ГОСТ 19.104-78 С.6

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Московский Государственный Технический Университет имени Н.Э. Баумана» (МГТУ им. Н.Э. Баумана)

|  |  |
| --- | --- |
| Согласовано  Преподаватель кафедры ИУ7 Романов Алексей Сергеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Романов А.С.) 1.03.2015 | Утверждено  Доцент кафедры ИУ7, кандидат физико-математических наук Романова Татьяна Николаевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Романова Т.Н.) 1.03.2015 |

Распределенная система обработки информации по заказу электроники из интернет-магазинов.

Лист утверждения  
Листов\_\_\_

Представители предприятия-разработчика

Доцент кафедры ИУ-7  
кандидат физико-математических наук  
Романова Татьяна Николаевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Романова Т.Н.)  
\_\_.05.2015

Студент МГТУ им. Баумана  
Косюра Ольга Владиславовна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Косюра О.В.)

**2015**

# 1 Техническое задание

## Глоссарий

|  |  |
| --- | --- |
| **Термин** | **Определение** |
| Валидация данных | Проверка на корректность, полноту и непротиворечивость входных, выходных и обрабатываемых данных |
| WEB-интерфейс | Интерфейс пользователя, предоставляемой системой через Web-браузер |
| Хеширование | Преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины |

## Принятые сокращения

|  |  |
| --- | --- |
| **Сокращение** | **Расшифровка** |
| ОС | Операционная система |
| СПО | Специальное программное обеспечение |

## Введение

Данное техническое задание разработано на программный комплекс «Распределенная система обработки информации по заказу электроники из интернет-магазинов», в дальнейшем именуемое система заказа электроники.

Данное техническое задание формируется на основе ГОСТ 9.201-78 «Техническое задание. Требования к содержанию и оформлению».

### Краткое описание предметной области

Проблема просмотра и выгодного заказа электронных товаров в интернете в последнее время является актуальной. Это связано с тем, что каждый интернет-магазин продает схожий набор товаров по разной цене. Также, помимо цены, разнится и цена за доставку. Покупатель, заказывая товар из интернет-магазина, надеется, что товар придет ему соответствующего качества и за минимальную цену, но часто выходит, что это не так, так как цена за этот же товар в другом интернет-магазине ниже.

### Существующие аналоги

### На данный момент существует сервис “Яндекс.Маркет”, на котором размещены товары и магазины, перейдя на сайт которых можно заказать то или иное устройство, но тогда, когда сайт магазина не доступен, заказать товар невозможно.

## 1.1 Описание системы

Для систематизации построения разрабатываемая система должна быть разделена на сервисы. Разбиение на сервисы призвано разделить и изолировать отдельные участки системы, структурировать построение, упростить формирование требований к отдельным частям системы.

При построении должны быть реализованы следующие системы:

1.       Подсистема “Фронтенд”

2.       Подсистема “Сервис сессии”

3.       Подсистема “Сервис логики”

4.       Подсистема “Сервис по работе с покупателем”

5.       Подсистема “Сервис по работе с фирмами”

6.       Подсистема “Сервис обработки заказов”

7.       Подсистема “Сервис по работе с товарами”

Подсистема “Фронтенд” отвечает за получение входной информации от пользователя (покупателя или фирмы) и обработку полученной информации.

Подсистема “Сервис сессии” предназначен для создания и проверки сессии пользователя. Также через этот сервис осуществляется доступ к базам данных «Фирмы» и «Покупатели», в которых хранятся логины и пароли пользователей.

Подсистема “Сервис по работе с покупателем” предназначен для обработки запросов пользователя-покупателя.

Подсистема “Сервис по работе с фирмами” предназначен для обработки запросов пользователя-фирмы. Также через этот сервис осуществляется доступ к базе данных «Фирма – Товар».

Подсистема “Сервис по работе с товарами” предназначен для доступа к базе данных «Товары».

Подсистема “Сервис обработки заказов” предназначен для доступа к базам данных «Заказы» и «Состав заказов».

Подсистема “Сервис логики” предназначен для обработки запросов и данных, полученных от других сервисов.

Рисунок 1 Архитектура разрабатываемой системы

## 1.2 Основания для разработки

Разработка ведется в рамках выполнения лабораторных работ по курсу «Методология программной инженерии» на основании утверждённого учебного плана. Разработка РСОИ должна включать в себя следующие этапы:

1. Разработка и утверждение технического задания, протокола о принципах взаимодействия систем и уточнения основных особенностей функционирования РСОИ.
2. Разработка программного обеспечения, в том числе разработку релизов. В первом релизе должны быть реализованы функции, которые позволяют оценить правильность стратегии в выборе архитектурных решений.
3. Подготовка сопроводительной документации
4. Внедрение системы
5. Тестирование в реальных условиях

## 1.3 Требования к системе

### 1.3.1 Требования к РСОИ

1. Разрабатываемое программное обеспечение должно обеспечивать функционирование системы в режиме 24/7/365.
2. Время восстановления системы после сбоя не должно превышать 3-х часов.
3. Система должна поддерживать возможность «горячего» переконфигурирования системы.
4. Система должна обеспечивать хранение заказов, данных о пользователях и товарах
5. Система должна обеспечивать децентрализацию сбора информации с помощью формирования вынесенных локальных серверов сбора информации, которые, в свою очередь, передают информацию в центральный сервер
6. Система должна обеспечивать визуализацию данных о товарах и заказах
7. Система должна поддерживать добавление новых пользователей
8. Система не должна выходить из строя при выходе из строя одной из подсистем
9. Обмен информацией в системе должен производиться исходя из предположения, что каналы связи небезопасны и ненадежны

### 1.3.2 Требования к функциональным характеристикам

1. Время реактивности системы -3-5сек. (по требованию заказчика);
2. Время выдачи списка предложений от фирмы - не более 7 сек.;
3. Синхронные запросы между подсистемами не должны превышать 10 секунд;
4. Время отклика системы на запрос пользователя не должно превышать 20 секунд;
5. Время отклика системы на запрос «Зарегистрировать пользователя в системе» не должно превышать 21 секунды;
6. Обеспечить заданные временные характеристики для 20 одновременно подключённых пользователей.

### 1.3.3 Требования к реализации

1. Система должна обеспечивать возможность запуска всех компонентов как сервисов ОС;
2. Интерфейсы управления должны быть реализованы в виде WEB-интерфейсов;
3. Пароли учетных записей должны подвергаться хешированию;
4. Система должна предоставлять как минимум два интерфейса - интерфейс пользователя - покупателя и интерфейс пользователя – магазина;
5. Для хранения данных о пользователях, товарах, заказах и их составов использовать СУБД SQLite.
6. Для разработки сервисов использовать язык программирования Python 3.4, используя фреймворк Flask.

### 1.3.4 Функциональные требования к системе интернет магазина

#### 1.3.4.1 Общие функциональные требования к системе:

Система должна представлять собой законченный программный комплекс. Внедряемая система должна обеспечивать выполнение следующих действий и функций:

1. С точки зрения покупателя

* Аутентификация пользователей
* Регистрация пользователей
* Просмотр списка всех товаров
* Просмотр списка всех фирм
* Просмотр товаров, которыми торгует определенная фирма
* Просмотр фирм, которые торгуют определенным товаром
* Просмотр данных о товаре
* Составление заказа
* Просмотр состава заказа

1. С точки зрения менеджера фирмы

* Аутентификация пользователей
* Регистрация пользователей
* Просмотр списка всех товаров
* Просмотр списка всех фирм
* Просмотр товаров, которыми торгует определенная фирма
* Просмотр фирм, которые торгуют определенным товаром
* Просмотр данных о товаре
* Просмотр заказов
* Редактирование статуса заказа для своей фирмы
* Редактирование предоставляемых товаров своей фирмы
* Просмотр состава заказа

#### 1.3.4.2 Входные данные

Входными параметрами для системы являются:

* + 1. Логин и пароль пользователя (строки не более 20 символов каждая)
    2. Код сессии (строка длиной 20 символов)
    3. Данные о товаре
  + Id товара(целое число)
  + Марка производителя(строка)
  + Модель(строка)
  + Цвет(строка)
  + Сопроводительный текст(строка)

4. Данные о заказе

* + Id пользователя
  + Id магазина
  + Id товара

5. Данные о фирме

* Продаваемые товары (массив целых чисел – id товаров)
* Цена (Действительное число)

#### 1.3.4.3 Выходные данные

На выходе система должна выдавать:

* + - 1. Список товаров (массив целых чисел – id товаров)
      2. Список фирм (массив целых чисел – id фирм)
      3. Описание фирм
* Название (строка)
* Продаваемый товар (массив целых чисел – id товаров)
  + - 1. Полное описание товара с ценой
* Наименование(строка)
* Марка(строка)
* Модель (строка)
* Цвет(строка)
* Фирма продавец (целое число – id фирмы)
* Сопроводительный текст(строка)
* Цена (Действительное число)
  + - 1. Данные о заказе
* Фирма продавец (целое число – id фирмы)
* Товар (целое число)
* Цена (действительное число)

### 1.3.5 Функциональные требования к системе фирмы

#### 1.3.5.1 Общие функциональные требования к системе:

Система должна представлять собой законченный программный комплекс. Внедряемая система должна обеспечивать выполнение следующих действий и функций:

1. С точки зрения менеджера фирмы:
   * + - Аутентификация пользователей
       - Регистрация пользователей
       - Просмотр товаров, которыми торгует определенная фирма
       - Просмотр данных о товаре
       - Просмотр заказов
       - Редактирование товара
       - Редактирование списка товаров
       - Редактирование статуса заказа
       - Добавление товара
       - Удаление товара
       - Просмотр состава заказа

#### 1.3.5.2 Входные данные

Входными параметрами для системы являются:

* + Информация о заказе
    1. Id пользователя
    2. Id магазина
    3. Id товара
  + Логин и пароль пользователя
  + Данные о товаре
    1. Наименование (строка)
    2. Марка(строка)
    3. Модель (строка)
    4. Цвет(строка)
    5. Сопроводительный текст(строка)
    6. Цена (действительное число)

#### 1.3.5.3 Выходные данные

На выходе система должна выдавать:

* + Список товаров (массив целых чисел – id товаров)
  + Описание товара
    1. Наименование (строка)
    2. Марка (строка)
    3. Модель (строка)
    4. Цвет (строка)
    5. Сопроводительный текст (строка)
    6. Цена (действительное число)
  + Данные о заказе
    1. Цена (действительное число)
    2. Покупатель (целое число – id пользователя)
    3. Состав (массив)
    4. Состояние (строка)

### 1.3.6 Функциональные требования к системе удаленного сервера

#### 1.3.6.1 Общие функциональные требования к системе:

Система должна представлять собой законченный программный комплекс. Внедряемая система должна обеспечивать выполнение следующих действий и функций:

* + - 1. Резервное копирование базы данных
      2. Восстановление базы данных

#### 1.3.6.2 Входные данные

Входными параметрами для системы являются:

* + 1. Актуальная копия базы данных

#### 1.3.6.3 Выходные данные

На выходе система должна выдавать:

* + - 1. Резервная копия базы данных

## 1.4 Условия эксплуатации

Для обеспечения бесперебойного функционирования РСОИ на центральном сервере необходимо установить кондиционеры, поддерживающие следующие климатические условия:

1. Обеспечить температурный режим от 15 до 30 градусов Цельсия;
2. Относительная влажность воздуха - не более 80%;
3. Расположить компоненты сервера на жестких металлических стойках;
4. Оборудование должно быть заземлено.

## 1.5 Требования к составу и параметрам технических средств

Минимальные технические требования для центрального сервера:

1. 2-х ядерный процессор с тактовой частотой 2 ГГц;
2. ОЗУ 4 ГБ;
3. ПЗУ 300 ГБ для РСОИ;
4. Сетевая карта Ethernet стандарта 100BASE-T;
5. Монитор с разрешение экрана 1366х768 точек;
6. Клавиатура;
7. Мышь.

Общие требования к программному окружению:

1. Web-браузер Chrome 17.0.

## 1.6 Требования к надежности

Для повышения надёжности необходимо предусмотреть возможность установки дублирующего сервера для сервера, обеспечивающего работу пользователей через web-интерфейс, поскольку он является наиболее уязвимым и важным звеном в архитектуре системы.

Администратор баз данных должен обеспечить периодическое создание резервных копий базы данных (1 раз в сутки).

Для предотвращения сбоев в работе СУБД при сбое в подаче электропитания, необходимо обеспечить подключение серверов к устройствам бесперебойного электропитания, которые обеспечат не менее 30 минут автономной работы.

## 1.7 Требования к документации

Документация должна включать:

1. Руководство по развертыванию системы;
2. Руководство по использованию системы;
3. Документация к системе должна быть подготовлена на русском языке и представлена как на бумаге, так и в электронном виде.

# Технический проект

## 1 Сценарий использования системы

Пользователям системы являются физические лица, которые хотят оформить заказ в электронном магазине. Таким образом, в системе выделяется две роли пользователей – «покупатель» и «продавец». Диаграмма прецедентов пользователя подсистемы электронного магазина изображена на рисунке 2.



Рисунок 2 Диаграмма прецедентов пользователя подсистемы электронного магазина.

### 1.1 Спецификация прецедента «Регистрация»

Цель прецедента использования: клиент регистрируется в системе, сообщая почтовый адрес, логин и пароль.

Оптимистический сценарий:

* Клиент вводит данные в окне веб-страницы
* Система правильность заполнения полей
* Система запоминает данные в таблице базы данных
* Система перенаправляет пользователя на страницу пользователя

Альтернативный сценарий:

* Клиент вводит данные в окне веб-страницы
* Система правильность заполнения полей
* Система помечает поля с неверно введенными данными

### 1.2 Спецификация прецедента «Вход в систему»

Цель прецедента использования: клиент входит в систему под своим аккаунтом.

Оптимистический сценарий:

* Клиент вводит данные в окно веб-страницы
* Система проверяет наличие данных о пользователе в базе данных
* Если данные существуют, система перенаправляет пользователя на страницу пользователя.

Альтернативный сценарий:

* Клиент вводит данные в окно веб-страницы
* Система проверяет наличие данных о пользователе в базе данных
* Если данные отсутствуют, система выводит сообщение о неверных введенных данных

### 1.3 Спецификация прецедента «Просмотр всех товаров»

Цель прецедента использования: клиент осуществляет просмотр списка всех товаров.

Оптимистический сценарий:

* Система проверяет наличие товаров в базе данных
* Если данные существуют, система показывает список товаров в окне веб-страницы

Альтернативный сценарий:

* Система проверяет наличие товаров в базе данных
* Если данные отсутствуют, система показывает сообщение о том, что товары не найдены

### 1.4 Спецификация прецедента «Просмотр всех фирм»

Цель прецедента использования: клиент осуществляет просмотр списка всех фирм.

Оптимистический сценарий:

* Система проверяет наличие товаров в базе данных
* Если данные существуют, система показывает список фирм в окне веб-страницы

Альтернативный сценарий:

* Система проверяет наличие товаров в базе данных
* Если данные отсутствуют, система показывает сообщение о том, что данные не найдены

### 1.5 Спецификация прецедента «Просмотр товара»

Цель прецедента использования: клиент осуществляет просмотр информации о товаре.

Оптимистический сценарий:

* Система проверяет наличие товара в базе данных
* Если данные существуют, система показывает данные о товаре в окне веб-страницы

Альтернативный сценарий:

* Система проверяет наличие товара в базе данных
* Если данные отсутствуют, система показывает сообщение о том, что данные не найдены

### 1.6 Спецификация прецедента «Сделать заказ»

Цель прецедента использования: клиент осуществляет составление заказа.

Оптимистический сценарий:

* Система проверяет наличие товара в базе данных
* Если данные существуют, система перенаправляет пользователя на страницу подтверждения заказа
* После подтверждения заказа, заказ заносится в базу данных

Альтернативный сценарий:

* Система проверяет наличие товара в базе данных
* Если данные существуют, система перенаправляет пользователя на страницу подтверждения заказа
* Если подтверждения заказа не произошло, система перенаправляет пользователя на страницу списка всех товаров

### 1.7 Спецификация прецедента «Просмотр заказа»

Цель прецедента использования: клиент осуществляет просмотр заказа.

Оптимистический сценарий:

* Система проверяет наличие заказа в базе данных
* Если данные существуют, система показывает данные о заказе в окне веб-страницы

Альтернативный сценарий:

* Система проверяет наличие заказа в базе данных
* Если данные отсутствуют, система перенаправляет пользователя на страницу с описанием ошибки

### 1.8 Спецификация прецедента «Просмотреть все заказы»

Цель прецедента использования: клиент осуществляет просмотр списка всех заказов.

Оптимистический сценарий:

* Система проверяет наличие заказов в базе данных
* Если данные существуют, система показывает данные о товаре в окне веб-страницы

Альтернативный сценарий:

* Система проверяет наличие заказов в базе данных
* Если данные не существуют, система показывает сообщение о том, что ни одного заказа пока не сделано

## 2 Архитектура системы

Проведенный анализ показал, что в состав РСОИ входят три типа узлов, проектирование которых рассматривается в рамках данного проекта:

* Система интернет-магазина
* Система фирмы
* Система удаленного сервера

Система интернет-магазина должна предоставлять WEB-интерфейс пользователю для взаимодействия.

На следующем рисунке показаны компоненты РСОИ и интерфейсы, которыми они соединены.

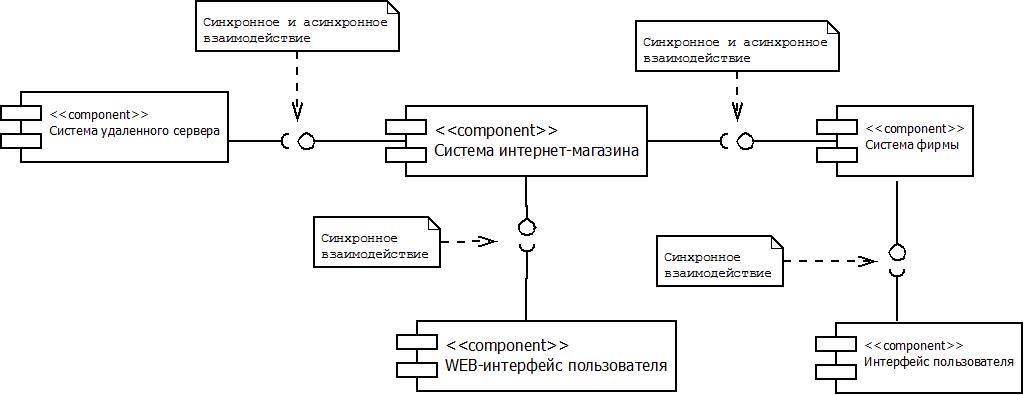


Рисунок 3. Диаграмма компонент

## 3 Модель данных

ER-диаграмма модели данных для системы интернет-магазина изображен на следующем рисунке.

Рисунок 4. ER-диаграмма модели данных для системы интернет-магазина

## 4 Диаграмма классов



Рисунок 5. Концептуальная диаграмма классов

### 4.1 Класс User

Класс представляет собой профиль покупателя. Используется для работы с интернет-магазином.

Атрибуты класса User:

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| ID | public : int | Идентификатор пользователя |
| Name | public: string | Имя пользователя |
| Surname | public: string | Фамилия пользователя |
| password\_hash | public: string | MD5-хэш пароля пользователя |

Методы Класса User:

|  |  |
| --- | --- |
| Имя атрибута | Описание |
| User(string, string) | param: email [ string - in ]  param: password\_hash [ string - in ] Конструирует объект профиля пользователя из строк адреса электронной почты и хэша пароля |
| add\_order(Order) : int | param: new\_order [ string - in ] Добавить заказ |
| Show() : int | Просмотр данных |

### 4.2 Класс Firm

Класс представляет собой профиль фирмы.

Используется для работы с интернет-магазином.

Атрибуты класса Firm:

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| ID | public : int | Идентификатор пользователя |
| Name | public: string | Имя пользователя |
| Surname | public: string | Фамилия пользователя |
| password\_hash | public: string | MD5-хэш пароля пользователя |

Методы Класса Firm:

|  |  |
| --- | --- |
| Имя атрибута | Описание |
| Firm(string, string) | param: email [ string - in ]  param: password\_hash [ string - in ] Конструирует объект профиля пользователя из строк адреса электронной почты и хэша пароля |
| edit\_order(Order) : int | param: id\_order [ string - in ] Редактировать статус заказа |
| Show() : int | Просмотр данных |
| add\_goods(Good) : int | param: new\_good [ string - in ] Добавить новый товар |
| Edit\_order(Order) : int | param: id\_good [ string - in ] Редактировать товар |

### 4.3 Класс Goods

Класс представляет собой данные о товаре.

Атрибуты класса Goods:

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| ID | public : int | Идентификатор товара |
| Name | public: string | Наименование товара |
| Mark | public: string | Марка товара |
| Model | public: string | Модель |
| Color | public: string | Цвет |
| Description | public: string | Описание товара |

Методы Класса Goods:

|  |  |
| --- | --- |
| Имя атрибута | Описание |
| Add\_good(string, string,string,string,string) | param: name [ string - in ]  param: Mark [ string - in ] param: Model [ string - in ]  param: color [ string - in ] param: description [ string - in ]  Добавление товара |

### 4.4 Класс GoodFirm

Класс GoodFirm представляет собой связку между товаром и продавцом.

Атрибуты класса GoodFirm:

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| IDGood | public : int | Идентификатор товара |
| Price | public: int | Цена товара |
| IDFirm | public: int | Идентификатор продавца |

## 5 Бизнес-логика

В задачи бизнес-логики входит:

* Обработка сообщений пользователя
* Отправка запросов узлам РСОИ, согласно логике обработки
* Обработка ответов от узлов РСОИ

Структура подсистемы бизнес-логики изображена на следующем рисунке.



Рисунок 8. Диаграмма компонент подсистемы бизнес-логики.

## 6 Диаграмма пакетов

В этом разделе представлена диаграмма пакетов разрабатываемой системы интернет-магазина.



Рисунок 9. Диаграмма пакетов системы интернет-магазина

## 7 Диаграмма размещения компонентов

В этом разделе представлена диаграмма размещения разрабатываемой системы.



Рисунок 10. Диаграмма размещения разрабатываемой системы.

## 8 Диаграмма последовательности подсистемы интернет-магазина

В данном разделе представлена диаграмма последовательности разрабатываемой системы.



Рисунок 11. Диаграмма последовательности

## 9 Диаграмма активности

В данном разделе представлена диаграмма активности для составления заказа



Рисунок 12. Диаграмма активности

# Тестирование

## 1 Венгерский метод

Венгерский метод применяется в решении задачи о назначениях.

*Содержательная постановка задачи о назначениях*

Пусть необходимо выполнить  работ. Для этого используется  исполнителей, каждый из которых в состоянии выполнить любую работу. Известны затраты  на выполнение j-ым исполнителем i-й работы. Требуется назначить каждого исполнителя на одну работу так, чтобы минимизировать суммарные затраты.

*Математическая постановка задачи о назначениях*

Необходимо минимизировать значение целевой функции Z:



при ограничениях:





Причем xij = 1, если i-ая работа выполняется j-ым исполнителем, иначе xij=0

### 1.1 Требования к программе

Программа должна:

1. реализовать венгерский метод
2. быть реализована на языке С
3. в качестве входных данных должна принимать двумерную матрицу (int) и ее размерность
4. по завершении работы программы во входной матрице должно содержаться решение

### 1.2 Анализ входных данных

Разобьем входные данные на классы эквивалентности: корректные и некорректные данные.

Входными данными являются:

* Двумерная матрица из элементов типа int
* Размерность матрицы

*Класс корректных данных*

Согласно постановке задачи о назначениях все элементы входной матрицы неотрицательны. Размерность матрицы должна быть положительна.

Следовательно, в качестве корректных входных данных выступают матрицы С: cij >=0 , i = , j = , где n – размерность матрицы (n > 0)

*Класс некорректных данных*

В качестве некорректных данных следует задать матрицы, в которых хотя бы один элемент отрицателен. Следует также проверить поведение программы при заданной размерности n = 0 и n < 0, при передаче NULL в качестве значения указателя на двумерную матрицу.

### 1.3 Анализ граничных условий

Следует проверить пограничные значения (это подклассы корректных значений):

* все элементы матрицы имеют значения INT\_MAX
* все элементы матрицы имеют значения 0
* размерность матрицы n = 1
* в матрице максимальные значения среди элементов матрицы располагаются по диагонали
* в матрице минимальные значения среди элементов матрицы располагаются по диагонали

### 1.4 Тестовые входные данные

|  |  |  |  |
| --- | --- | --- | --- |
| Матрица | Размерность | Результат работы программы | Резолюция |
|  | 2 | [1, 0]  [0, 1] | Верно |
|  | -1 | Экстренное завершение | Ошибка |
|  | 0 | Программа вернула код ошибки -1 | Верно |
| NULL | 3 | Программа вернула код ошибки -1 | Верно |
|  | 2 | [0, 1]  [1, 0] | Ошибка |
|  | 2 | [0, 1]  [1, 0] | Ошибка |
|  | 2 | [1, 2]  [-1, 10] | Ошибка |
|  | 2 | [1, 0]  [0, 1] | Ошибка |
|  | 2 | [1, 0]  [0, 1] | Верно |
|  | 2 | [1, 0]  [0, 1] | Верно |
| (2) | 1 | [1] | Верно |
|  | 2 | [0, 1]  [1, 0] | Верно |
|  | 2 | [1, 0]  [0, 1] | Верно |

Код тестовой программы представлен в приложении 1.

***Выводы***

В результате данного этапа тестирования было выявлено, что в программе имеются следующие ошибки, которые необходимо устранить:

* при отрицательной размерности матрицы программа экстренно завершается , хотя должна возвращать ошибку
* программа работает с матрицей, имеющей отрицательные элементы. Корректное поведение программы – это завершение с возвращением кода ошибки.

***Проверка функциональности***

*1. Проверка расстановки единиц*

Известно, что в результате работы венгерского алгоритма всегда получается матрица, состоящая из единиц и нулей, причем в каждом столбце и каждой строке матрицы содержится ровно одна единица. Необходимо проверить, удовлетворяет ли решение, предоставленное на выходе тестируемой программы, этому требованию.

*2. Проверка нахождения минимального значения целевой функции*

Расстановка единиц должна обеспечивать минимальное значение целевой функции Z.

Предлагается проверить данное требование путем генерирования всех возможны матриц с уникальной расстановкой единиц и нулей. Для матрицы размерности n количество таких уникальных матриц n!.  
Далее для полученных матриц следует рассчитать значение целевой функции и среди всех полученных значений выбрать минимальное.

Программа должна выдать такое же решение, как и полученное с помощью перебора минимальное значение.

Стоит заметить, что такой способ подходит для тестирования матриц размерности не больше 8 т.к. 8! = 40320. Те необходимо хранить в памяти 40320 матриц.

В качестве решения можно предложить генерировать единичные матрицы заранее и сохранять их в файле. При прохождении теста загружать их из файла.

## ***2 Функциональное тестирование***

Для функционального тестирования программы был написан тестовый скрипт на языке Perl. Этот скрипт использует XS модуль для вызова функции hungarian\_solve() из динамической библиотеке libhungarian\_method.so, которая собиралась из исходного кода на языке С.

*Результаты тестирования*

В результате проведенного тестирования обнаружились следующие ошибки в программе:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 2 | 8 | 5 | 8 | 2 | 9 | | 7 | 4 | 2 | 2 | 9 | 2 | | 1 | 7 | 4 | 1 | 6 | 9 | | 7 | 5 | 4 | 5 | 1 | 5 | | 5 | 1 | 9 | 2 | 8 | 6 | | 1 | 1 | 3 | 7 | 8 | 5 | | F\_min = 8 | F\_min = 15 – Неверно |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 22222132 | 7 | 12 | 5 | 11 | 12 | | 11 | 14 | 8 | 12 | 5 | 8 | | 5 | 12 | 8 | 10 | 7 | 6 | | 2 | 1 | 3 | 12 | 8 | 3 | | 5 | 7 | 7 | 11 | 11 | 9 | | 10 | 1 | 11 | 7 | 7 | 1 | | F\_min = 24 | F\_min = 25 – Неверно |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 71 2 | 9 | 4 | 1 | 9 | 7 | | 9 | 8 | 6 | 3 | 3 | 1 | | 4 | 6 | 1 | 7 | 8 | 9 | | 5 | 3 | 2 | 8 | 7 | 4 | | 4 | 7 | 8 | 4 | 4 | 5 | | 9 | 2 | 4 | 2 | 3 | 2 | | F\_min = 13 | Зацикливание |

Код тестового скрипта представлен в приложениях 2 и 3.

***Выводы***

Программа работает некорректно на приведенных в отчете данных: имеются случаи зацикливания программы и предоставления неверного решения на тестовых данных.

## Вывод

Программу необходимо отправить на доработку.

# Приложение 1

Исходный код программы проверки входных данных на языке С

#include "hungarian\_method.h"

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

static void print\_matrix(int \*\*matrix,int n)

{

if (matrix == NULL)

return;

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

for (int j = 0; j < n; j++) {

if (j == 0)

printf("\t[%d", matrix[i][j]);

else

printf(", %d", matrix[i][j]);

}

printf("]\n");

}

}

struct hung\_data {

int \*\*matrix;

int \*row\_sign;

int \*column\_sign;

int n;

};

static void test(struct hung\_data \*input, struct hung\_data \*exp, int exp\_res)

{

static int test\_n = 0;

test\_n++;

printf("Test %d started...\n", test\_n);

if (input->matrix != NULL && input->n > 0) {

printf("\tsource matrix:\n");

print\_matrix(input->matrix, input->n);

}

int res = hungarian\_solve(input->matrix, input->n, input->row\_sign, input->column\_sign);

if (res != exp\_res) {

printf("Test FAILED (returned res %d)\n", res);

return;

}

if (exp->matrix != NULL) {

for (int i = 0; i < input->n; i++) {

for (int j = 0; j < input->n; j++) {

if (input->matrix[i][j] != exp->matrix[i][j]) {

printf("Test FAILED (matrix mismatch)\n");

printf("\tGot: \n");

print\_matrix(input->matrix, input->n);

printf("\tCorrect: \n");

print\_matrix(exp->matrix, exp->n);

return;

}

}

}

}

printf("Test PASSED\n");

}

int main()

{

int n = 2;

srand(time(NULL));

int \*\*matrix = (int \*\*)malloc(n \* sizeof(int\*));

int \*\*exp\_matrix = (int \*\*)malloc(n \* sizeof(int\*));

int \*row\_sign = (int \*)malloc(n \* sizeof(int));

int \*column\_sign = (int \*)malloc(n \* sizeof(int));

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

matrix[i] = (int \*)malloc(n \* sizeof(int));

exp\_matrix[i] = (int \*)malloc(n \* sizeof(int));

}

struct hung\_data source = {

.matrix = matrix,

.row\_sign = row\_sign,

.column\_sign = column\_sign,

.n = n

};

struct hung\_data exp = {

.matrix = exp\_matrix,

.row\_sign = NULL,

.column\_sign = NULL,

.n = n

};

// Test 1

matrix[0][0] = 2;

matrix[0][1] = 3;

matrix[1][0] = 4;

matrix[1][1] = 1;

exp\_matrix[0][0] = 1;

exp\_matrix[0][1] = 0;

exp\_matrix[1][0] = 0;

exp\_matrix[1][1] = 1;

test(&source, &exp, 0);

// Test 2

matrix[0][0] = 2;

matrix[0][1] = 3;

matrix[1][0] = 4;

matrix[1][1] = 1;

source.n = -1;

test(&source, &exp, -1);

// Test 3

matrix[0][0] = 2;

matrix[0][1] = 3;

matrix[1][0] = 4;

matrix[1][1] = 1;

source.n = 0;

test(&source, &exp, -1);

// Test 4

source.matrix = NULL;

source.n = 3;

test(&source, &exp, -1);

// Test 5

matrix[0][0] = -1;

matrix[0][1] = 2;

matrix[1][0] = 3;

matrix[1][1] = 10;

source.n = 2;

exp.matrix = NULL;

test(&source, &exp, -1);

// Test 6

matrix[0][0] = 1;

matrix[0][1] = -1;

matrix[1][0] = 3;

matrix[1][1] = 10;

test(&source, &exp, -1);

// Test 7

matrix[0][0] = 1;

matrix[0][1] = 2;

matrix[1][0] = -1;

matrix[1][1] = 10;

test(&source, &exp, -1);

// Test 8

matrix[0][0] = 1;

matrix[0][1] = 2;

matrix[1][0] = 3;

matrix[1][1] = -1;

test(&source, &exp, -1);

// Test 9

matrix[0][0] = INT\_MAX;

matrix[0][1] = INT\_MAX;

matrix[1][0] = INT\_MAX;

matrix[1][1] = INT\_MAX;

exp\_matrix[0][0] = 1;

exp\_matrix[0][1] = 0;

exp\_matrix[1][0] = 0;

exp\_matrix[1][1] = 1;

exp.matrix = exp\_matrix;

test(&source, &exp, 0);

// Test 10

matrix[0][0] = 0;

matrix[0][1] = 0;

matrix[1][0] = 0;

matrix[1][1] = 0;

test(&source, &exp, 0);

// Test 11

int m = 2;

int \*m\_ptr = &m;

source.matrix = &m\_ptr;

source.n = 1;

int exp\_m = 1;

int \*exp\_m\_ptr = &exp\_m;

exp.matrix = &exp\_m\_ptr;

exp.n = 1;

test(&source, &exp, 0);

// Test 12

matrix[0][0] = 5;

matrix[0][1] = 2;

matrix[1][0] = 1;

matrix[1][1] = 5;

source.matrix = matrix;

source.n = 2;

exp\_matrix[0][0] = 0;

exp\_matrix[0][1] = 1;

exp\_matrix[1][0] = 1;

exp\_matrix[1][1] = 0;

exp.matrix = exp\_matrix;

exp.n = 2;

test(&source, &exp, 0);

// Test 13

matrix[0][0] = 1;

matrix[0][1] = 3;

matrix[1][0] = 4;

matrix[1][1] = 1;

exp\_matrix[0][0] = 1;

exp\_matrix[0][1] = 0;

exp\_matrix[1][0] = 0;

exp\_matrix[1][1] = 1;

test(&source, &exp, 0);

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

free(matrix[i]);

free(exp\_matrix[i]);

}

free(matrix);

free(exp\_matrix);

free(row\_sign);

free(column\_sign);

return 0;

}

# Приложение 2

Исходный код тестового скрипта на этапе функционального тестирования

на языке Perl

#!/usr/bin/perl

use ExtUtils::testlib;

use List::Util qw(min max);

use Storable qw(dclone);

use HungarianTest;

my $n = shift() or 3;

my $limit\_int = shift() or 15;

my $test\_cnt = shift() or 20;

my $log\_file = "TEST.LOG";

open(my $LOG, ">", $log\_file) or die "unable to open $log\_file";

$BRUTE\_FORCE = 0;

$BRUTE\_FORCE = 1 if ($n < 8);

my @matrix\_variants;

if ($BRUTE\_FORCE == 1) {

print $LOG "Brute firce test is on\n";

build\_variants($n, \@matrix\_variants);

my $cur\_val = 1;

(($cur\_val = get\_fact($n, $cur\_val)) == scalar(@matrix\_variants)) or

die "incorrect matrix variants generation (need $cur\_val, got ". scalar(@matrix\_variants);

print "Generated marked matrix variants (dim = $n variants = $cur\_val)\n";

} else {

print $LOG "Rute force test is off \n";

}

my $err\_cnt = 0;

for $i (1 .. $test\_cnt) {

print "Test $i .." ;

random\_matrix\_test($n, $limit\_int) eq 0 ? print "PASS\n" : (print "ERR\n" and $err\_cnt++);

print $LOG "=============\n";

}

print "==========\n";

print "Result: \n\tPASSED ". ($test\_cnt - $err\_cnt) . " \n\tERR: $err\_cnt\n";

close($LOG);

sub print\_matrix(\@) {

my $m\_ref = shift() or die "need matrix ref";

foreach $arr (@{$m\_ref}) {

foreach $el (@{$arr}) {

print $LOG "$el ";

}

print $LOG "\n";

}

print $LOG "\n";

}

sub get\_fact() {

my $n = shift() or die "set n";

my $cur\_val = shift() or die "set cur val";

return $cur\_val if ($n == 1);

return get\_fact($n - 1, $cur\_val \* $n);

}

sub internal\_build\_variants($\@\@\@\@) {

my $n = shift() or die "specify dimension";

my $matrix\_arr\_ref = shift() or die "specify matrix\_arr ref";

my $marked\_matrix\_ref = shift() or die "specify marked matrix ref";

my $marked\_row\_ref = shift() or die "specify marked row ref";

my $marked\_column\_ref = shift() or die "specify marked column ref";

my @marked\_matrix= @{$marked\_matrix\_ref};

my @marked\_row = @{$marked\_row\_ref};

my @marked\_column = @{$marked\_column\_ref};

my $matrix\_was\_built = 1;

for $i (0 .. ($n - 1)) {

next if ($marked\_row[$i] == 1);

for $j (0 .. ($n - 1)) {

next if ($marked\_column[$j] == 1);

$marked\_row[$i] = 1;

$marked\_column[$j] = 1;

$marked\_matrix[$i][$j] = 1;

$matrix\_was\_built = 0;

internal\_build\_variants($n, $matrix\_arr\_ref, \@marked\_matrix, \@marked\_row,

\@marked\_column);

$marked\_row[$i] = 0;

$marked\_column[$j] = 0;

$marked\_matrix[$i][$j] = 0;

}

return if ($matrix\_was\_built == 0);

}

my @new\_matrix = @{ dclone(\@marked\_matrix) };

#print\_matrix(@new\_matrix);

push(@{$matrix\_arr\_ref}, \@new\_matrix);

}

sub build\_variants() {

my $n = shift() or die "set dimension";

my $matrix\_var\_ref = shift() or die "set matrix variants ref\n";

my @marked\_matrix, @marked\_row, @marked\_column;

for $i (0 .. ($n - 1)) {

$marked\_row[$i] = 0;

for $j (0 .. ($n - 1)) {

$marked\_column[$j] = 0;

$marked\_matrix[$i][$j] = 0;

}

};

internal\_build\_variants($n, @{$matrix\_var\_ref}, @marked\_matrix, @marked\_row, @marked\_column);

}

sub count\_f\_by\_selection(\@\@) {

my $matrix\_ref = shift() or die "specify matrix ref";

my $selection\_ref = shift() or die "specify selection matrix ref";

my $n = @{$matrix\_ref};

die "diff dimension sizes" if (@{$selection\_ref} != $n);

my $f = 0;

for $i (0 .. ($n - 1 )) {

for $j (0 .. ($n - 1 )) {

$f += $matrix\_ref->[$i][$j] if ($selection\_ref->[$i][$j] == 1);

}

}

return $f;

}

sub brute\_force\_opt(\@\@) {

my $matrix\_ref = shift() or die "specify matrix ref";

my $result\_matrix = shift() or die "specify matrix ref";

my $count\_f\_min = count\_f\_by\_selection(@{$matrix\_ref}, @{$result\_matrix});

my $min = $limit\_int \* $n;

my $min\_matrix;

for $m\_ref (@matrix\_variants) {

my $f = count\_f\_by\_selection(@{$matrix\_ref}, @{$m\_ref});

if ($f < $min) {

$min = $f;

$min\_matrix = $m\_ref;

}

}

if ($min != $count\_f\_min) {

print $LOG "ERR: diff min: must be $min, got $count\_f\_min\n";

print $LOG "Correct solution:\n";

print\_matrix(@{$min\_matrix});

return -1;

}

return 0;

}

sub hungarian\_method(\@) {

my $source\_matrix\_ref = shift() or die "incorrect parametrs list";

my %hash = (

'cost\_matrix\_n' => scalar(@{$source\_matrix\_ref}),

'cost\_matrix' => $source\_matrix\_ref

);

print $LOG "Source matrix\n";

print\_matrix(@{$source\_matrix\_ref});

HungarianTest::hungarian\_method(\%hash);

print $LOG "Result matrix\n";

print\_matrix(@{$hash{'cost\_matrix'}});

return $hash{'cost\_matrix'};

}

sub check\_result\_matrix(\@) {

my $matrix\_ref = shift() or die "incorrect parametrs list";

my $n = @{$matrix\_ref};

my @columns\_trace;

map{ push @columns\_trace, 0} @{$matrix\_ref};

for $i (0 .. ($n - 1)) {

my $has\_one = 0;

for $j (0 .. ($n - 1)) {

if (${$matrix\_ref}[$i][$j] == 1) {

if ($has\_one == 1) {

print $LOG "ERR: has two `1' in matrix row\n";

return 1;

}

$columns\_trace[$j]++;

if ($columns\_trace[$j] > 1) {

print $LOG "ERR: has two `1' in matrix column\n";

return 1;

}

$has\_one = 1;

}

}

}

return 0;

}

sub random\_matrix\_test() {

my $n = shift() or die "no matrix size is specidied";

my $limit = shift() or die "specify number highest limit";

my @matrix;

for $i (1 .. $n) {

my @arr;

for $j (1 .. $n) {

push @arr, (int(rand($limit - 1)) + 1);

}

push @matrix, \@arr;

}

my @copy\_matrix = @{ dclone(\@matrix) };

my $result\_matrix = hungarian\_method(@copy\_matrix);

my $res = check\_result\_matrix(@{$result\_matrix});

return -1 if ($res != 0);

if ($BRUTE\_FORCE == 1) {

return brute\_force\_opt(@matrix, @{$result\_matrix});

}

return 0;

}

# Приложение 3

Код XS модуля

#include "EXTERN.h"

#include "perl.h"

#include "XSUB.h"

#include "ppport.h"

#include "../../hungarian\_method.h"

#include <stdbool.h>

#include <stdlib.h>

#define hv\_fetch2(hv\_, name\_) hv\_fetch(hv\_, #name\_, sizeof(#name\_) - 1, 0)

#define hv\_fetch\_int(hv\_, name\_, has\_default\_, default\_...) ({ \

SV \*\*sv\_; \

int res\_; \

\

sv\_ = hv\_fetch2(hv\_, name\_); \

if (sv\_ == NULL) { \

if (has\_default\_) \

res\_ = (({res\_ = 0; res\_;}), ##default\_); \

else \

croak("`"#name\_"' is undefined"); \

} else if (!SvIOK(\*sv\_)) \

croak("`"#name\_"' must be an integer"); \

else \

res\_ = SvIVX(\*sv\_); \

\

res\_; \

})

static void print\_matrix(int \*\*matrix, int n)

{

int i, j;

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

if (j == 0)

printf("[%d", matrix[i][j]);

else

printf(", %d", matrix[i][j]);

}

printf("]\n");

}

}

static bool fetch\_array(AV \*av, int \*\*matrix, int n)

{

if ((av\_len(av) + 1) != n)

return false;

int i, j;

for (i = 0; i <= av\_len(av); i++) {

SV \*sv = \*av\_fetch(av, i, 0);

if (!SvROK(sv) ||

SvTYPE(SvRV(sv)) != SVt\_PVAV)

croak("fetch\_array: must be an array reference");

AV \*av\_internal = (AV \*)SvRV(sv);

if ((av\_len(av\_internal) + 1) != n)

return false;

for (j = 0; j <= av\_len(av\_internal); j++) {

SV \*sv = \*av\_fetch(av\_internal, j, 0);

int element = SvIV(sv);

matrix[i][j] = element;

}

}

return true;

}

static void set\_array(AV \*av, int \*\*matrix, int n)

{

if ((av\_len(av) + 1) != n)

return;

int i, j;

for (i = 0; i <= av\_len(av); i++) {

SV \*sv = \*av\_fetch(av, i, 0);

if (!SvROK(sv) ||

SvTYPE(SvRV(sv)) != SVt\_PVAV)

croak("fetch\_array: must be an array reference");

AV \*av\_internal = (AV \*)SvRV(sv);

if ((av\_len(av\_internal) + 1) != n)

return;

for (j = 0; j <= av\_len(av\_internal); j++) {

SV \*sv = \*av\_fetch(av\_internal, j, 0);

sv\_setiv(sv, matrix[i][j]);

}

}

}

MODULE = HungarianTest PACKAGE = HungarianTest

PROTOTYPES: ENABLE

void hungarian\_method(hv)

HV \* hv

PPCODE:

if (SvTYPE(hv) != SVt\_PVHV)

croak("must be a hash");

int n = hv\_fetch\_int(hv, cost\_matrix\_n, 0);

int \*\*cost\_matrix = (int \*\*)malloc(n \* sizeof(int \*));

int \*row\_sign = (int \*)malloc(n \* sizeof(int));

int \*column\_sign = (int \*)malloc(n \* sizeof(int));

int i;

for (i = 0; i < n; i++)

cost\_matrix[i] = (int \*)malloc(n \* sizeof(int));

SV \*\*sv = hv\_fetch2(hv, cost\_matrix);

if (!SvROK(\*sv) || SvTYPE(SvRV(\*sv)) != SVt\_PVAV)

croak("`cost\_matrix' must be an array reference");

AV \*av = (AV \*)SvRV(\*sv);

fetch\_array(av, cost\_matrix, n);

//print\_matrix(cost\_matrix, n);

// hungarian method

hungarian\_solve(cost\_matrix, n, row\_sign, column\_sign);

// pack result

set\_array(av, cost\_matrix, n);