**МЕТОДИЧЕСКОЕ ПОСОБИЕ**

**ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ ЗАДАНИЙ**

**УП.02 ПО ПРОФЕССИОНАЛЬНОМУ МОДУЛЮ**

**ПМ. 02 ОСУЩЕСТВЛЕНИЕ ИНТЕГРАЦИИ ПРОГРАММНЫХ МОДУЛЕЙ**

для специальности: 09.02.07 «Информационные системы и программирование»

Содержание

|  |  |  |
| --- | --- | --- |
| № пп | Наименование практических работ | Стр. |
|  | Введение | 2 |
| 1 | Практическая работа 1 | 4 |
| 2 | Практическая работа 2 | 8 |
| 3 | Практическая работа 3 | 10 |
| 4 | Практическая работа 4 | 18 |
| 5 | Практическая работа 5 | 20 |
| 6 | Практическая работа 6 | 23 |
| 7 | Практическая работа 7 | 25 |
| 8 | Практическая работа 8 | 29 |
| 9 | Практическая работа 9 | 32 |
| 10 | Практическая работа 10 | 41 |
| 11 | Практическая работа 11 | 43 |
|  | Рекомендуемая литература | 45 |

**Введение**

В соответствии с Государственным образовательным стандартом подготовки специалистов профессиональный модуль ПМ.02 «Осуществление интеграции профессиональных модулей» является обязательным для студентов СПО по направлению подготовки «Информационные системы и программирование». Поскольку курс имеет достаточно серьезную практическую составляющую, востребованность в методическом обеспечении для студентов очной формы обучения достаточно высока.

Цель создания данного пособия – оказание методической помощи студентам при выполнении практических работ по учебной практике УП.02 по ПМ.02 «Осуществление интеграции профессиональных модулей».

В пособии представлены примеры практических заданий по всем основным темам курса. Перед каждым заданием дан краткий перечень основных понятий по конкретной теме, которые студент должен изучить, приступая к решению задания. В конце каждой темы даны методические рекомендации по выполнению практических заданий.

**Практическая работа 1.**

Инструментальные средства разработки программного обеспечения

**Цель работы:** изучить современные инструментальные средства разработки программного обеспечения.

**Оборудование:** ПК, браузер.

**Ход работы:**

**1.** **Основные понятия темы, схему циклической модели проектирования ПО:**

**Инструментальные средства разработки ПО** — программное обеспечение ,предназначенное для использования в ходе проектирования, разработки и сопровождения программ, в отличие от прикладного обеспечения.

**Ассемблеры** — компьютерные программы, осуществляющие преобразование программы в форме исходного текста на языке ассемблера в машинные коды.

**Трансляторы** — программы или технические средства ,выполняющие трансляцию программы.

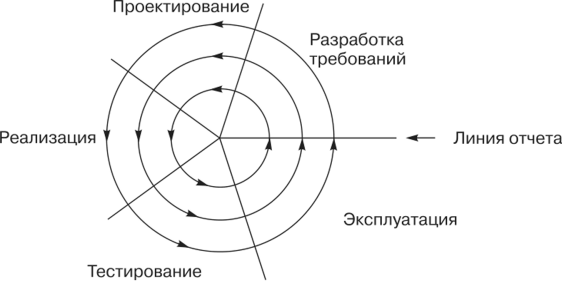
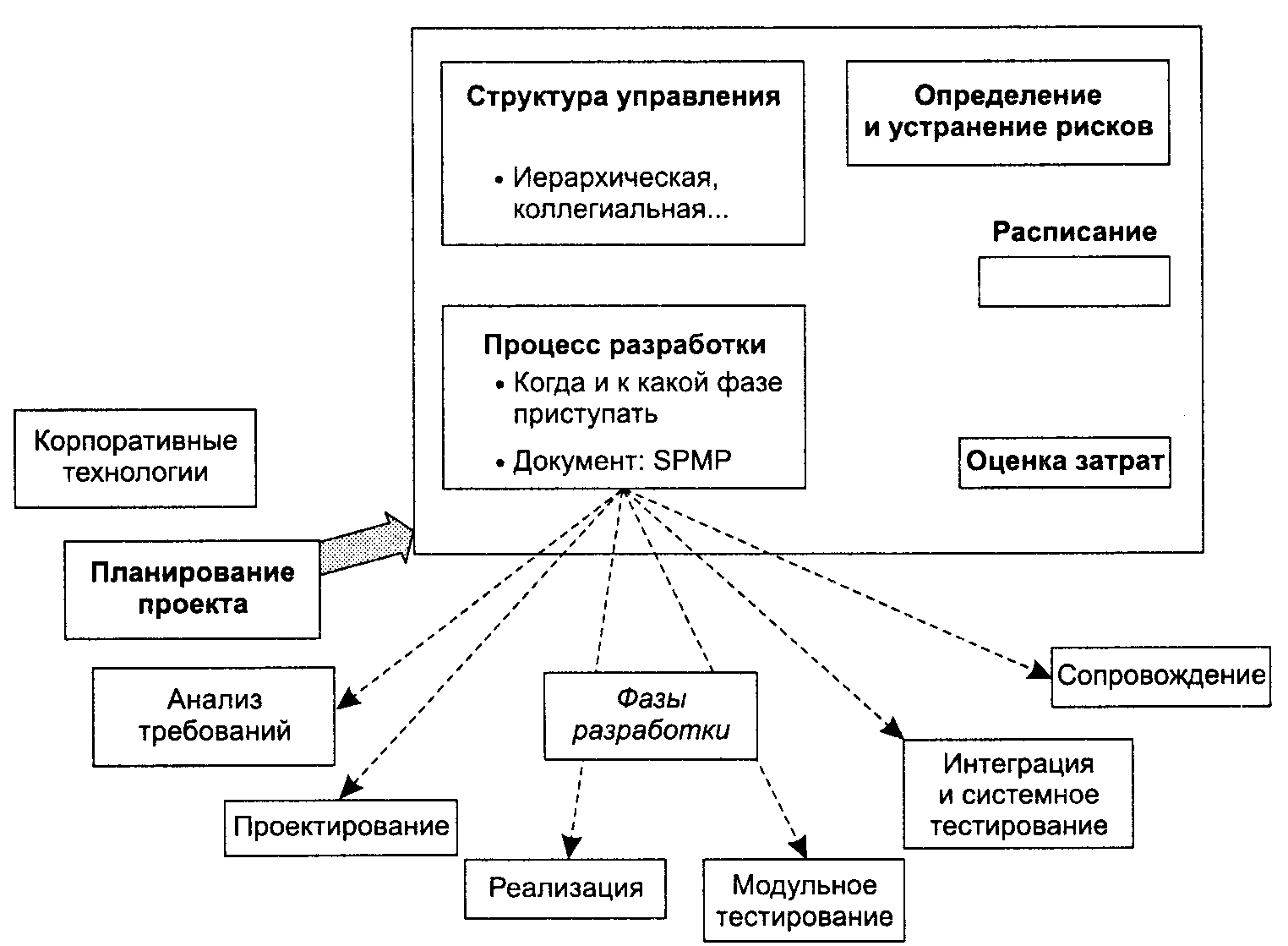
Циклическая модель проектирования ПО (схема 1)

Схема 1. Циклическая модель проектирования ПО

**2.** Схема процесса описания реализации программного кода с подробным описанием каждого этапа;

Схема 2. Процесс описания реализации программного кода.

**3.** Состав современных систем программирования;

* интегрированная среда разработчика программ (комплекс программных средств используемых программистами);
* транслятор языка программирования (перевод языка в машинный код);
* компоновщик или редактор связей (сборка файлов в единый исполняемый файл);
* библиотеки стандартных программ и функций;
* вспомогательные программные средства — программы отладки (исправление ошибок);
* справочная система.

**4.**  Функции современных компиляторов;

1. Лексический анализ;
2. Синтаксический анализ;
3. Генерация объектного кода.

**5.** Современные средства программирования с кратким описанием каждого (Python, C++. VSS, MS Visual Studio, Oracle, MS SQL, MySQL):

**Python** – универсальный современный язык программирования высокого уровня, к преимуществам которого относят высокую производительность программных решений и структурированный код.

**C++** - компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объекто-ориентированное программирование и др.

**VSS –** (Microsoft Visual SourceSafe) система управления версиями программного кода.

**MS Visual Studio -** интегрированная среда разработки программного обеспечения и ряд других инструментальных средств. Содержит редактор исходного кода с поддержкой технологии IntelliSense.

**Oracle —** объектно-реляционная система поддерживающая некоторые технологии, реализующие объектно-ориентированный подход.

**MS SQL –** система управления реляционными базами данных,разработанная **Microsoft.** Основной используемый язык запросов — **Transact-SQL.**

**MySQL** – свободная реляционная система управления базами данных.

**6.** Этапы проектирования приложений;

* Сбор требований к приложению;
* Предварительная оценка стоимости разработки;
* Проектирование прототипа;
* Составление технического задания;
* Разработка первого релиза продукта;
* Тестирование;
* Публикация;
* Техническая поддержка.

**7.** Нотации и средства для этапа проектирования;

**В состав проектирования ПО входят следующие действия:**

* Выбор метода и стратегии;
* Выбор представления внутренних данных;
* Разработка алгоритма;
* Документирование ПО;
* Тестирование;
* Выбор представления входных данных.

**Список средств для Проектирования ПО:**

* построение блок-схем (Pencil Project);
* UML-диаграммы(LibreOffice Draw);
* Разработка макетов(Lumzy);
* Разработка математических моделей.

**Практическая работа 2.**

Основные понятия и стандартизация требований к программному обеспечению

**Цель:**  Научится проводить анализ предметной области разрабатываемого ПО.

**Оборудование:** ПК

**Программное обеспечение**: Теоретические сведения к практической работе

**Теоретические сведения к практической работе**

**Требования к ПО**определяют, какие свойства и характеристики оно должно иметь для удовлетворения потребностей пользователей и других заинтересованных лиц.

Однако, в большинстве случаев пользователи могут перечислить только часть свойств, которые они хотели бы видеть и в не всегда понятной формулировке.

**Анализом предметной области** (или **бизнес-моделированием**, если речь идет о потребностях коммерческой организации) называют деятельность, направленную на:

выявление реальных потребностей людей и организаций (которые часто отличаются от непосредственно выражаемых пользователями желаний), выяснения смысла высказанных *требований* пользователей

1. выявление свойств желаемых результатов
2. определение набора задач, для их достижения
3. определение набора сущностей, необходимых при решении этих задач
4. определение области ответственности будущей программной системы.

После этого можно уже более точно сформулировать требования к ПО.

Задание

Опишите процесс работы кафедры вуза с точки зрения преподавателя.

Разработать программный модуль «Кафедра», содержащий сведения о сотрудниках кафедры (ФИО, должность, ученая степень, дисциплины, нагрузка, общественная работа, совместительство и др.). Модуль предназначен для использования сотрудниками отдела кадров и деканата.

Таблица 1. Программный модуль «Кафедра»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Данные  **ЧТО** | Функции  **КАК** | Дислокация  **ГДЕ** | Люди  **КТО** | Время  **КОГДА** | Мотивация  **ПОЧЕМУ** |
| Деканат | Учебные планы, программы курсов на факультете. | Административное обеспечение учебного процесса. | Калининград | Контроль работы по учебному плану. | В течение рабочего дня. | Оплата труда. |
| Заведующий кафедрой | Учебные программы и отчёт о работе преподавателей. | Организация проведения учебных занятий. | Кафедра | Утверждение графика работы. | В течение рабочего дня. | Оплата труда и карьерный рост. |
| Лаборант | Лабораторные работы. | Выполнение лабораторных анализов, измерений и испытаний. | Лаборатория. | Лабораторные работы учебной организации. | В течение рабочего дня. | Оплата труда и карьерный рост. |
| Преподаватель | Информация по количеству часов дисциплины. | Обучение, воспитание, организация учебного процесса. | Учебный сектор (Аудитория, кабинет). | Люди учебного процесса. | В течение рабочего дня. | Карьерный рост. |
| Разработчик | Описание структуры данных. | Программный код. | Серверная. | Реальные люди организации. | План рабочего дня. | Разработка программного модуля |
|  | **Данные** | **Функции** | **Дислокация** | **Люди** | **Время** | **Цель** |

**Практическая работа 3.**

Разработка и оформление технического задания

**Цель:**  Научится составлять и анализировать требования к программе и разрабатыватьтехническое задание на разработку программного средства.

**Оборудование:** ПК

**Теоретические сведения к практической работе;**

*Техническое задание* представляет собой документ, в котором сформулированы основные цели разработки, требования к программному продукту, определены сроки и этапы разработки и регламентирован процесс приемо-сдаточных испытаний. В разработке технического задания участвуют как представители заказчика, так и представители исполнителя. В основе этого документа лежат исходные требования заказчика, анализ передовых достижений техники, результаты выполнения научно-исследовательских работ, предпроектных исследований, научного прогнозирования и т. п.

**Порядок разработки технического задания;**

Разработка технического задания выполняется в следующей последовательности. Прежде всего, устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных. Затем определяют перечень результатов, их характеристики и способы представления.

Далее уточняют среду функционирования программного обеспечения: конкретную комплектацию и параметры технических средств, версию используемой операционной системы и, возможно, версии и параметры другого установленного программного обеспечения, с которым предстоит взаимодействовать будущему программному продукту.

В случаях, когда разрабатываемое программное обеспечение собирает и хранит некоторую информацию или включается в управление каким-либо техническим процессом, необходимо также четко регламентировать действия программы в случае сбоев оборудования и энергоснабжения.

**Оценка качества процессов создания программного обеспечения.**

Переход от штучной разработки Программных продуктов к промышленному программированию обусловил повышение требований к качеству создаваемого ПО. В настоящее время существуют несколько стандартов:

* Международные стандарты серии ISO 9000 (ISO 9000 – ISO 9004)
* CMM – Capability Maturity Model – модель зрелости процессов создания ПО.
* Процесс сертификации программ на базе информации об их использовании.

**Серия стандартов ISO 9000**

Одной из важнейшей проблем обеспечения качества ПС является формализация характеристик качества и методология их оценки. Основой регламентирования показателей качества ПС раннее являлся Международный стандарт ISO 9126: 1991 «Информационные технология. Оценка программного продукта. Характеристики качества и руководства по их применению».

В России в области обеспечения жизненного цикла и качества сложных комплексов программ в основном применяется устаревших ГОСТов, которые отстают от Мирового уровня на 5-10 лет.

Первая часть стандарта — ISO 9126-1 — распределяет атрибуты качества ПС по 6 характеристикам используемых в остальных частях стандарта. К ним применимы разные категории метрик:

* Категорийным, или описательным, метрикам наиболее адекватны функциональные возможности ПС.
* Количественные метрики применимы для измерения надежности и эффективности сложных комплексов программ.
* Качественные метрики в наибольшей степени соответствуют практически, сопровождаемости и мобильности ПС.

**Выбор показателей качества**

Исходными данными и высшим приоритетом при выборе показателей качества в большинстве случаев являются назначения, функции и функциональная пригодность, соответствующего ПС.

Процессы выбора и установления метрик и шкал для описания характеристик качества ПС можно разделить на 2 этапа:

* Выбор и обоснование набора исходных данных отражающих общие возможности и этапы жизненного цикла ПС и его потребителей.
* Выбор, установление и утверждение конкретных метрик и шкал измерения характеристик и атрибутов качества проектов для последующих оценок.

Оценка качества

Методология и стандартизация оценки характеристик качества готовых ПС и их компонентов на различных этапах жизненного цикла посвящен Международный стандарт ISO 14598.

Рекомендуется следующая общая съема процессов оценки:

* Установка исходных требований для оценки — определение целей испытаний.
* Селекция метрик качества, установление рейтингов и уровней приоритета метрик субхарактеристик и атрибутов.
* Планирование и проектирование процессов оценки характеристик и атрибутов качества в жизненном цикле ПС
* Выполнение измерений для оценки, сравнение результатов с критериями и требованиями.

1. **Функциональная пригодность** — наиболее неопределенная и объективно трудно оцениваемая субхарактеристика ПС.
2. **Оценка корректности ПС** — состоит в формальном определении степени соответствия комплекса реализованных программ исходным требованиям контракта.
3. **Оценка способности к взаимодействию** — состоит в определении качества совместной работы компонентов ПС и баз данных.
4. **Оценка защищенности ПС** — включает определение полноты использования доступных методов и средств защиты ПС от потенциальных угроз.
5. **Оценка надежности** — измерение количественных метрик атрибутов субхарактеристик в использовании: завершенности, устойчивости к дефектам, восстанавливаемости, доступности.

**Потребность в ресурсах памяти и производительности** компьютеров в процессе решения задач значительно изменяется в зависимости от состава и объема данных.

**Оценка практичности** программных средств проводится экспертами и включает определение понятности, простоты использования, изучаемости и привлекательности ПС.

**Сопровождаемость** можно оценивать полнотой и достоверностью документации о состояниях ПС и его компонентов.

**Оценка мобильности** — качественное определение экспертами адаптируемости простоты установки, совместимости и замещаемости программ, выражаемое в баллах.

**Система управления качеством**

Определения характеристик и субхаракеристик качества:

**Функциональные возможности** — способность ПС обеспечивать решение задач.

**Функциональная пригодность** — набор и описание субхарактеристики и ее атрибутов, определяющие назначение.

**Правильность (корректность)** — способность ПС обеспечивать правильные или приемлемые для пользователя результаты.

**Способность к взаимодействию** — свойство ПС и их компонентов взаимодействовать с одной или большим числом компонентов внутренней и внешней среды

**Защищенность** — способность компонентов ПС защищать программы и информацию.

**Надежность** — обеспечение комплексом программ достаточно низкой вероятности отказа в процессе функционирования ПС в реальном времени.

**Эффективность** — свойства ПС, обеспечивающие требуемую производительность решения функциональных задач.

**Практичность (применимость)** — свойства ПС, обуславливающие сложность его понимания, изучения и использования.

**Сопровождаемость** — приспособленность ПС к модификации и изменению конфигураций и функций.

**Мобильность** — подготовленность ПС к переносу из одной аппаратно-операционной среды в другую.

Пример

**Титульный лист**

**1. Введение**

Настоящее техническое задание распространяется на разработку программного модуля «Кафедра», содержащий сведения о сотрудниках кафедры (ФИО, должность, ученая степень, дисциплины, нагрузка, общественная работа, совместительство и др.).

**2. Основание для разработки**

* + Программа разрабатывается на основе сведений о сотрудниках кафедры.
  + Наименование работы:

« Программный модуль «Кафедра» ».

* + Исполнитель: компания BestSoft.
  + Соисполнители: нет.

**З. Назначение**

Модуль предназначен для использования сотрудниками отдела кадров и деканата.

**4. Требования к программе или программному изделию**

4.1. Требования к функциональным характеристикам

4. l. l. Программа должна обеспечивать возможность выполнения следующих функций:

* Хранение вносимой информации в памяти и/или в базе данных;
* Вывод текстовой информации;

• Визуальное предоставление информации о сотрудниках кафедры;

* Распределение определенных файлов по соответствующим папкам/архивам (Например: Список со сведениями о преподавателях находится в C:\Users\User\Documents\ProgrModulKafedra\Учителя).

4.1.2. Исходные данные вводятся с устройства ввода или загружаются уполномоченным пользователем.

4.1.3. Организация входных и выходных данных

Входные данные поступают с клавиатуры или из текстовых и графических документов.

Выходные данные отображаются на экране и при необходимости выводятся на печать.

4.2. Требования к надежности

* Контроль вводимой информации.
* Регистрация и последующая авторизация уполномоченного пользователя.
* Блокировка некорректных действий пользователя при работе с системой.

4.3. Требования к составу и параметрам технических средств. Система должна работать на 1ВМ-совместимых персональных компьютерах.

Минимальная конфигурация:

* + - тип процессора. Pentium и выше;
    - объем оперативного запоминающего устройства 32 Мб и более;
    - объем свободного места на жестком диске 64 Мб.

Рекомендуемая конфигурация:

* + - тип процессора. Pentium П 400;
    - объем оперативного запоминающего устройства 128 Мб;
    - объем свободного места на жестком диске 60 Мб.

4.4. Требования к программной совместимости.

Программа должна работать под управлением семейства операционных систем Win 64 (Windows vista/7/8/10/ и т. п.).

**5. Требования к программной документации**

Основными документами, регламентирующими разработку будущих программ, должны быть документы Единой Системы Программной Документации (ЕСПД): руководство пользователя, руководство администратора, описание применения.

**6. Технико-экономические показатели**

Эффективность системы определяется удобством использования системы для контроля и управления основными параметрами теплообеспечения помещений Московского института, а также экономической выгодой, полученной от внедрения аппаратно-программного комплекса.

**7. Порядок контроля и приемки**

После передачи Исполнителем отдельного функционального модуля программы Заказчику последний имеет право тестировать модуль в течение 7 дней. После тестирования Заказчик дол-

жен принять работу по данному этапу или в письменном виде изложить причину отказа принятия. В случае обоснованного отказа Исполнитель обязуется доработать модуль.

**8. Календарный план работ**

|  |  |  |  |
| --- | --- | --- | --- |
| Этап | Название этапа | Сроки этапа | Чем заканчивается этап |
| 1 | Изучение предметной области. Проектирование системы. Разработка предложений по реализации системы. | 05.10.2020 — 07.10.2020 | Предложения по работе системы. Акт сдачи-приемки. |
| 2 | Разработка программного модуля по сбору и анализу информации со счетчиков и устройств управления. Внедрение системы для отделения кафедры. | 8.10.2020 — 15.10.2020 | Программный комплекс, решающий поставленные задачи для пилотного отделения кафедры сдачи-приемки |
| 3 | Тестирование и отладка модуля. Внедрение системы во все отделения кафедры. | 16.10.2020 — 20.10.2020 | Готовая система контроля данными кафедры,  установленная в диспетчерском пункте. Программная  документация.  Акт сдачи-приемки работ |

Руководитель работ Иванов И.И.

**Практическая работа 4.**

Решение транспортной задачи в пакете ms Excel (LibreOffice)

**Транспортная задача** – это задача о минимизации транспортных расходов, связанных с обеспечением пунктов потребления определенным количеством однородной продукции, производимой в нескольких пунктах производства. В общем виде задача может быть сформулирована следующим образом. Однородный продукт, сосредоточенный в ***m*** пунктах производства, необходимо распределить между ***n*** пунктами потребления. Стоимость перевозки единицы продукции известна для всех маршрутов. Необходимо составить такой план перевозок, при котором запросы всех пунктов потребления были бы удовлетворены за счет имеющихся продуктов в пунктах производства и общие транспортные расходы по доставке продуктов были бы минимальными.

Примем следующие обозначения: i – номер пункта производства, j –номер пункта потребления, ***ai***– количество продукта, имеющееся в i-ом пункте производства, ***bj*** – количество продукта, необходимое для j-го пункта потребления, ***cij***– стоимость перевозки единицы продукта из i-го пункта производства в j-й пункт потребления, ***xij***– количество груза, планируемого к перевозке из i-го пункта производства в j-й пункт потребления,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 33 | 10 | 22 | 32 |
| 28 | 40 | 39 | 34 | 33 |
| 23 | 13 | 18 | 16 | 32 |
| 11 | 18 | 29 | 33 | 46 |
| 33 | 37 | 14 | 10 |  |

Решение представлено на Рис.2.

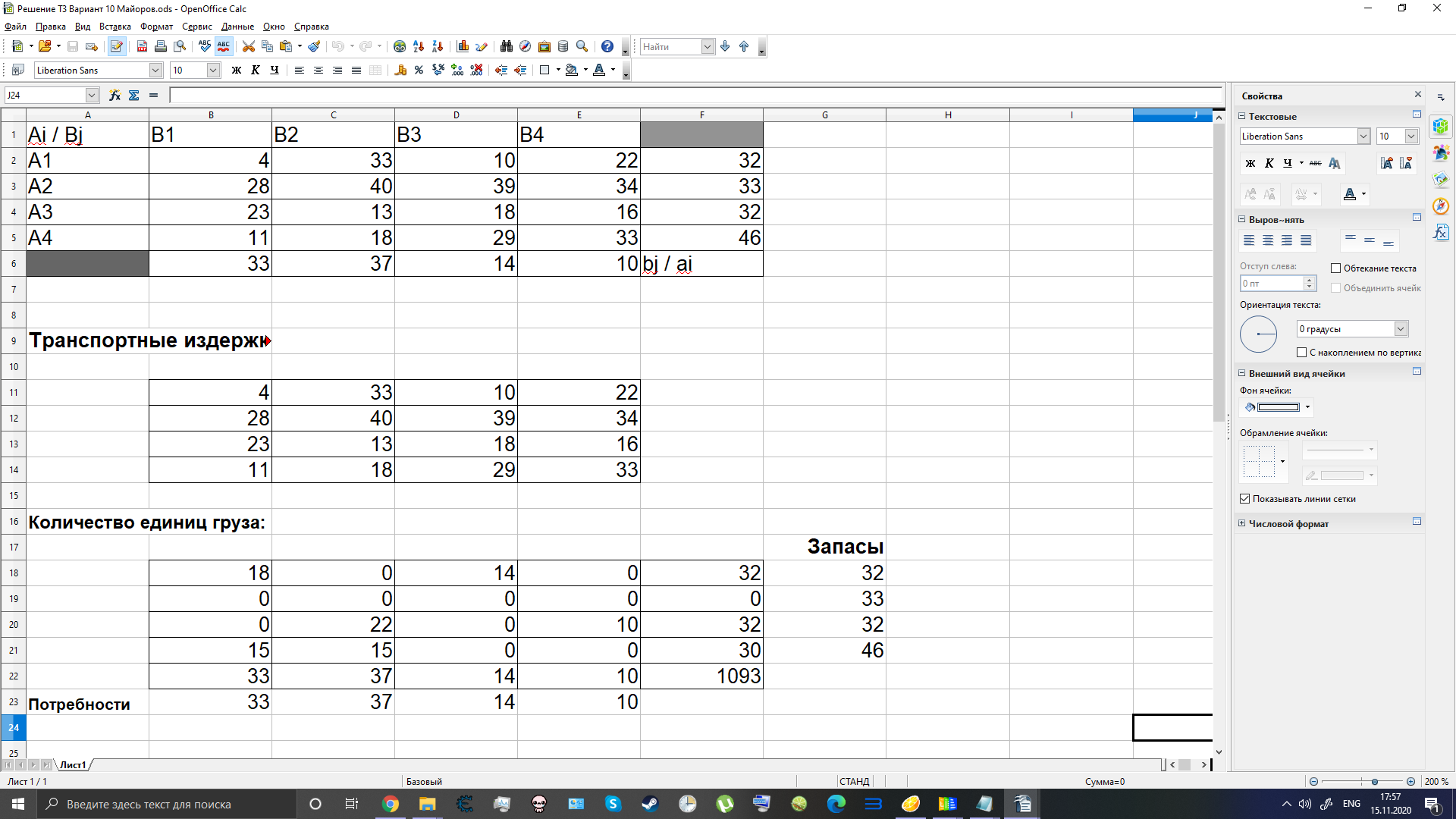


Рис.2 (Готовое решение задачи)

**Практическая работа 5.**

Функциональная модель бизнес-процесса и CASE – средства

Цель: Приобрести навыки построения функциональной модели бизнес-процесса в предметной области, используя CASE-средства и методологию IDEF0.

***Бизнес-процесс*** (БП) – упорядоченная во времени и пространстве совокупность взаимосвязанных работ, направленных на получение определенного результата (продукции, работ или услуг).

***Моделирование бизнес-процессов*** – это описание бизнес-процессов предприятия, позволяющее руководителю знать, как работают рядовые сотрудники, а рядовым сотрудникам – как работают их коллеги и на какой результат направлена вся их деятельность.

***Модель бизнес-процесса*** – представление бизнес-процесса на специализированном языке (с помощью специализированной нотации – текстовой, табличной, графической).

**Методологии:**

**IDEF0** (BusinessProcess, функциональная модель) методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. В ней система представляется как совокупность взаимодействующих работ или функций;

**ARIS** (Architecture of Integrated Information Systems) – методология и тиражируемый программный продукт для построения организационной и функциональной структур, структур данных и процессов;

**DFD** (Data Flow, поток данных) – методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.

**Системы моделирования бизнес-процессов организации:**

* **AllFusionProcessModelerr7** (Computer Associates);
* **RationalRose** (Rational Software);
* **OracleDesigner** (Oracle);
* **ARIS** (IDSScheerAG);
* **PowerDesigner** (Sybase).

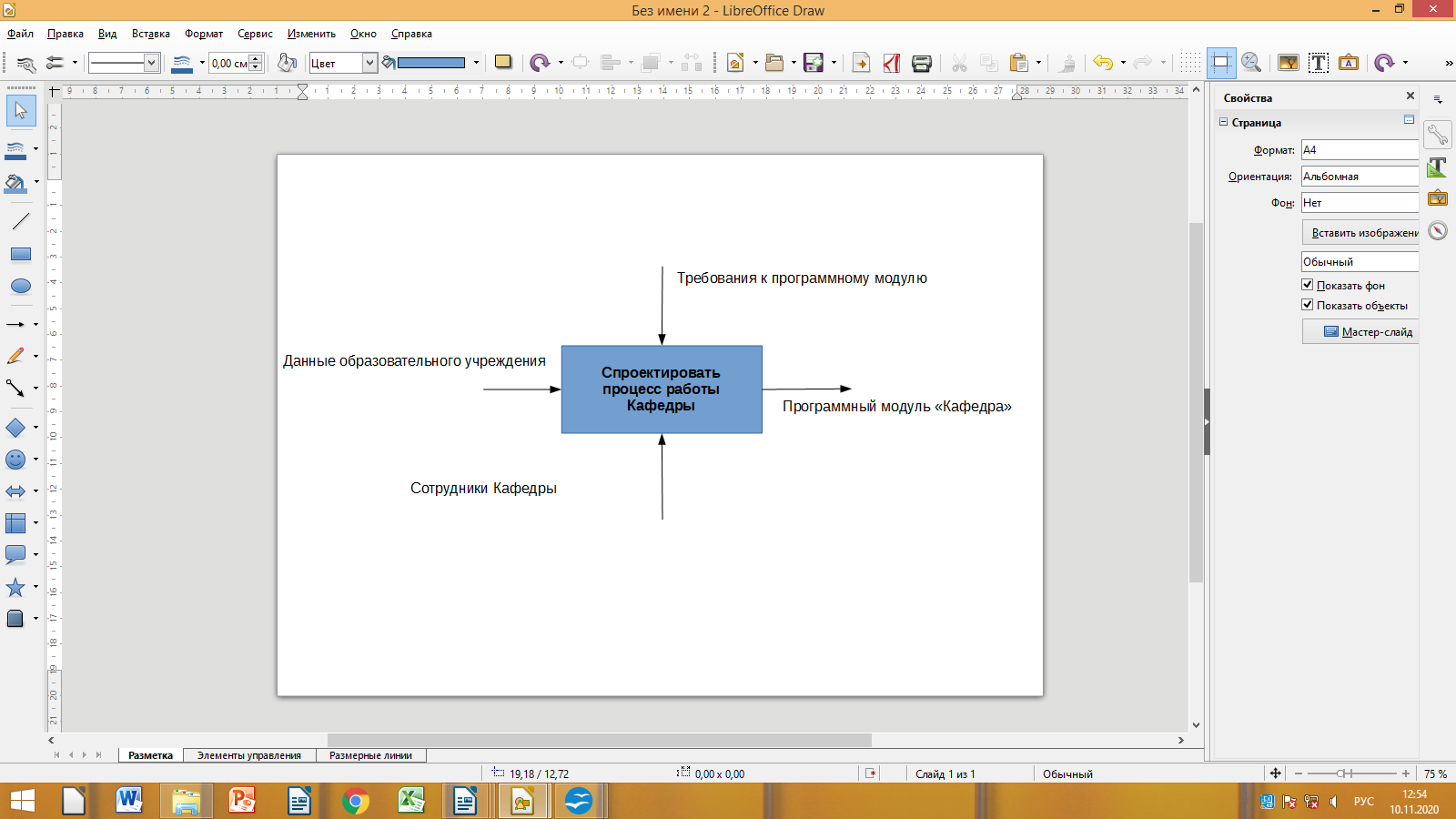
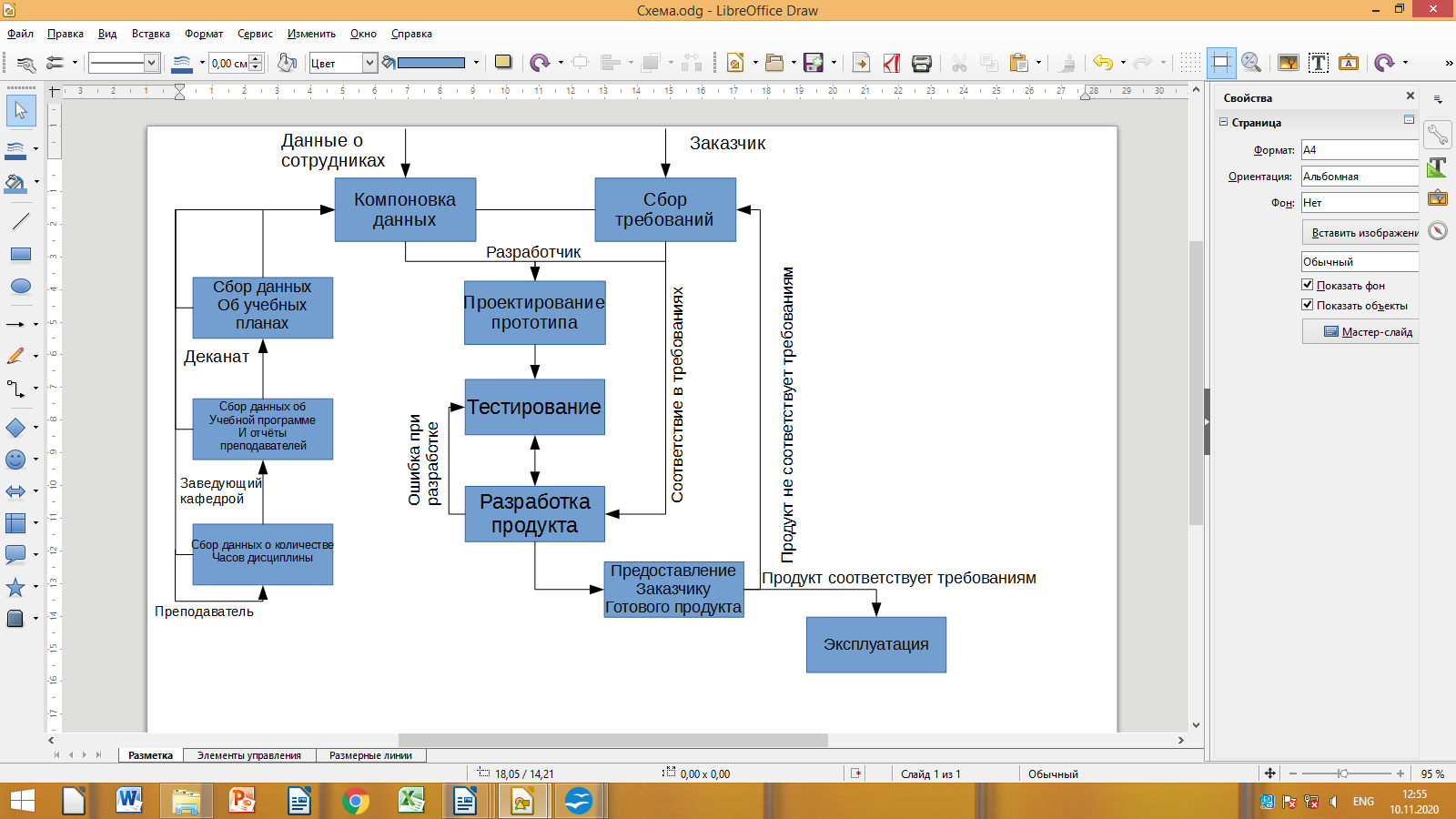


Рис.3 (Функциональная модель IDEF0)

**Моделирование бизнес-процесса с использованием методологии IDEF0**

**Методология семейства IDEFO:**

Рис.4 (Функциональная модель бизнес процесса второго уровня)

**IDEF0 методология** функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. В ней система представляется как совокупность взаимодействующих работ или функций.

Практическая работа 6.

# Задача поиска кратчайшего пути.

Цель работы: решение задачи нахождения кратчайшего пути в графе средствами LibreOffice Calc.

**Теоретическая часть:** Для решения задач и в процедуре LibreOffice Calc «Поиск решения», представим ее как транспортную задачу с промежуточными пунктами. Будем считать, что транспортные расходы при перевозке одной единицы груза равны (в условных единицах) расстояниям между вершинами.

# Порядок выполнения работы

1. В соответствии с вариантом задания, определенным преподавателем, по графу составить матрицу транспортных расходов и найти ее решение.
2. Оформить отчет о выполнении задания с приведением условия задачи, формул для расчета, результатов решения и заключения.

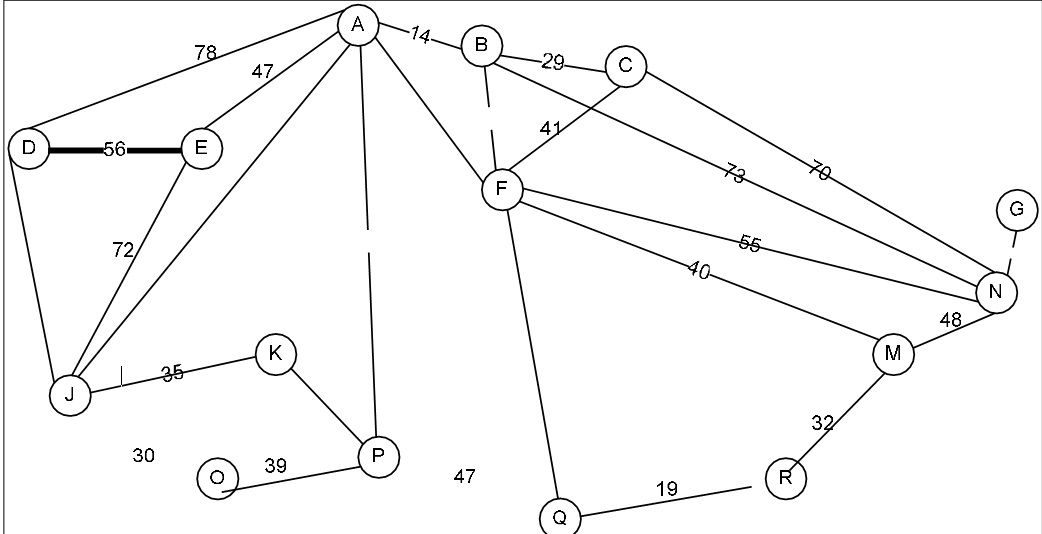
**Ход работы:**

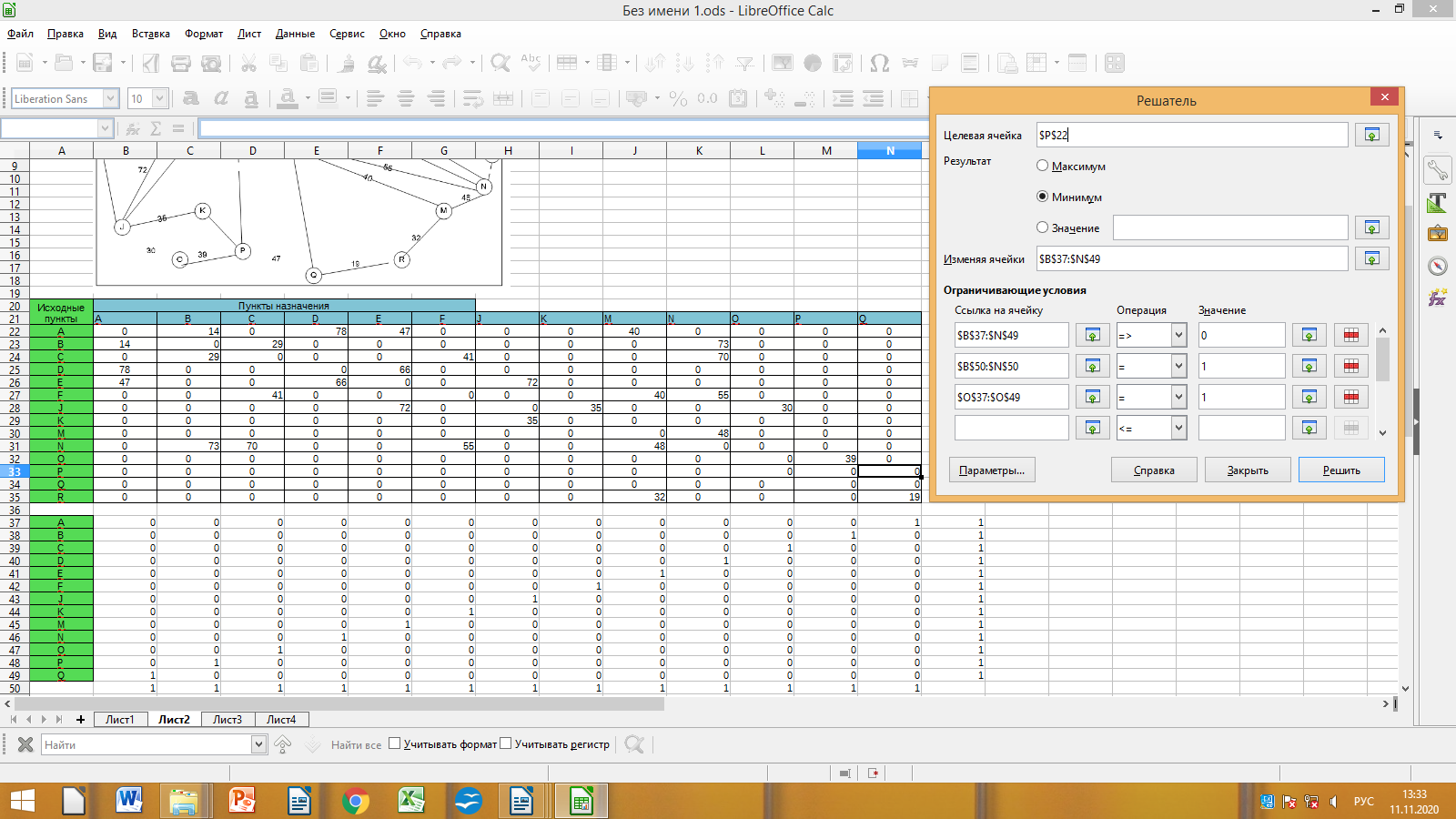
Варианты заданий

На Рисунке 5 показана транспортная сеть, соединяющая 16 населенных пунктов, и расстояния между ними. Найдите кратчайшие маршруты между следующими населенными пунктами.

|  |  |
| --- | --- |
| **Вариант** | **Маршрут** |
| 4 | R - E |

Схема .3

Требуется определить такую последовательность вершин, по которым должна перемещаться единица груза, отправленная из вершины R, при которой стоимость транспортных расходов будет минимальна и груз попадет в вершину E.

Рис.5

В этом варианте задания нам выдали условие найти длину пути от пункта R до пункта E(Схема.3). Заполнили таблицу значениями (Рис.5), приведенными на схеме задания.

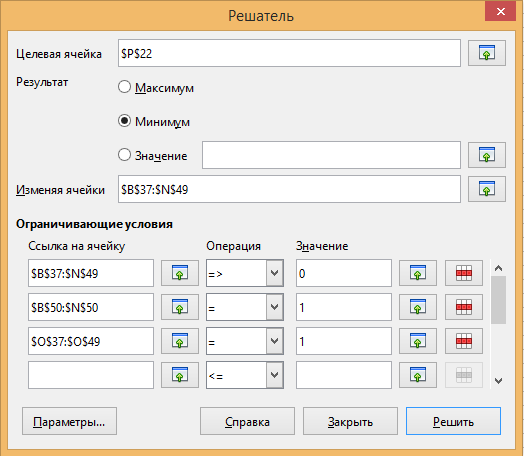


Рис.6

**Ввели в «Решатель» ограничения (Рис.6):**

В целевой ячейке вы вставили адрес ячейки, в которую будет записан результат длины пути.

Результат должен быть минимальным.

В Ограничивающих условиях мы ввели адреса таблицы и n-ное количество ячеек. Первое ограничение: Значения таблицы не должны быть отрицательными; Второе и Третье ограничения должны быть равны единице.

В результате мы получили значение минимального значения задачи поиска кратчайшего пути.

**Практическая работа 7.**

Структурная схема разрабатываемого ПО

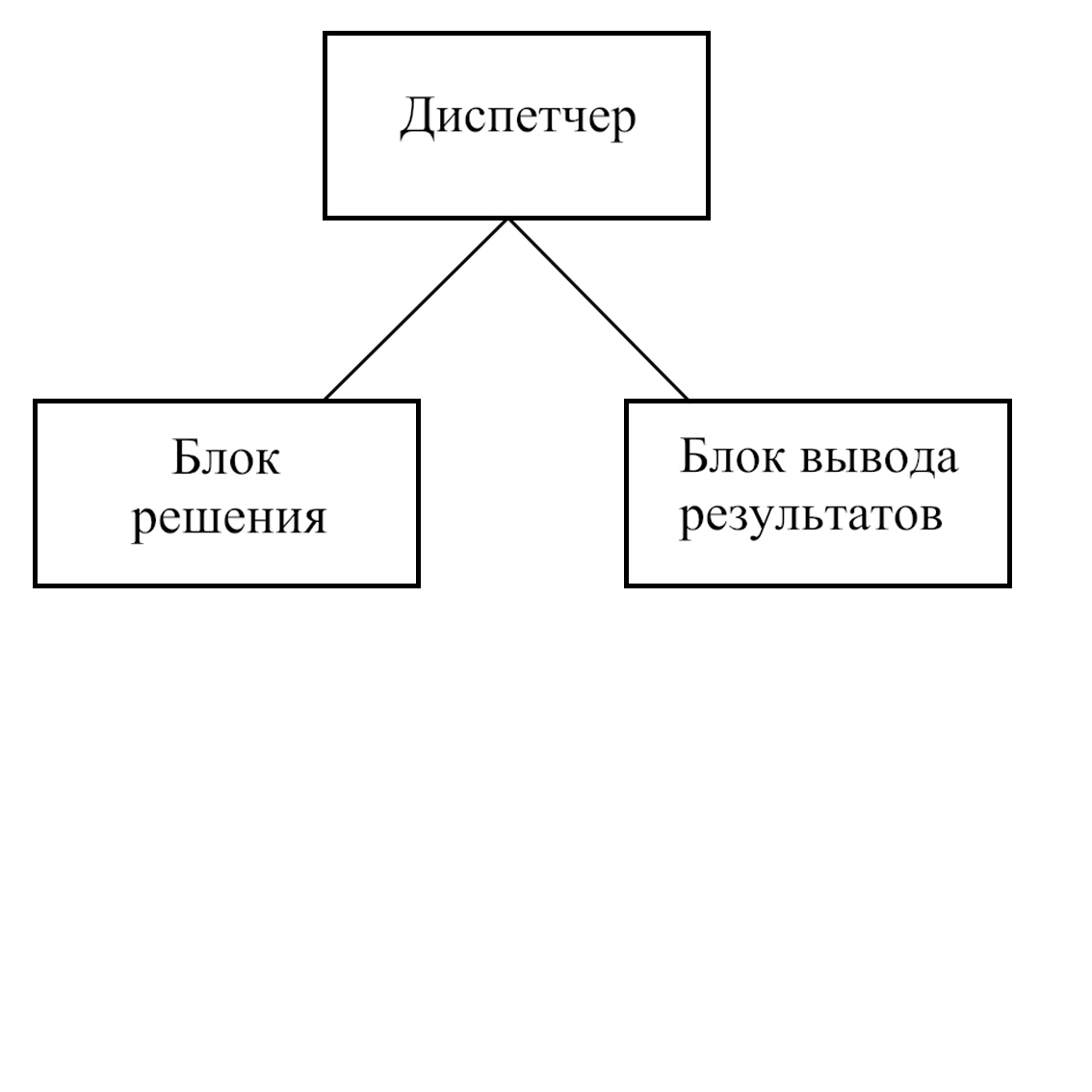
Структурной называют схему, отображающую состав и взаимодействие по управлению частей разрабатываемого ПО.

Разработку структурной схемы **программы** обычно выполняют методом пошаговой детализации.

Структурные схемы **пакетов программ** разрабатывают для каждой программы пакета по отдельности.

Компонентами структурной схемы **программной системы** или **программного комплекса** могут служить программы, подсистемы, базы данных, библиотеки ресурсов и т.п.

Пример структурной схемы программного комплекса для решения математических задач изображен на рис.7

Рис.7.

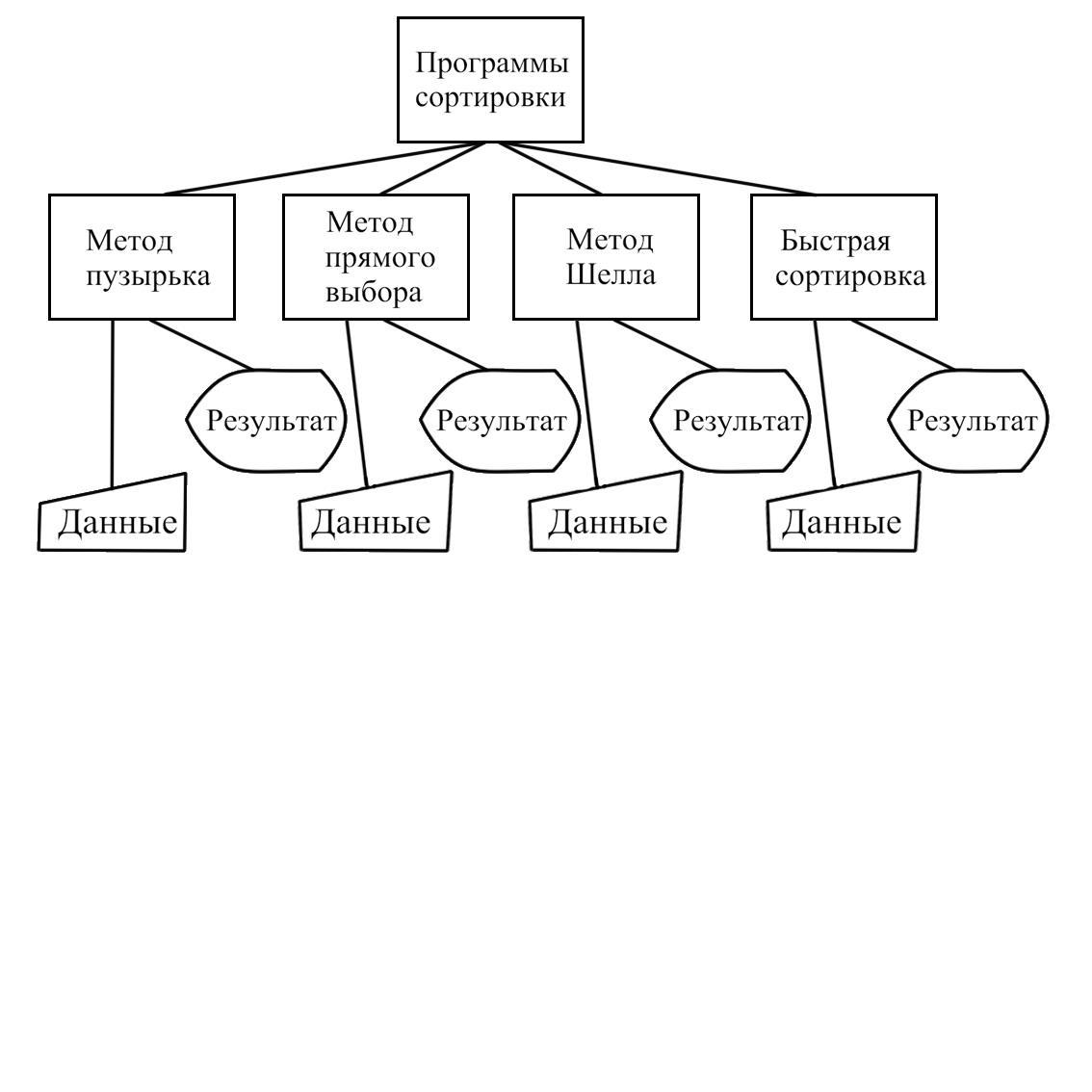
**Функциональная схема**

Функциональная схема (ГОСТ 19.701- 90) - это схема взаимодействия компонентов программного обеспечения с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и устройств. Для изображения функциональных схем используют специальные обозначения, установленные стандартом (табл.2).

Таблица 2. Обозначения, установленные стандартом



Функциональные схемы более информативны, чем структурные. На схеме 3 приведена функциональная схема программного комплекса, реализующего различные методы сортировки массивов

Схема.3

**Пример 1.**

Пусть требуется определить наибольшее значение в некотором наборе данных и вывести эти данные, поделенные на наибольшее значение. Скажем, если данные представляют собой последовательность чисел:

5.0, - 3.24, 10.0, -1.25, 8.33.

Код примера на языке Python (Рис.8)

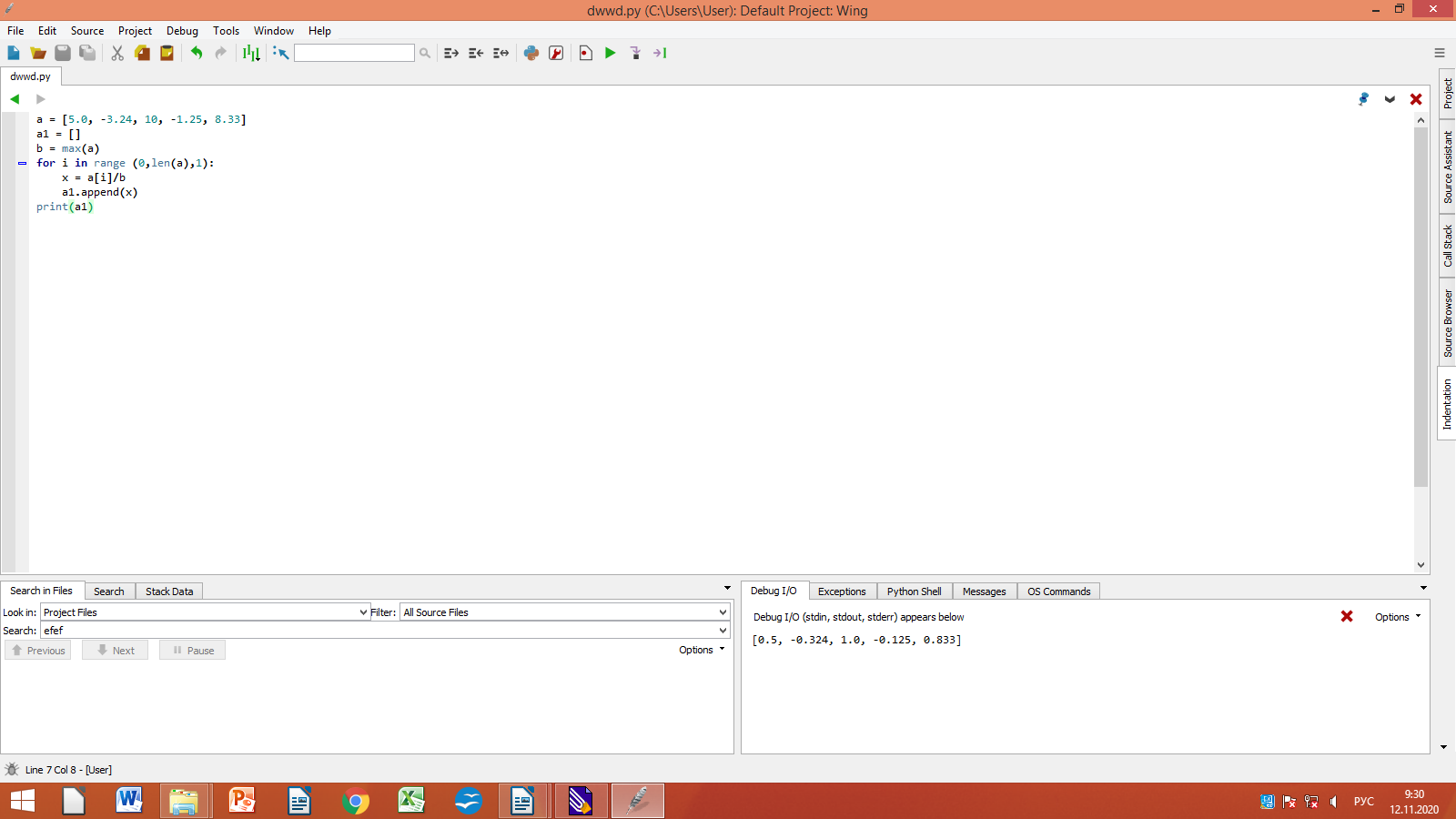
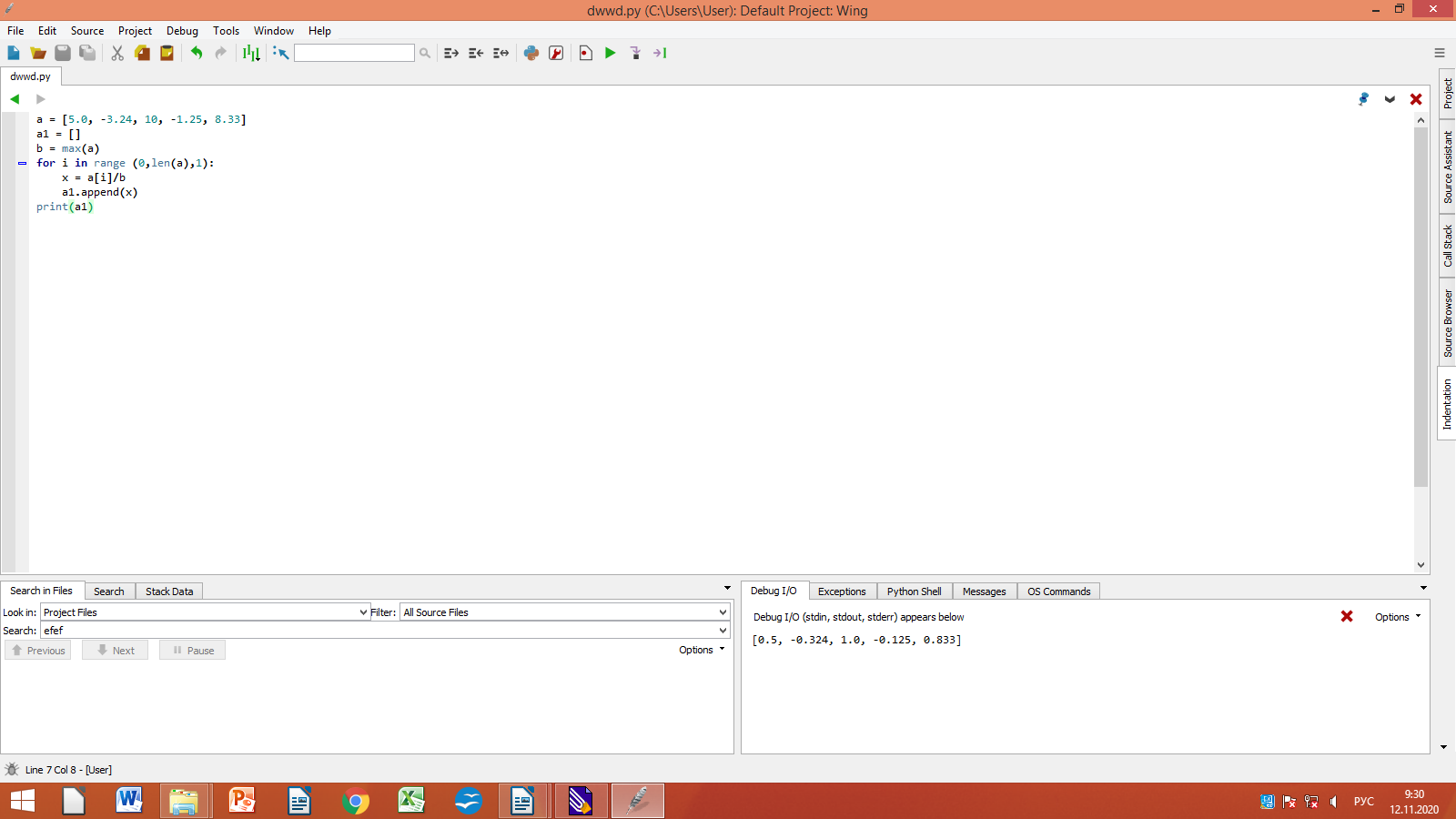


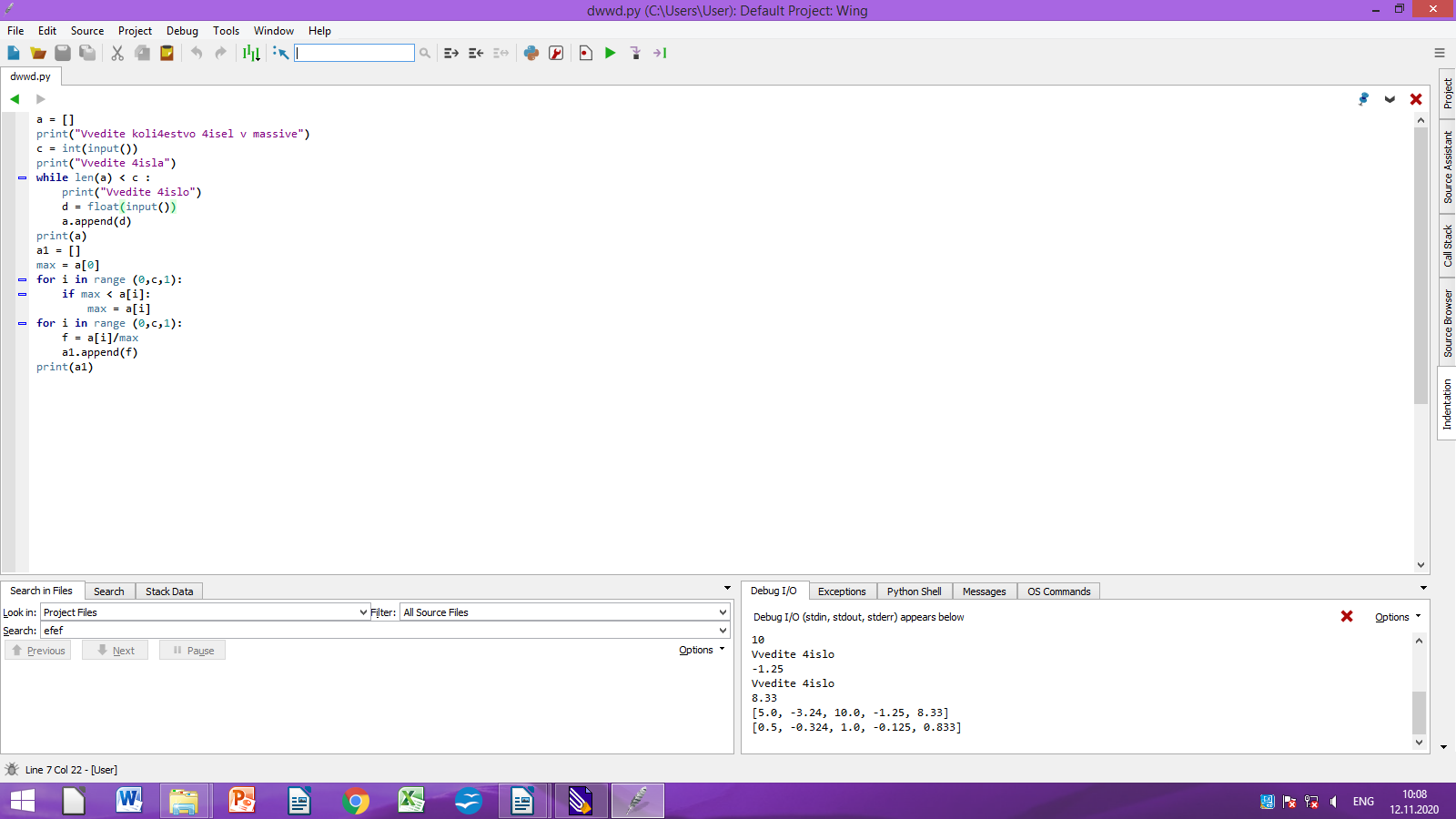
Рис.8.

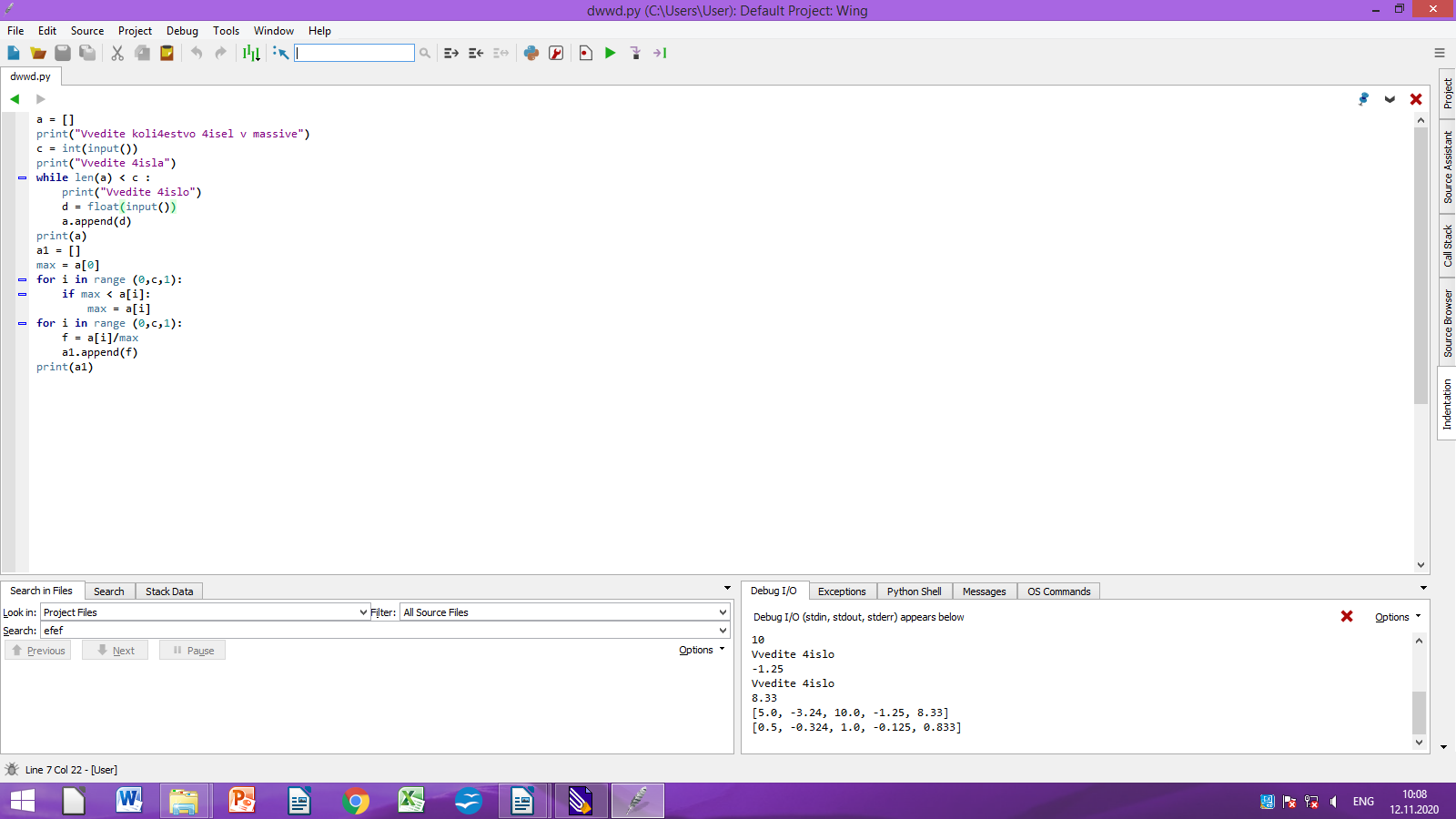
То вывод должен выглядеть следующим образом (Рис.9)

Рис.9.

0.5, -0.324, 1.0, -0.125, 0.833.

**Детализация:**





**Практическая работа 8**

Симплекс-метод

Цель работы:Расширить и углубить практические знания и навыки в области применения стандартных прикладных программ при постановке и решении задач линейного программирования. Использование симплекс-метода для задач линейного программирования в таблицах Excel.

## Алгоритм симплекс-метода

Последовательность действий можно описать следующим образом:

* путем преобразований система ограничений приводится к необходимой, так называемой базисной, форме;
* находится так называемое опорное решение, служащее «точкой отсчета»;
* последовательно перебираются вершины симплекса. Если в данной точке значение критерия больше (или меньше) предыдущего, то процесс продолжается. Когда значение критерия уже нельзя улучшить, значит, решение найдено.

То есть, смысл симплексного метода следующий: все линейные неравенства, которым в многомерном пространстве соответствуют полуплоскости, ограничивают некий симплекс. При этом уравнению, описывающему оптимизируемый критерий, соответствует гиперплоскость.

Теперь нужно просто найти ту вершину симплекса, одновременно принадлежащую этой гиперплоскости, координаты которой максимизируют (минимизируют) критерий. Следовательно, выбирается базисная вершина и по ней мы передвигаемся от одной вершины к другой, пока не найдем точку оптимума.

## Практический пример применения симплексного метода

Решим симплексным методом задачу. Максимизируем функцию

***L=X+Z→max***

при ограничениях

***{Y−X+Z=1,X−2Z+T=2.***

У нас есть четыре переменные – ***X,Y,Z,T*** – причем критерий зависит лишь от двух переменных. Примем ***Т*** и ***Y*** за базисные переменные и выразим их через остальные две свободные переменные. Получим:

***L=X+Z→max,***

***{Y=1+X−Z,T=2−X+2Z.***

При ***X*** и ***Z*** равных нулю, базисные переменные равны ***Y=1,T=2***. Значение критерия ***L=0***. Значит, точка ***(1,0,0,2)*** является базисным решением. Начнем перебор вершин симплекса. Увеличить критерий можно увеличив ***Z*** до единицы. Тогда при ***Z=1,X=0*** базисные переменные примут значения ***Y=0,T=4***. Новое допустимое решение – это точка ***(0,0,1,4)***, критерий равен ***L=1***.

Теперь выразим ***Z*** и ***T*** через ***Y*** и ***X***:

***L=1−Y+2X→max,***

***{Z=1−Y+Z,T=4−2Y+X.***

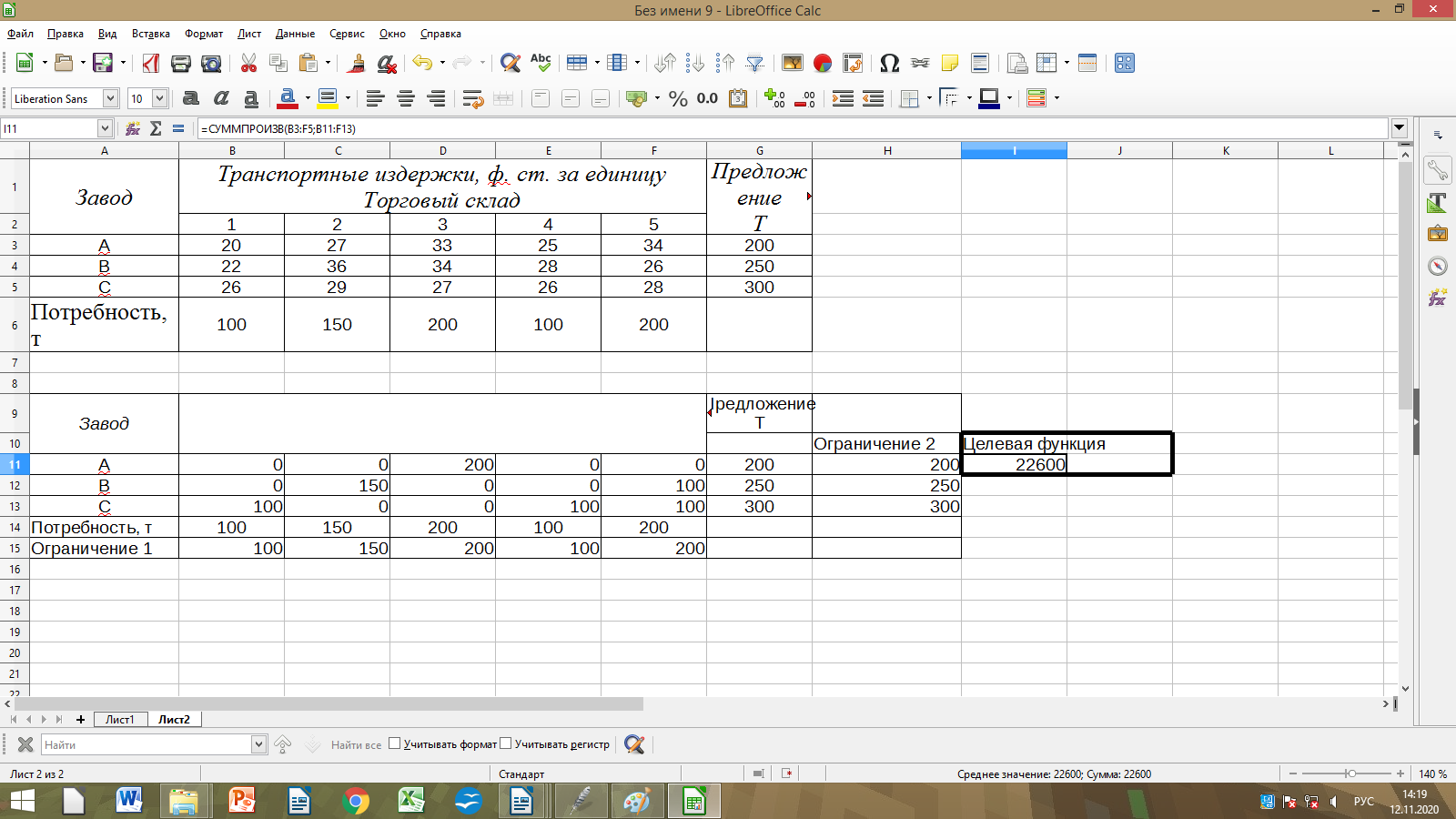
Увеличить ***L*** можно только увеличив ***X***. Однако ***X*** можно увеличивать бесконечно, исходя из системы уравнений. Следовательно, критерий ***L*** будет принимать неограниченно большие значения, решения задачи симплекс-методом не существует. В этом случае говорят, что имеет случай бесконечного симплекса.

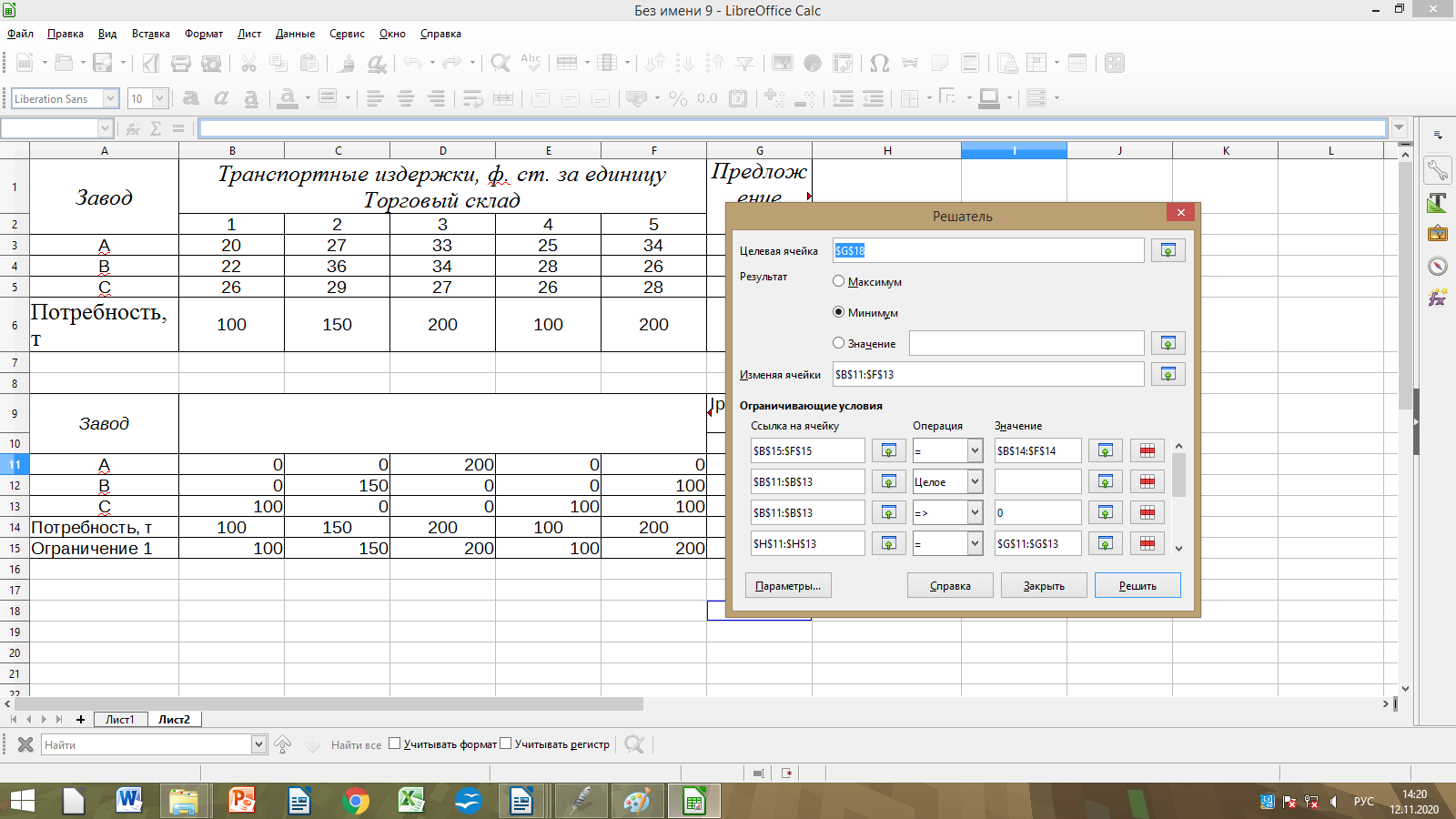
**Ход работы:**

Три завода поставляют некоторую разновидность стали на пять торговых складов. Спрос каждого торгового склада в декабре, наличие стали на заводах, а также значения стоимости транспортировки 1 т стали приведены в нижеследующей таблице.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Завод* | *Транспортные издержки, ф. ст. за единицу*  *Торговый склад* | | | | | *Предложение*  *т* |
| *1* | *2* | *3* | *4* | *5* |
| A | 20 | 27 | 33 | 25 | 34 | 200 |
| B | 22 | 36 | 34 | 28 | 26 | 250 |
| C | 26 | 29 | 27 | 26 | 28 | 300 |
| Потребность, т | 100 | 150 | 200 | 100 | 200 |  |

Требуется определить минимальную стоимость транспортировки на декабрь.

Решатель:



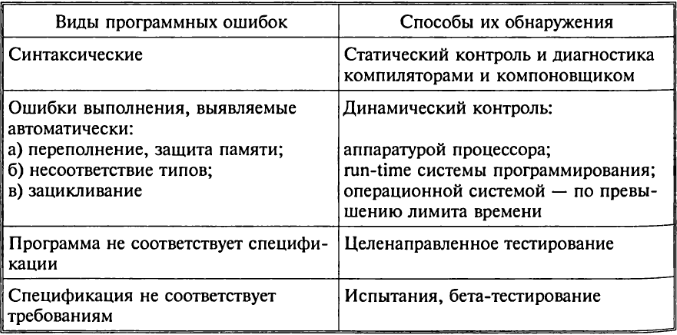
**Практическая работа 9.**

Тестирование и отладка программ.

Цель: изучить методы тестирования и отладки программ.

Выполнение программы с целью обнаружения ошибок называется тестированием. Виды ошибок и способы их обнаружения (Таблица 4)

Таблица 4.



**Тест** – это набор контрольных входных данных совместно с ожидаемыми результатами. В число входных данных время зависимых программ входят события и временные параметры. Ключевой вопрос – это полнота тестирование: какое количество, каких тестов гарантирует, возможно, более полную проверку программу? Исчерпывающая проверка на всем множестве входных данных недостижима.

* В любой нетривиальной программе на любой стадии ее готовности содержатся необнаруженные ошибки;
* Тестирование – технико-экономическая проблема, основа на компромиссе время – полнота. Поэтому нужно стремиться к возможно меньшему количеству хороших тестов с желательными свойствами.

Свойства Теста:

***Детективность*:** тест должен с большой вероятностью обнаруживать возможные ошибки.

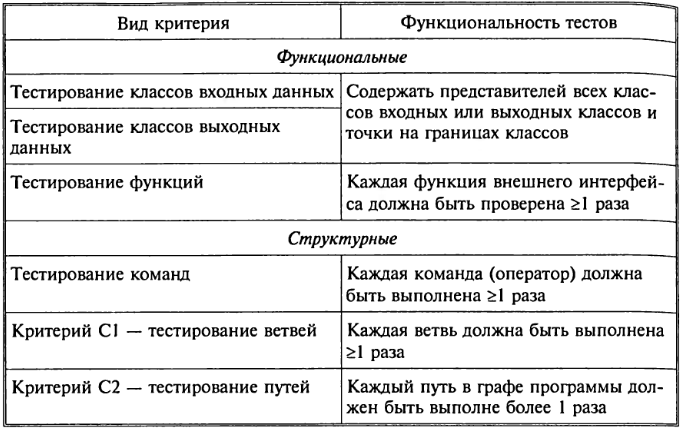
***Покрывающая способность*:** один тест должен выявлять как можно больше ошибок.

***Воспроизводимость*:** ошибка должна выявляться независимо от изменяющихся условий (временные соотношения) – это труднодостижимо для времязависимых программ, результаты которых часто невоспроизводимы.

Используются два вида критериев:

* Функциональные тесты составляются исходя из спецификации программы;
* Структурные тесты составляются исходя из текста программ.

Таблица 5. Виды критериев и их функциональность



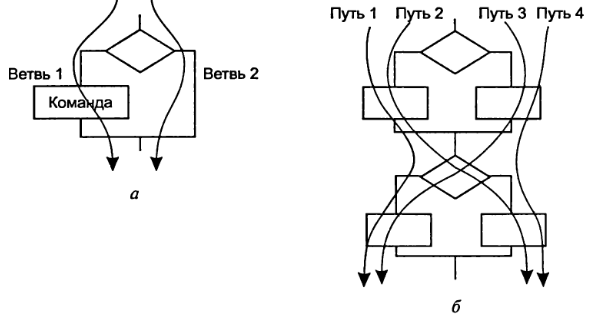


Рис.10 (Траектории вычислений при структурном тестировании).

**Тестирование «белого ящика» и «чёрного ящика».**

При тестировании «белого ящика» разработчик теста имеет доступ к исходному коду и может писать код, который связан с библиотеками тестируемого ПО. Это типично для юнит-тестирования, при котором тестируются только отдельные части системы.

При тестировании «чёрного ящика» тестировщик имеет доступ к ПО только через те же интерфейсы, что и пользователь, либо через внешние интерфейсы, позволяющие другому компьютеру или другому процессу подключиться к системе для тестирования.

**Порядок разработки тестов.**

По внешней спецификации разрабатываются тесты:

* Для каждого класса входных данных;
* Для граничных и особых значений входных данных.

Контролируются все ли классы выходных данных, при этом проверяются и добавляются при необходимости нужные тесты.

Систематическое тестирование предполагает также ведение журнала отладки, в котором фиксируется ошибка и в дальнейшем – исправление.

Готовятся тесты, проверяющие исключительные ситуации, недопустимые входные данные, аварийные ситуации.

**Аксиомы тестирования:**

1. Тест должен быть направлен на обнаружение ошибки, а не на подтверждение правильности программы.
2. Автор теста – не автор программы.
3. Тесты разрабатываются одновременно или до разработки программы.
4. Необходимо предсказывать ожидаемые результаты теста до его выполнения и анализировать причины расхождения результатов.
5. Предыдущее тестирование необходимо повторять после каждого внесения исправлений в программу.
6. Следует повторять полное тестирование после внесения изменений в программу или после переноса её в другую среду.
7. В те программы, в которых обнаружено много ошибок, необходимо дополнить первоначальный набор тестов.

**Автоматизация тестирования.**

А. Автоматизация прогона тестов актуальная для 5-6 аксиомы Майерса. Пишутся командные файлы для запуска программы с каждым тестом из набора и сравнением реального результата с ожидаемым.

Б. Средства автоматизации подготовки тестов и анализа их результатов.

1. Генераторы случайных тестов в заданных областях входных данных. Отладчики (для локализации ошибок).
2. Анализаторы динамики. Обычно входят в состав отладчиков; применяются для проверки соответствия тестовых наборов структурных критериев тестирования.
3. Средства автоматической генерации структурных тестов методом «символического выполнения» Кинга.

**Модульное тестирование.**

Модульное тестирование – это тестирование программы на уровне отдельно взятых модулей, функций или классов. Цель модульного тестирования заключается в выявлении локализованных в модуле ошибок в реализации алгоритмов, а также в определении степени готовности системы, к переходу на следующий уровень разработки и тестирования. Модульное тестирование проводится по типу «белого ящика», т.е. основан на знании внутренней структуры программы и часто включает те или иные методы анализа покрытия кода.

На уровне модульного тестирования проще всего обнаружить дефекты, связанные с алгоритмическими ошибками. Ошибки, связанные с неверной трактовкой данных, некорректной реализацией интерфейсов, совместимостью, производительностью и т.п., обычно пропускаются на уровне модульного тестирования и выявляются на более поздних стадиях тестирования.

Являясь по способу исполнения структурным тестированием или тестированием «белого ящика», модульное тестирование характеризуется степенью, в которой тесты выполняют или покрывают логику программы. Тесты, связанные с структурным тестирование, строятся по следующим принципам:

* На основе анализа потока управления. В этом случае элементы которые должны быть покрыты при прохождении тестов, определяются на основе структурных критериев тестирования С0, С1, С2. К ним относятся вершины, дуги, пути управляющего графа программы (УГП), условия, комбинации условий и т.п.
* На основе анализа потока данных, когда элементы должны быть покрыты, определяются на основе потока данных, т.е. информационного графа программы.

**Тестирование на основе потока управления.**

К особенностям использования структурных критериев тестирования С0, С1, С2 следует добавить критерий покрытия условий, заключающийся в покрытии всех логических условий в программе. Критерии покрытия решений и условий не заменяют друг друга, поэтому на практике используются комбинированный критерий покрытия условий/решений, совмещающий требования по покрытию и решений, и условий.

**Тестирование на основе потока данных.**

Этот вид тестирования направлен на выявление ссылок на неинициализированные переменные и избыточные присваивания.

Методы проектирования тестовых путей для достижения заданной степени тестированности в структурном тестировании.

Процесс построения набора тестов при структурном тестировании принято делить на три фазы:

* Конструирования УГП;
* Выбор тестовых путей;
* Генерация тестов, соответствующих тестовым путям.

Первая фаза соответствует статистическому анализу программы, задача которого состоит в получении графа программы и зависящего от него и от критерия тестирования множества элементов ,которые необходимо покрыть тестами.

На третей фазе по известным путям тестирования осуществляется поиск подходящий тестов, реализующих прохождение этих путей.

Вторая фаза обеспечивает выбор тестовых путей. Выделяют три подхода к построению тестовых путей:

* Статические методы;
* Динамические методы;
* Методы реализуемых путей.

Статические методы. Самое простое и лёгкое реализуемое решение – построение каждого пути посредством постепенного его удлинения за счёт добавления дуг, пока не будет достигнута выходная вершина управляющего графа программы.

Динамические методы. Такие методы предполагают построение полной системы тестов, удовлетворяющих заданному критерию, путём одновременного решения задачи построения покрывающего множества путей и тестовых данных.

Методы реализуемых путей. Данная методика заключается в выделении из множества путей подмножества всех реализуемых путей.

**Интеграционное тестирование.**

Интеграционное тестирование – это тестирование части системы, состоящей из двух и более модулей. Основная задача интеграционного тестирования – поиск дефектов.

Интеграционное тестирование применяется на этапе сборки модульного оттестированных модулей в единый комплекс. Известны два метода сборки модулей:

* Монолитный характеризующийся одновременным объединением всех модулей в тестируемых комплекс.
* Инкрементальный, характеризующийся пошаговым наращиванием комплекса программ с пошаговым тестирование собираемого комплекса. Выделяют две стратегии:

-“Сверху вниз” и “Снизу вверх”.

Особенности монолитного тестирования заключается в следующем для замены не разработанных к моменту тестирования модулей, кроме самого верхнего, необходимо дополнительно разрабатывать драйверы и или заглушки.

Недостатки нисходящего тестирования:

* Проблема разработки достаточно интеллектуальных заглушек, т.е заглушек , способных к использованию при моделировании различных режимов работы комплекса необходимых для тестирования;
* Сложных организации и разработки среды для реализации исполнения модулей в нужной последовательности
* Параллельная разработка модулей верхних и нижних уровней приводит к ней всегда эффективной реализации модулей из-за подстройки ещё тестировании модулей нижних уровней уже к оттестированным модулям верхних уровней

**Системное тестирование.**

Основная задача системного тестирования – выявление дефектов, связанных с работой системы в целом, таких как неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня несовместимость с окружением непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство в применении и т.п.

Категории тестов системного тестирования:

1. Полнота решения функциональных задач.
2. Стрессовое тестирование (предельные объемы).
3. Корректность использования ресурсов.
4. Оценка производительности.
5. Эффективность защиты.
6. Проверка инсталляции.
7. Корректность документации.

**Эффективность и оптимизация.**

Средства оптимизации, используемые компиляторами, делят на две группы:

* Машинно-зависимые (ориентированы на конкретный язык, выполняют оптимизацию кода на уровне машинных команд);
* Машинно-независимые (выполняют оптимизацию на уровне входного языка).

Способы уменьшения времени выполнения.

При их написании необходимо:

* Выносить вычисление константных, т.е не зависящих от параметров цикла, выражений из циклов;
* Избегать «длинных» операций;
* Минимизировать преобразование типов в выражениях;
* Оптимизировать записи условных выражений;
* Исключать многократные обращения к элементам массивов по индексам;
* Избегать использования различных типов в выражении и т.п.

**Практическая работа 10.**

Сопровождение и коллективная разработка программного обеспечения.

Цель: изучить виды программных документов, методы сопровождения и программы помогающие в коллективной разработке ПО.

**Виды программных документов.**

К программным документам относят документы, содержащие сведения, необходимые для разработки, сопровождения и эксплуатации ПО. Документирование с Единой системой программной документации.

*Спецификация* должна содержать перечень и краткое описание назначения всех файлов ПО, в том числе и файлов документации на него, и является обязательной для программных систем, а также компонентов.

*Ведомость держателей подлинников* должна содержать список предприятий, на которых хранятся подлинники программных документов. Необходимость этого документа определяется на этапе разработки и утверждения тех. задания только для программного обеспечения со сложной архитектурой.

*Текст программы* должен содержать текст программы с необходимыми комментариями. Необходимость этого документа определяется на этапе разработки и утверждения тех. задания.

*Описание программы* должно содержать сведения о логической структуре и функционировании программы.

*Ведомость эксплуатационных документов* должна содержать перечень эксплуатационных документов на программу, к котором относятся документы с кодами. Необходимость этого документа определяется на этапе разработки и утверждения тех. задания.

*Формуляр* должен содержать основные характеристики ПО, комплектность и сведения об эксплуатации программы.

*Описание применения* должно содержать основные характеристики ПО, комплектность и сведения об эксплуатации программы.

*Руководство системного программиста* должно содержать сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения.

*Руководство программиста* должно содержать сведения для эксплуатации ПО.

*Руководство оператора* должно содержать сведения для обеспечения процедуры общения оператора с вычислительной системой в процессе выполнения программного обеспечения.

*Описание языка* должно содержать описание синтаксиса и семантики языка.

*Руководство по тех. обслуживанию* должно содержать сведения для применения тестовых и диагностических программ при обслуживании технических средств.

*Программы и методика испытаний* должны содержать требования, подлежащие проверке при испытании ПО, а также порядок и методы их контроля.

*Пояснительная записка* должна содержать информацию о структуре и конкретных компонентах ПО, в том числе схемы алгоритмов, их общее описание, а также обоснование принятых технических и технико-экономических решений. Составляются на стадии эскизного и технического проектов.

Прочие документы могут составляться на любых стадиях разработки.

**Пояснительная записка.**

Пояснительная записка должна содержать всю информацию, необходимую для сопровождения и модификации ПО: сведения о его структуре и конкретных компонентах, общее описание алгоритмов и их схемы, а также обоснование принятых технических и технико-экономических решений.

Содержание пояснительной записки по стандарту ГОСТ должно включать следующие разделы:

1. Введение;

2. Назначение и область применения;

3. Технически характеристики;

4. Ожидаемые технико-экономические показатели;

5. Источники, используемые при разработке.

Пояснительная записка составляется профессионалами в области разработки ПО для специалистов того же уровня квалификации. Следовательно, в ней уместно использовать специальную терминологию.

**Руководство пользователя.**

Составление документации для пользователей имеет свои особенности, связанные с тем, что пользователь не является профессионалом в области. Поэтому существует рекомендации по написание подобной программной документации:

1. Учитывайте интересы пользователей – руководство должно содержать все инструкции, необходимые пользователю;

2. Излагайте ясно, используйте короткие предложения;

3. Избегайте технического жаргона;

4. Будьте точны и рациональны – длинные и запутанные руководства никто не читает.

Руководство пользователя, как правило, содержит следующие разделы:

1. Общие сведения о программном продукте;

2. Описание установки;

3. Описание запуска;

4. Инструкции по работе (или описание пользовательского интерфейса);

5. Сообщение пользователю.

**Руководство системного программиста.**

Руководство системного программиста должно содержать следующие разделы:

1. Общие сведения о программном продукте;

2. Структура;

3. Настройка;

4. Проверка;

5. Дополнительные возможности;

6. Сообщения системного программисту.

**Практическая работа 11.**

Коллективная разработка программного обеспечения.

**Коллективная разработка ПО** - стратегия, рабочий процесс и набор программного обеспечения, способствующие совместной работе различных организаций, программистов над одним изделием.

Из-за больших объемов проектов разработка программного обеспечения ведется коллективом специалистов. Работая в коллективе, отдельные специалисты должны взаимодействовать друг с другом, обеспечивая целостность проекта, что при отсутствии удовлетворительных средств описания проведения сложных систем, достаточно сложно. Причем чем больше коллектив, тем сложнее организовать процесс работы.

**Пакеты прикладных программ.**

Для визуализации, специфицирования, конструирования и документирования программных систем необходимо рассматривать их с различных точек зрения. Рассмотрим наиболее применяемые пакеты прикладных программ для коллективной разработки ПО.

**Система контроля версий Microsoft Visual SourceSafe**.

Microsoft Visual SourceSafe (VSS) – файл-серверная система управления версиями, предназначенная для небольших команд разработчиков. Она позволяет хранить в общем хранилище файлы, разделяемые несколькими пользователями, для каждого файла хранится история версий.

Там, где VSS недостаточно, на замену предлагается другой продукт – Visual Studio Team Foundation Server.

**Система контроля версий Subversion.**

Subversion – свободно распространяемая система управления версиями с открытым кодом. Это централизованная система, данные хранятся в едином хранилище. При сохранении новых версий используется дельта-компрессия, т.е система находит отличия новой версии от предыдущей и записывает только их, избегая ненужного дублирования данных.

**Литература**

1. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: учеб. пособие / Г.Н. Федорова. – М. :КУРС : ИНФРА-М, 2018. – 336 с. (Среднее Профессиональное образование)./ ЭБС «ZNANIUM»
2. Технология разработки программного обеспечения: учеб. пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д Сидорова-Виснадул; под ред. Л.Г. Гагариной. М. : ИД «Форум» : ИНФРА-М, 2018. – 400 с. / ЭБС «ZNANIUM»
3. Математическое моделирование технических систем: учебник / В.П. Тарасик. – Минск : Новое знание; М. : ИНФРА-М, 2018. – 592 с. / ЭБС «ZNANIUM»

**Источники из интернета:**

1. http:// www.interface.ru/home.asp?artId=3987
2. http://www.tehprog.ru/index.php\_page=lecture0111.html
3. https://unetway.com/tutorial/soprovozdenie-podderzka-programmnogo-obespecenia