
 w.execute("SELECT \* FROM worker")  
 d = conn.cursor()  
 d.execute("SELECT \* FROM clothes")  
 while True:  
 work = w.fetchone()  
 if work == None:  
 w.close()  
 break  
 if (row[1] == work[0]):  
 while True:  
 cl = d.fetchone()  
 if cl == None:  
 d.close()  
 break  
 if (row[2] == cl[0]):  
 lbl6 = Label(window, text=str((row[0])))  
 lbl6.grid(column=a, row=b)  
 lbl7 = Label(window, text=str(work[1]))  
 lbl7.grid(column=a + 1, row=b)  
 lbl8 = Label(window, text=str(cl[1]))  
 lbl8.grid(column=a + 2, row=b)  
 lbl9 = Label(window, text=str(row[3]))  
 lbl9.grid(column=a + 3, row=b)  
 lbl10 = Label(window, text=str(row[4])+'/'+str(row[5]))  
 lbl10.grid(column=a + 4, row=b)  
 b += 1  
 break  
 btn7 = Button(window, text='Главное меню', command=clear)

btn7.grid(column=a + 5, row=1)  
#Меню выбора таблицы для вывода  
def pr():  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 btn4 = Button(window, text='Таблица: Спец. одежда', command = pc)  
 btn4.grid(column=1, row=1)  
 btn5 = Button(window, text='Таблица: Сотрудники', command = pw)  
 btn5.grid(column=1, row=2)  
 btn6 = Button(window, text='Таблица выдачи', command= pd)  
 btn6.grid(column=1, row=3)  
 btn7 = Button(window, text='Главное меню', command=clear)  
 btn7.grid(column=1, row=4)

#Подсчет закупки  
def tz\_done(txt,txt2,txt3,txt4):  
 window.geometry('400x950')  
 M = txt.get()  
 Y = txt2.get()  
 TM = txt3.get()  
 TY = txt4.get()  
 if (M == "") or (Y == "") or (TM == "") or (TY == ""):  
 error = Label(window, text='Введены некорректные данные')  
 error.grid(column=1, row=5)  
 return  
 elif (not M.isdigit()) or (not Y.isdigit()) or (not TM.isdigit())or (not TY.isdigit()):  
 error = Label(window, text='Введены некорректные данные')  
 error.grid(column=1, row=5)  
 return  
 elif (int(M) <= 0 and int(M)>12) or (int(Y) <= 2000) or (int(TM) <= 0 and int(TM)>12) or (int(TY) <= 2000):  
 error = Label(window, text='Введены некорректные данные')  
 error.grid(column=1, row=5)  
 return  
 l = Label(window, text = 'На '+str(M)+'/'+str(Y)+':')  
 l.grid(column = 1, row = 7)  
 z = 0  
 k = 0  
 r = 6  
 c.execute("SELECT \* FROM clothes")  
 while True:  
 cl = c.fetchone()  
 if cl == None:  
 break  
 a = conn.cursor()  
 a.execute("SELECT \* FROM data")  
 while True:  
 row = a.fetchone()  
 if row == None:  
 a.close()  
 k += z  
 lbl = Label(window, text= str((cl[1])))  
 lbl.grid(column = 1, row = r)  
 lbl2 = Label(window, text=str((z)))  
 lbl2.grid(column = 2, row = r)  
 r+=1  
 z = 0  
 break  
 if (row[2] == cl[0]):  
 if (((row[4] <= int(M)) and (row[5] <= int(Y))) and row[3] <= cl[3]):  
 if (int(Y) - row[5] == 0):  
 z += cl[3]  
 else:  
 z += (int(Y) - row[5]) \* cl[3]  
 elif ((row[3] < cl[3])):  
 if (int(Y) - row[5] == 0):  
 z += cl[3]  
 else:  
 z += (int(Y) - row[5]) \* cl[3]  
 lbl3 = Label(window, text="Итого: "+str((k)))  
 lbl3.grid(column = 1, row = r)  
 txt.delete(0, 'end')  
 txt2.delete(0, 'end')  
 txt.focus()

#Ввод пользователем даты закупки  
def tz():  
 window.geometry('400x250')  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
  
 lbl = Label(window, text = 'Введите месяц закупки:')  
 lbl.grid(column=1,row=1)  
 txt = Entry(window, width=20)  
 txt.grid(column = 2, row = 1)  
 txt.focus()  
 lbl2 = Label(window, text = 'Введите год закупки:')  
 lbl2.grid(column=1,row=2)  
 txt2 = Entry(window,width=20)  
 txt2.grid(column=2,row=2)  
 lbl3 = Label(window, text = 'Текущий месяц:')  
 lbl3.grid(column = 1, row = 3)  
 txt3 = Entry(window, width=20)  
 txt3.grid(column=2, row=3)  
 lbl4 = Label(window, text='Текущий год:')  
 lbl4.grid(column=1, row=4)  
 txt4 = Entry(window, width=20)  
 txt4.grid(column=2, row=4)  
 btn = Button(window, text = 'Подсчитать', command = lambda: tz\_done(txt,txt2,txt3,txt4))  
 btn.grid(column=3,row=5)  
 btn2 = Button(window, text = 'Главное меню', command = clear)  
 btn2.grid(column = 3, row=6)

#Удаление из таблицы "спец. одежда"  
def del\_cl(txt):  
 DEL = txt.get()  
 if (DEL == ""):  
 error = Label(window, text = 'Введены некорректные данные')  
 error.grid(column=1, row=3)  
 return  
 c.execute("DELETE FROM clothes WHERE clothname = '%s';" % DEL.strip())  
 conn.commit()  
 lbl = Label(window, text = 'Удалено.')  
 lbl.grid(column = 1, row = 3)  
 txt.delete(0, 'end')  
 txt.focus()  
#Ввод пользователем данных для удаления из таблицы "спец.одежда"  
def del\_c():  
 window.geometry('700x250')  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 lbl = Label(window, text = 'Введите наименование одежды, которую нужно удалить из базы данных:')  
 lbl.grid(column = 1, row = 1)  
 txt = Entry(window, width = 20)  
 txt.grid(column = 1, row = 2)  
 txt.focus()  
 btn = Button(window, text = 'Удалить', command = lambda: del\_cl(txt))  
 btn.grid(column = 2, row = 2)  
 btn1 = Button(window, text = 'Главное меню', command = clear)  
 btn1.grid(column = 2, row = 3)

#Удаление из таблицы "Сотрудники"  
def del\_wr(txt):  
 DEL = txt.get()  
 if (DEL == ""):  
 error = Label(window, text='Введены некорректные данные')  
 error.grid(column=1, row=3)  
 return  
 c.execute("DELETE FROM worker WHERE surname = '%s';" % DEL.strip())  
 conn.commit()  
 lbl = Label(window, text='Удалено.')  
 lbl.grid(column=1, row=3)  
 txt.delete(0, 'end')  
 txt.focus()  
#Ввод пользователем данных для удаления из таблицы "Сотрудники"  
def del\_w():  
 window.geometry('700x250')  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 lbl = Label(window, text='Введите ФИО сотрудника, которого нужно удалить из базы данных:')  
 lbl.grid(column=1, row=1)  
 txt = Entry(window, width=20)  
 txt.grid(column=1, row=2)  
 txt.focus()  
 btn = Button(window, text='Удалить', command=lambda: del\_wr(txt))  
 btn.grid(column=2, row=2)  
 btn1 = Button(window, text='Главное меню', command=clear)  
 btn1.grid(column=2, row=3)

#Меню выбора таблицы для удаления  
def dl():  
 window.geometry('400x250')  
 list = window.grid\_slaves()  
 for l in list: l.destroy()  
 btn4 = Button(window, text='Таблица: Спец. одежда', command = del\_c)  
 btn4.grid(column=1, row=1)  
 btn5 = Button(window, text='Таблица: Сотрудники', command = del\_w)  
 btn5.grid(column=1, row=2)  
 btn7 = Button(window, text='Главное меню', command=clear)  
 btn7.grid(column=1, row=4)

#Главное меню  
def gl():  
 window.geometry('150x150')  
 btn1 = Button(window, text='Заполнить таблицу', command = wr)  
 btn1.grid(column=1, row=0)  
 btn2 = Button(window, text='Вывести таблицу', command = pr)  
 btn2.grid(column=1,row=1)  
 btn4 = Button(window, text='Удалить из таблицы', command = dl)  
 btn4.grid(column=1, row=2)  
 btn3 = Button(window, text = 'Подсчет закупки', command = tz)  
 btn3.grid(column=1, row=3)  
  
  
window = Tk()  
window.title("Подчет закупки спец. одежды")  
  
window.geometry('150x150')  
gl()  
window.mainloop()