Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальныйисследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01– «Информатика и вычислительная техника»

**Лабораторная работа**

**по дисциплине**

**«Теория алгоритмов и структуры данных»**

**на тему**

**«АРМспециалист и задача коммивояжера»**

Выполнила студентка гр. ИВТ-23-1б

Хорошилова Ксения Павловна

Проверил:

Доцент каф. ИТАС

Яруллин Денис Владимирович

(оценка) (подпись)

(дата)

г. Пермь, 2024

Задача коммивояжера

**Цель и задачи работы**

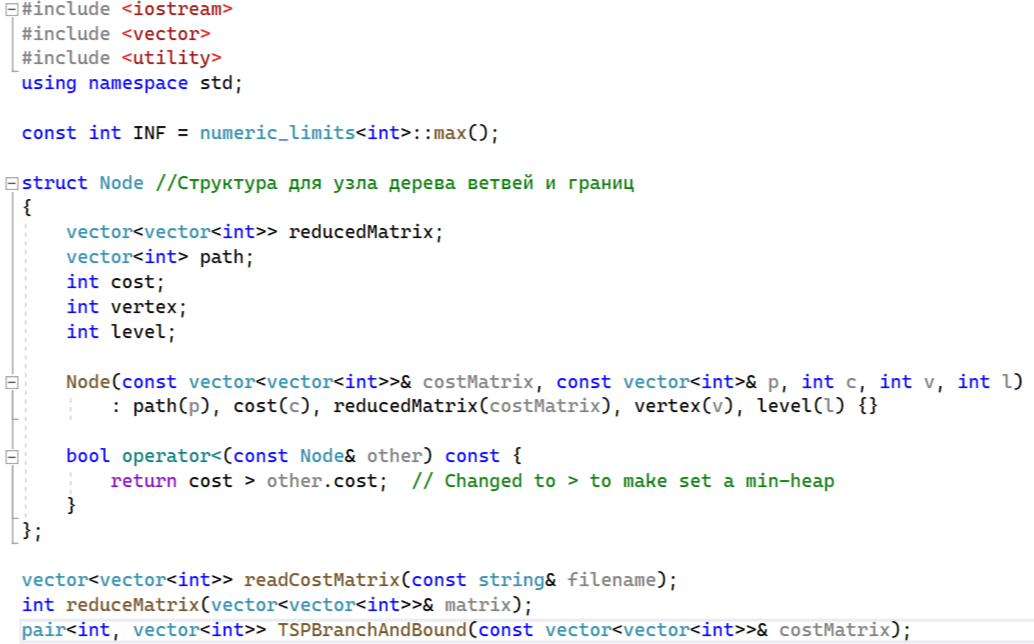
Изучить алгоритм решения задачи коммивояжера и научиться использовать его в графе.

**Анализ задачи коммивояжера**

Имеется N городов. Выезжая из исходного города А1, коммивояжер должен побывать во всех городах по одному разу и вернуться в город А1. Задача заключается в определении последовательности объезда городов, при которой коммивояжеру требуется минимизировать некоторый критерий эффективности: стоимость проезда, время пути, суммарное расстояние. Для расчета затрат существует матрица условий, содержащая затраты на переход из каждого города в каждый, при этом считается, что можно перейти из любого города в любой, кроме того же самого. Целью решения является нахождения маршрута, удовлетворяющего всем условиям и при этом имеющего минимальную сумму затрат. Требования к Задаче коммивояжера: 1. Написать программу, решающую задачу коммивояжера, указанную в варианте. 2. Для решения задачи коммивояжера использовать метод ветвей и границ. 3. Поэтапно продемонстрировать изменение исходной матрицы. 4. Исходная матрица храниться в текстовом файле. 7. Исходные данные должны приниматься с консоли.

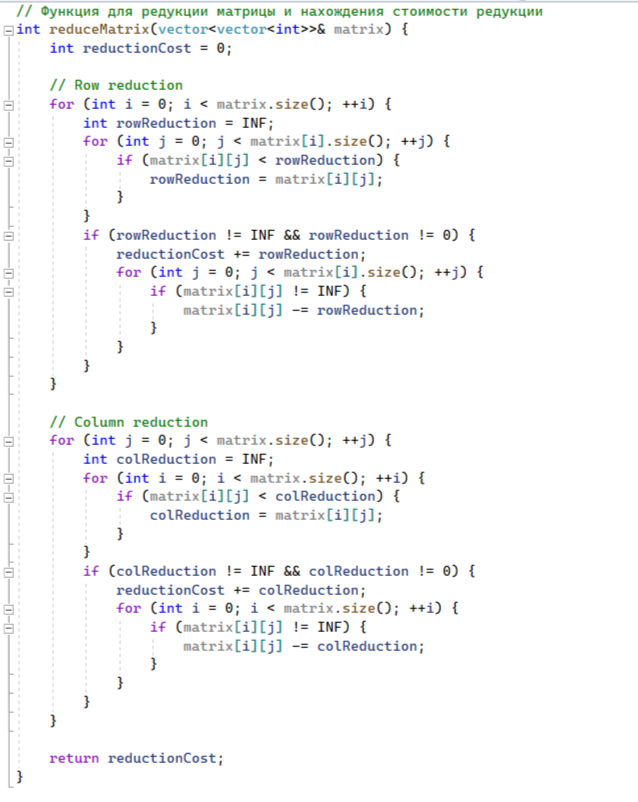
**Код**

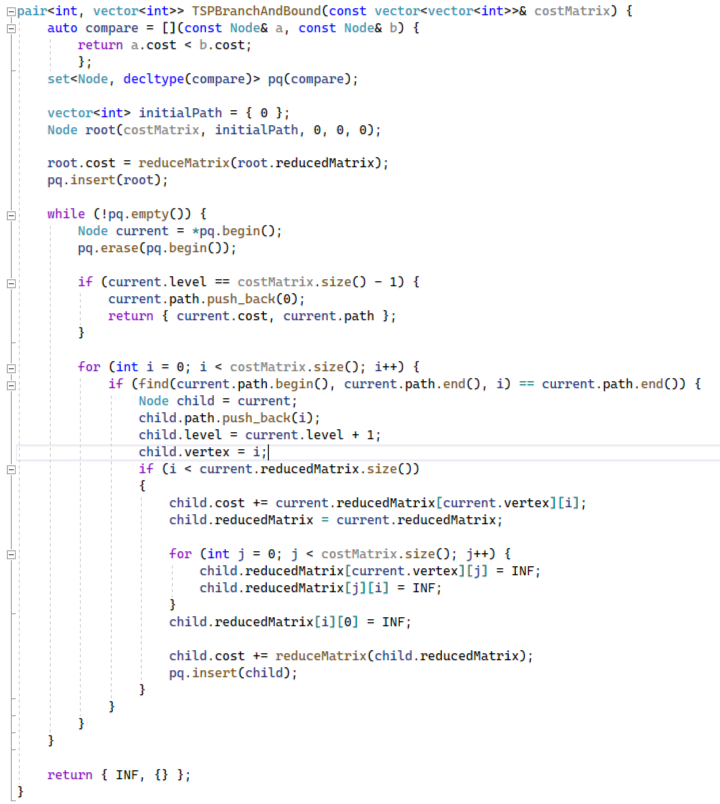
Файл ““kommivoyazher.h”



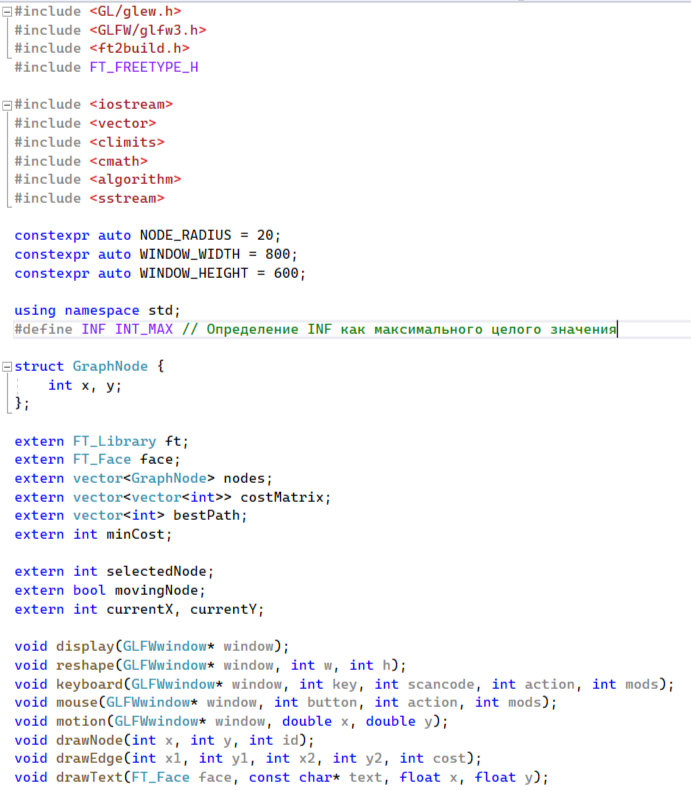
Файл “kommivoyazher.cpp”



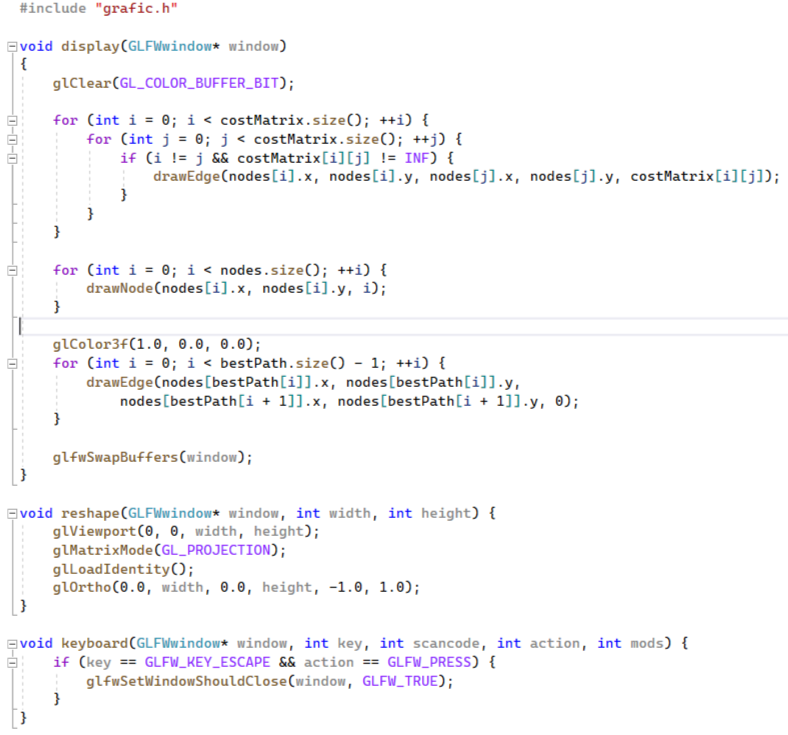


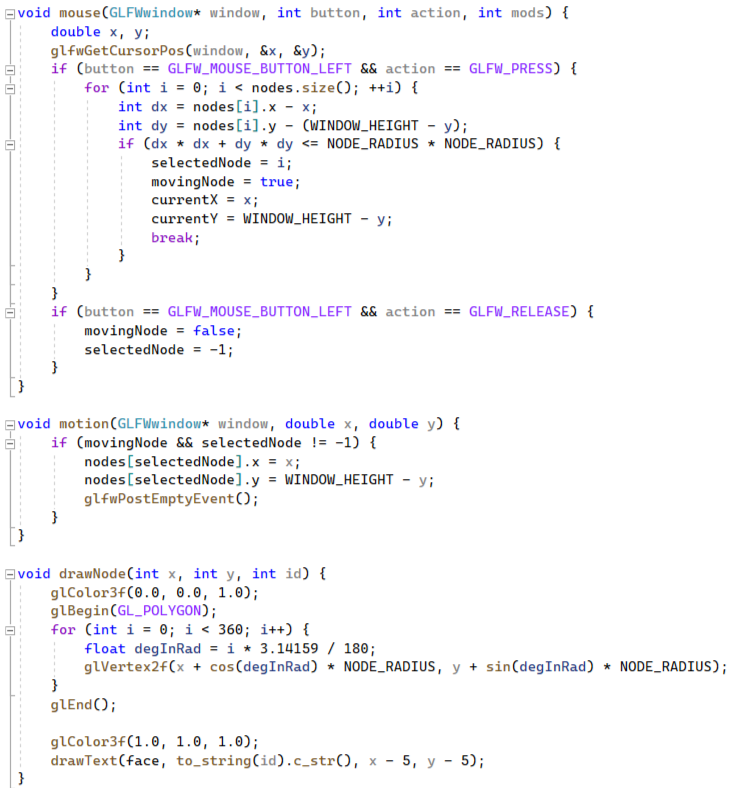
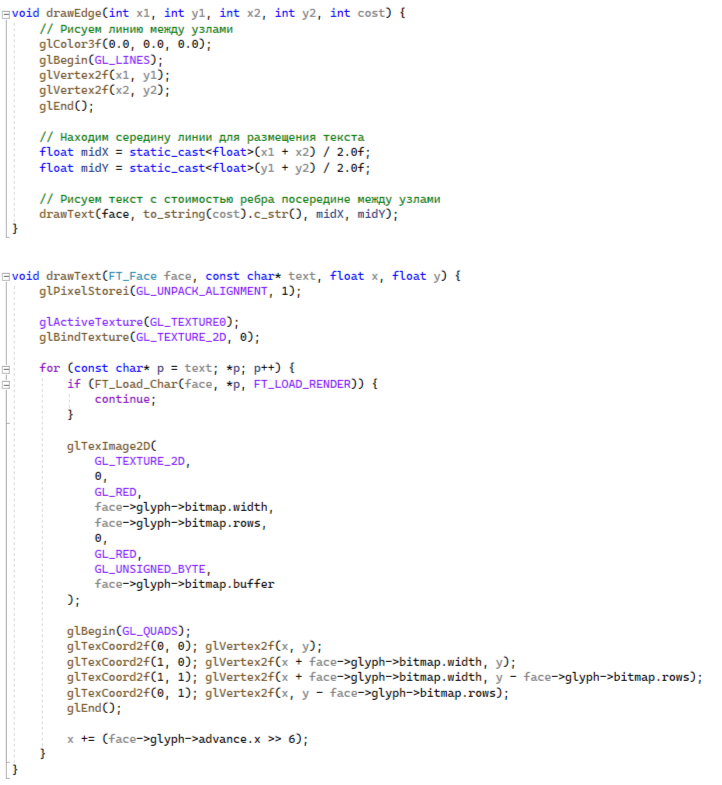


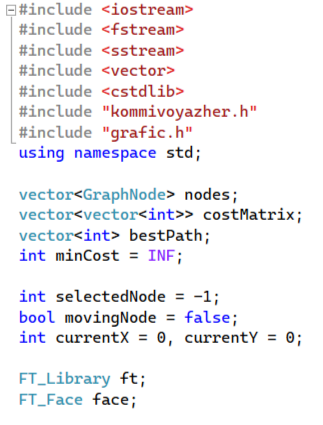
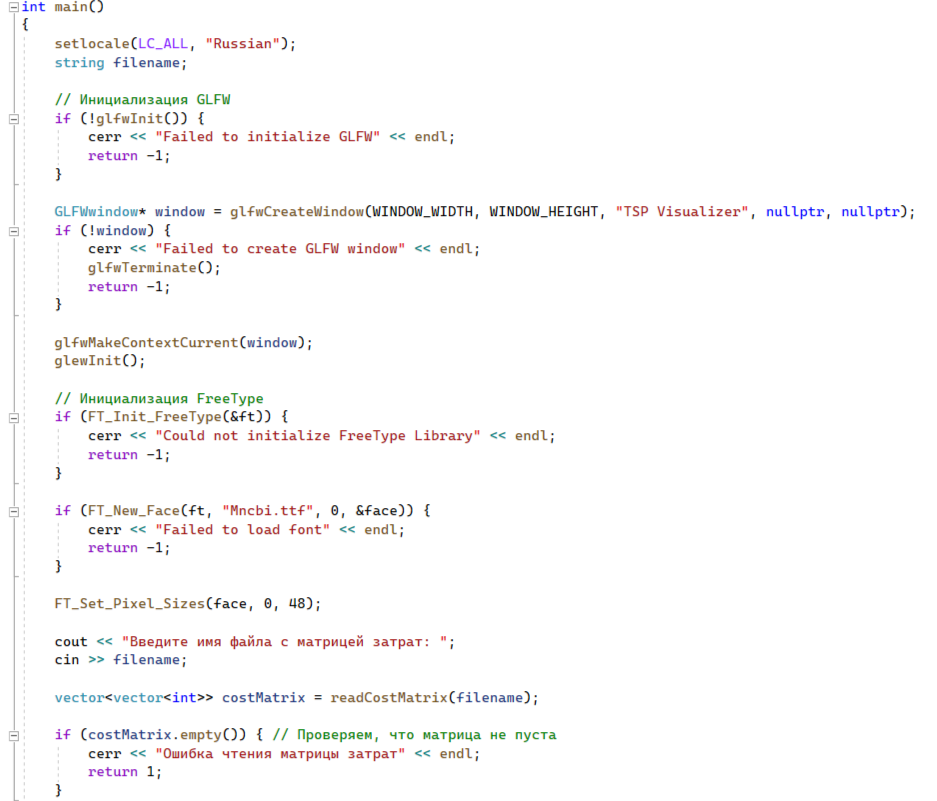
Файл “grafic.h”

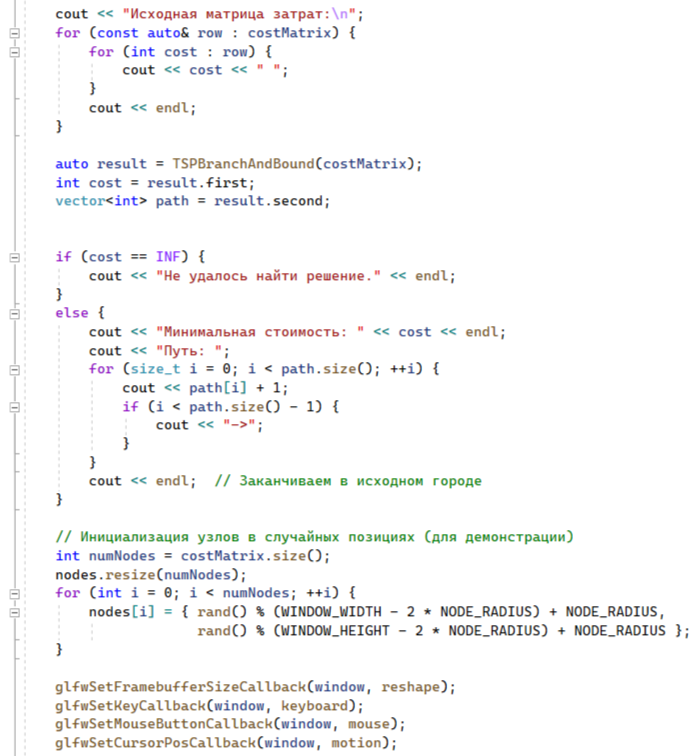


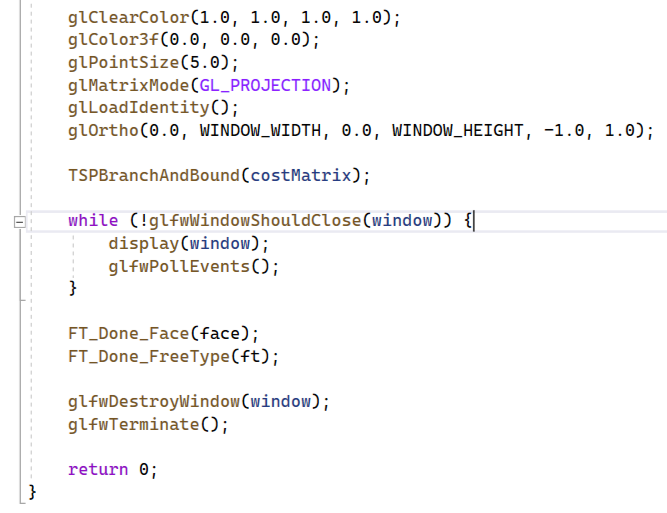
Файл “graphic.cpp”



Файл “main.cpp”





АРМспециалист

**Цели и задачи**

Реализовать приложение «Холодильник».

**Анализ задачи**

Программа, которая будет:

* Хранить перпечень продутов, их количество и срок годности;
* Позволит добавлять новые продукты в перечень продуктов;
* Позволит получать информацию о содержимом холодильника.

Хранение информации

Информация о каждом продукте – это

* Название продукта;
* Количество продукта;
* Срок хранения (дата до которой может храниться).

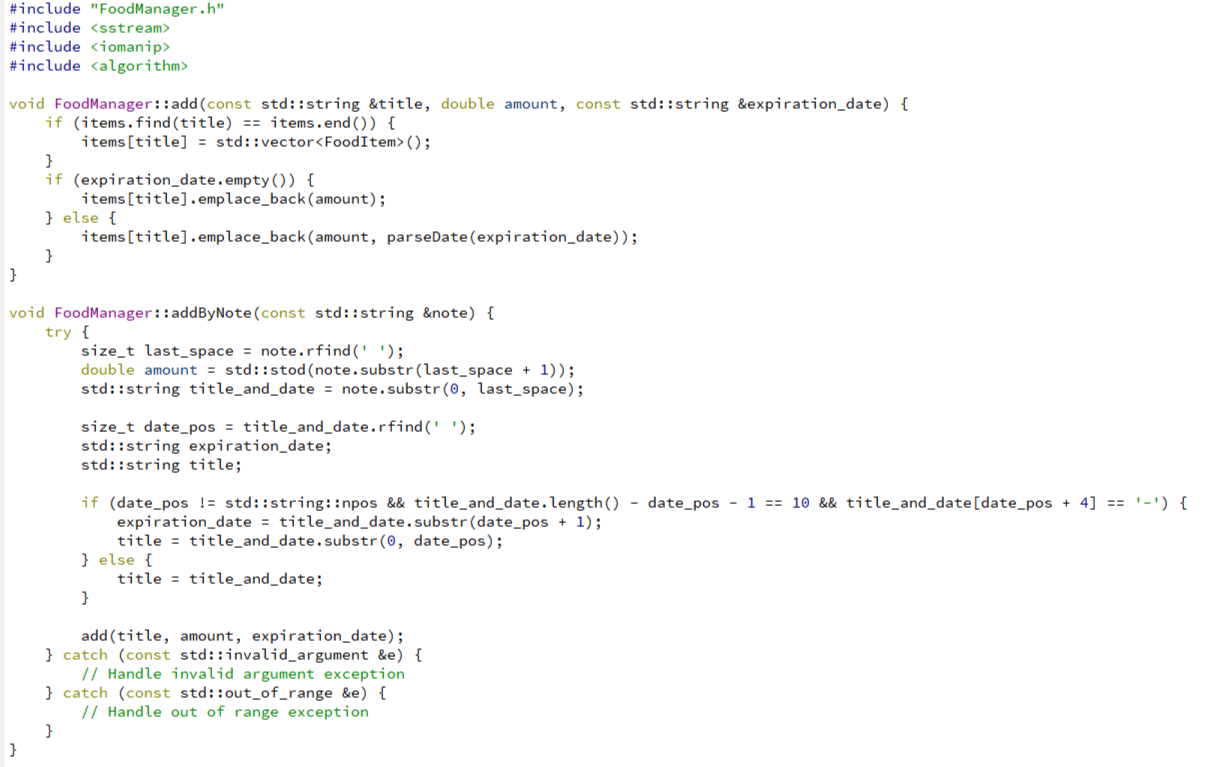
Информация о наборе продуктов в холодильнике должна храниться в виде словаря goods. В примере видно, что продукт «Вода» закуплен один раз в объёме 1.5 кг. Однако продукт с одним и тем же названием может быть закуплен и добавлен в холодильник несколько раз. Например, неделю назад пользователь купил килограмм пельменей под названием «Универсальные», половину съел, а потом пошёл и купил ещё два килограмма. Название у продукта одно и то же, но срок хранения может оказаться разным. Каждую такую закупку мы будем называть партия продукта. Таким образом, у любителя пельменей в холодильнике окажется две партии продукта «Пельмени Универсальные». В первой партии будет 0.5 кг продукта, во второй — 2 кг. Срок хранения у этих партий тоже будет разным. Ключ каждого элемента словаря — это название определённого продукта. Значение каждого элемента словаря — это список словарей. Каждый вложенный словарь содержит два элемента: amount — количество продукта. expiration\_date — срок годности. Количество продукта — это: целое число для исчисляемых продуктов (в штуках: например, значение 10 для яиц); дробное число для весовых продуктов (в килограммах: например, значение 1.5 для воды). Если срок годности не указан, вместо него должно быть указано None. Добавление и получение информации В программе должно быть пять функций для управления содержимым холодильника: add() — добавляет продукт в словарь goods; add\_by\_note() — добавляет продукт в словарь goods, преобразуя текстовое описание продукта в структурированные данные; find() — ищет в словаре goods заданное слово или строку и возвращает список продуктов, в названии которых есть это слово; amount() — возвращает количество запрошенного продукта; expire() — возвращает список просроченных продуктов. Опишите функции add(), add\_by\_note(), find(), amount() и expire(). С чего начать: Объявите словарь goods, добавьте в него пару продуктов — их можно скопировать из приведённых примеров. Займитесь функцией add() — научите её добавлять продукты в словарь. Протестируйте работу этой функции и переходите к следующей. Каждую готовую функцию вызовите несколько раз с разными аргументами: с необязательными аргументами и без них, со вчерашней и завтрашней датой (через сто лет — тоже попробуйте), передавайте в add() и add\_by\_note() новые продукты и те, что уже есть в словаре goods. Подсказки: Формат даты можно определить константой DATE\_FORMAT = '%Y-%m-%d'. Функция add(). Проверить, есть ли продукт (title) в словаре items. Преобразовать строку в дату с помощью модуля datetime. Применить append для добавления словаря с ключами 'amount' и 'expiration\_date' в список для конкретного title. Функция add\_by\_note(). Разделить строку на части с помощью str.split. Определить, является ли последняя часть строки датой. Нужную часть строки конвертировать в число типа Decimal Оставшуюся часть строки объединить, чтобы получить название продукта: если название состояло из нескольких слов — функция str.split разобъёт его на части. Вызвать функцию add(), передав в неё получившиеся данные — название, количество и срок хранения. Функция find(). Перебрать ключи словаря. Применить функцию lower, чтобы провести поиск без учёта регистра. Добавить найденные заголовки в результат поиска с помощью функции append. Функция amount(). Применить функцию find() для получения списка подходящих товаров. Суммировать значения amount для вычисления количества каждого найденного товара. Функция expire(). Получить текущую дату с помощью datetime.date.today(). Добавить определённое количество дней к текущей дате. Проверить, истекает ли срок годности товара: сравнить его с текущей датой и с датой истечения срока годности.

**Код**

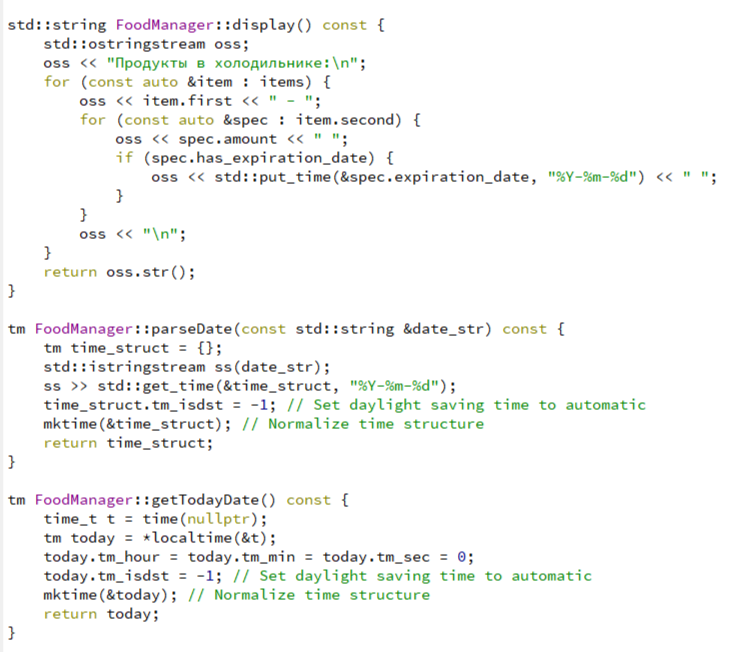
Файл “FoodManager.h”



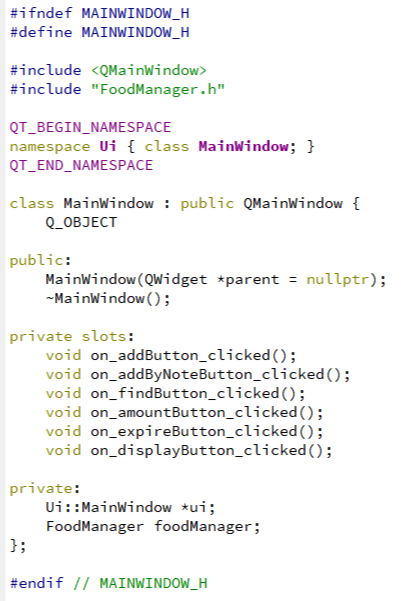
Файл “FoodManager.cpp”





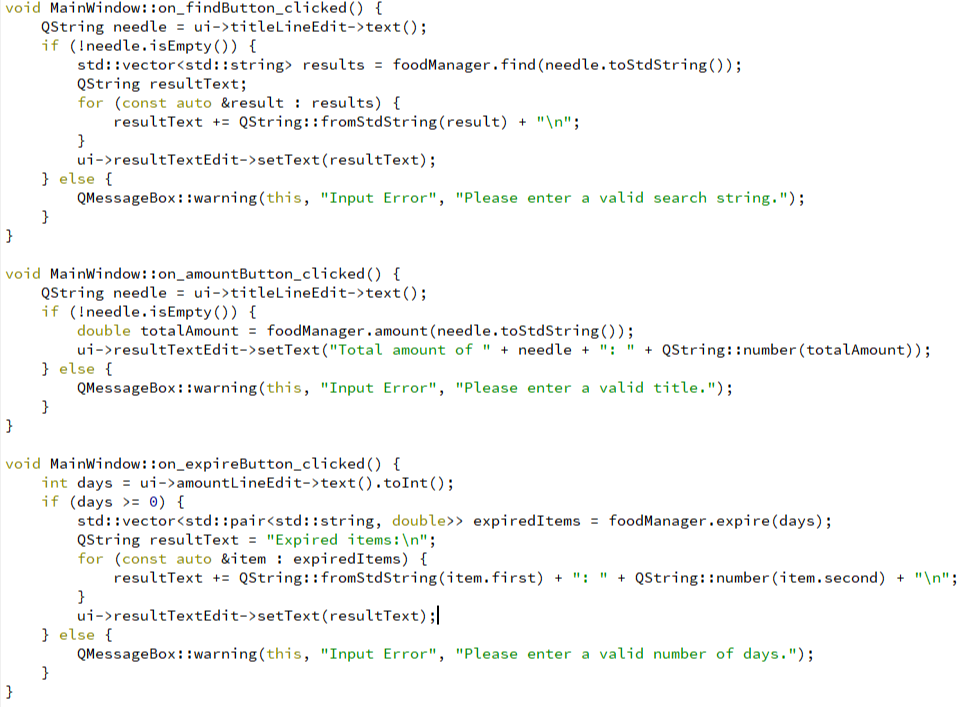


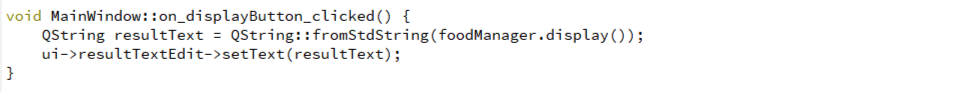
Файл “MainWindow.h”



Файл “MainWindow.cpp”







Файл “main.cpp”

