динаху

Вопросы к экзамену по модулю ПМ.05

1) Что такое СКД? Как создать?

СКД расшифровывается как система компоновки данных. СКД - механизм платформы 1С для разработки отчетов. Отчет создается без необходимости писать программный код. С помощью определенного конструктора и настроек, произведенных в нем, программист задает желаемый результат, система компоновки данных это понимает и выводит этот результат пользователю. Если потребовалось что-то изменить в отчете, достаточно вновь обратиться к конструктору, внести необходимые изменения и измененный отчет готов. Минус СКД в том, что его настройки достаточно сложны и не все пользователи усваивают их быстро.

Какова технология создания отчета СКД:

- Написать запрос 1С в СКД, который обеспечивает получение данных

- Указать СКД роль полей (вычисляемые поля, ресурсы)

- Ввести настройки СКД по умолчанию.

Пользователю остается возможность изменить множество настроек по своему желанию.

СКД имеет значительное преимущество как для пользователя, так и для программиста:

- Программист – избавляет от написания программы для выполнения отчета и настроек

- Пользователь – получает значительный доступ к настройкам отчета.

Во всех новых конфигурациях 1С все отчеты будут использованы только на СКД 1С.

2) Что такое запрос? Как написать запрос?

Механизм запросов — это один из способов доступа к данным, которые поддерживает платформа. Используя этот механизм, разработчик может читать и обрабатывать данные, хранящиеся в информационной базе; изменение данных с помощью запросов невозможно. Это объясняется тем, что запросы специально предназначены для быстрого получения и обработки некоторой выборки из больших массивов данных, которые могут храниться в базе данных.

Чтобы написать запрос в 1С, нужно открыть конструктор запросов. Для этого нужно кликнуть правой кнопкой мыши в любой части любого модуля в конфигураторе 1С.

Если кликнуть в чистом поле, то конструктор начнёт создавать новый запрос. Если случайно попасть в существующий запрос, то конструктор просто откроет его.

Также конструктор запросов можно использовать для проверки синтаксиса запроса.

3) Какие существуют циклы в 1С? Написать их синтаксис

Цикл Для

Оператор цикла Для используется для выполнения некоторого кода заданное количество повторений. Для этого используется счетчик. Синтаксис цикла выглядит так:

```
Для <Счетчик> = <Выражение1> По <Выражение2> Цикл

    // Операторы

КонецЦикла;
```

Цикл Пока

Данный тип цикла предназначен для циклического выполнения тела цикла до тех пор, пока выполняется условие.

Синтаксис цикла выглядит так:

```
Пока <Логическое выражение> Цикл

    // Операторы

КонецЦикла;
```

Цикл Для Каждого

Данный цикл служит для циклического обхода элементов какой-нибудь коллекции – массива, структуры, таблицы значений, и т.д. Как правило, при этом в теле цикла производятся какие-то действия с каждым полученным элементом.

```
Для каждого <Имя переменной 1> Из <Имя переменной 2> Цикл

// Операторы

[Прервать;]

// Операторы
```

```
[Продолжить;]

// Операторы

КонецЦикла;
```

4) Что такое процедура и функция? Написать их синтаксис

Процедура или **функция** – логически самостоятельная именованная часть программного модуля, предназначенная для выполнения некоторой последовательности операторов, в которую могут передаваться аргументы в виде параметров. Различие между ними состоит в том, что функция может возвращать вычисленное значения, а процедура – нет.

```
Процедура <Имя_процедуры>([[Знач] <Парам1> [=<ДефЗнач>], ... ,[Знач] <ПарамN> [=<ДефЗнач>]])[Экспорт]

// Объявления локальных переменных;

// Операторы;

...

[Возврат;]

// Операторы;

...

КонецПроцедуры
```

<Имя_проц>

Назначает имя процедуры.

Знач

Необязательное ключевое слово, которое указывает на то, что следующий за ним параметр передается по значению, т.е. изменение значения формального параметра при выполнении процедуры никак не повлияет на фактический параметр, переданный при вызове процедуры. Если это ключевое слово не указано, то параметр процедуры передается по ссылке, то есть изменение внутри процедуры значения формального параметра приведет к изменению значения соответствующего фактического параметра.

<Парам1>, ..., <ПарамN>

Необязательный список формальных параметров, разделяемых запятыми. Значения формальных параметров должны соответствовать значениям передаваемых при вызове процедуры фактических параметров. В этом списке определяются имена каждого из параметров так, как они используются в тексте процедуры. Список формальных параметров может быть пуст.

=<ДефЗнач>

Необязательная установка значения параметра по умолчанию. Параметры с установленными значениями по умолчанию можно располагать в любом месте списка формальных параметров (подробнее см. "Передача параметров процедур и функций").

Экспорт

Необязательное ключевое слово, которое указывает на то, что данная процедура является доступной из других программных модулей.

// Объявления локальных переменных

Объявляются локальные переменные, на которые можно ссылаться только в рамках этой процедуры (см. оператор Перем).

// Операторы

Исполняемые операторы процедуры.

Возврат

Необязательное ключевое слово, которое завершает выполнение процедуры и осуществляет возврат в точку программы, из которой было обращение к процедуре. Использование данного оператора в процедуре не обязательно.

КонецПроцедуры

Обязательное ключевое слово, обозначающее конец исходного текста процедуры, завершение выполнения процедуры.

Возврат в точку, из которой было обращение к процедуре.

```
Функция <Имя_функции>([[Знач] <Парам1>[=<ДефЗнач>], ... ,[Знач] <ПарамN>[=<ДефЗнач>]])[Экспорт]

//Объявления локальных переменных;

// Операторы ;

...

Возврат <Возвращаемое значение>;
```

```
// Операторы ;

...

КонецФункции
```

<Имя_функции>

Назначает имя функции.

Знач

Необязательное ключевое слово, которое указывает на то, что следующий за ним параметр передается по значению, т.е. изменение значения формального параметра при выполнении функции никак не повлияет на фактический параметр, переданный при вызове функции. Если это ключевое слово не указано, то параметр функции передается по ссылке, то есть изменение внутри функции значения формального параметра приведет к изменению значения соответствующего фактического параметра.

<Парам1>, ..., <ПарамN>

Необязательный список формальных параметров, разделяемых запятыми. Значения формальных параметров должны соответствовать значениям передаваемых при вызове функции фактических параметров. В этом списке определяются имена каждого из параметров так, как они используются в тексте функции. Список формальных параметров может быть пуст.

=<ДефЗнач>

Необязательная установка значения параметра по умолчанию. Параметры с установленными значениями по умолчанию можно

располагать в любом месте списка формальных параметров (подробнее см. раздел "Передача параметров процедур и функций").

Экспорт

Необязательное ключевое слово, которое указывает на то, что данная функция является доступной из других программных модулей.

// Объявления локальных переменных

Объявляются локальные переменные, на которые можно ссылаться только в рамках этой функции (см. оператор Перем).

// Операторы

Исполняемые операторы функции.

Возврат <Возвращаемое значение>

Ключевое слово, которое завершает выполнение функции и возвращает указанное значение в выражение, в котором используется функция.

В качестве возвращаемого значения может выступать выражение или переменная, значение которого содержит результат обращения к функции.

КонецФункции

Обязательное ключевое слово, обозначающее конец исходного текста функции.

5) Что такое события? Где находятся все события формы? Что такое обработчик события?

События в 1С — это особые ситуации, которые происходят при выполнении программного кода.

Они могут быть связаны с различными событиями, такими как:

нажатие кнопки,

изменение значения поля,

открытие формы и т.д.

Каждое событие вызывает определённую реакцию, которая может быть предопределена или настраиваема пользователем.

Управление событиями осуществляется на уровне объектов, которые могут генерировать и обрабатывать события.

События формы в 1С:Предприятие хранятся в модуле формы.

Обработчик события в 1С:Предприятие - это специальный программный код, который определяет реакцию программы на определенные события.

6) Что такое печатная форма? С помощью чего выполняется вывод печатной формы? Что представляет макет табличного документа?

Печатная форма в 1С:Предприятие - это специальный программный код, который определяет способ печати и форматирования документов. В 1С:Предприятие печатные формы используются для

создания отчетов, накладных, счетов-фактур и других документов, которые необходимо распечатать. Печатная форма может содержать различные элементы, такие как заголовки, подзаголовки, таблицы, графики и т.д.

Вывод печатной печатной формы выполняется с помощью команды?

Табличный документ в 1С 8.3 - это объект встроенного языка, который используется для создания печатных форм документов и отчетов. Является электронной таблицей, так как состоит из строк и столбцов и имеет функциональность, определенную следующими методами: ввод данных, группировка элементов, расшифровка, примечание в ячейках, оформление ячеек, форматирование, сохранение. Табличный документ использоваться как сам по себе, и может входить в состав любой из форм и служит для отображения информации.

7) Что такое планы видов характеристик? Когда их лучше всего использовать?

Планы видов характеристик — это прикладные объекты конфигурации. Они предназначены для хранения информации о характеристиках различных объектов.

С их помощью пользователь может создавать всевозможные характеристики, описывать тип этих характеристик и задавать их значения. Например, для того, чтобы описывать товары произвольным количеством произвольных характеристик (цвет, размер, запах и т. д.).

Планы видов характеристик могут быть настроены на отображение и скрытие определенных элементов управления в зависимости от условий. Например, при выполнении определенной команды может быть скрыт или отображен определенный элемент управления.

Планы видов характеристик могут быть настроены на выполнение определенных действий с данными. Например, при выполнении определенной команды может быть выполнен код, который сохраняет измененные данные в базе данных.

В целом, планы видов характеристик в 1С:Предприятие позволяют организовать структуру данных в системе и обеспечивают гибкость при работе с данными. Они могут быть использованы для создания различных типов документов, справочников, регистров и т.д.

8) Общие сведения о формах различных объектов.

В «1С:Предприятии 8.1» существует несколько основных типов форм:

1. Форма списка — используется для отображения списков элементов информации, хранящейся в конфигурации.
2. Форма элемента — отображает один логический объект конфигурации, позволяя просматривать и редактировать необходимую информацию о нём.
3. Форма выбора — позволяет выполнять действия, подобные форме списка, например, завести новый элемент.
4. Форма группы — специально разработанная форма для ввода данных, относящихся только к конкретной группе.

Для каждого объекта конфигурации может быть задано несколько форм, которые облегчают ввод и обработку информации, хранящейся в этом объекте.

9) Командный интерфейс формы. Виды команд.

Командный интерфейс формы в 1С:Предприятие - это набор команд, которые могут быть выполнены с помощью формы. Виды команд в 1С:Предприятие могут быть различными и зависят от конкретной задачи, которую нужно решить. Например, команды могут быть связаны с открытием, закрытием, сохранением формы, выполнением определенных действий с данными, отображением или скрытием элементов формы и т.д.

Командный интерфейс формы может быть настроен в конфигураторе 1С или с помощью встроенного языка программирования. Команды могут быть добавлены на форму с помощью специального элемента управления "Команда". Каждая команда имеет свой уникальный идентификатор и может быть связана с определенным обработчиком событий, который определяет, что должно произойти при выполнении команды.

Виды команд в 1С:Предприятие могут быть следующими:

1. Команды открытия и закрытия формы.
2. Команды сохранения и отмены изменений.
3. Команды навигации по форме.
4. Команды обработки данных.
5. Команды отображения и скрытия элементов формы.

6. Команды выполнения определенных действий с данными.

   Каждая команда может быть настроена на выполнение определенных действий при ее выполнении. Например, при выполнении команды открытия формы может быть выполнен код, который загружает данные из базы данных и отображает их на форме. При выполнении команды сохранения изменений может быть выполнен код, который сохраняет измененные данные в базе данных.

10) Из чего состоит редактор управляемой формы?

Редактор формы используется для создания и редактирования форм объектов прикладного решения. Формы объектов используются системой для визуального отображения данных в процессе работы пользователя.

Любая форма представляет совокупность нескольких составляющих:

- элементов — объектов, определяющих визуальное представление формы и осуществляющих взаимодействие с пользователем,
- командного интерфейса — совокупности команд, отображаемых в форме;
- реквизитов — объектов, данные которых форма использует в своей работе.
- команд — действий, которые определены в данной конкретной форме,
- параметров — объектов, значения которых характеризуют саму форму, используются при ее создании и остаются постоянными в процессе «жизни» формы,

- модуля — программы на встроенном языке, отвечающей за работу с элементами и за обработку событий;

11) Что такое модуль? Какие бывают модули?

Модуль – это текст на встроенном языке, расположенный в определенном месте конфигурации. Полностью название звучит как “программный модуль”, но в основном используется сокращенный вариант. Поэтому далее мы везде под словом “модуль” будем подразумевать именно программный модуль 1С.

Виды модулей:

Модули конфигурации. В зависимости от параметров конфигуратора, модуль приложения может быть один, либо разделиться на модуль управляемого и обычного приложения.

- Модуль управляемого приложения. Модуль служит для обработки событий запуска клиентского приложения и завершения работы, полнотекстового поиска, а также обработки внешних событий от оборудования (сканеры, кассы, весы и др.). Соответственно, содержимое этого модуля исполняется всегда только на клиенте.
- Модуль сеанса. Служит для инициализации параметров сеанса. Модуль исполняется на сервере, всегда в привилегированном режиме.

- Модуль внешнего соединения. Предназначен для того, чтобы обработать события начала и завершения работы при внешнем (программном) соединении. Окно клиентского приложения при этом не запускается, поэтому данный модуль исполняется на сервере.
- Модуль обычного приложения. Выполняет те же функции, что и модуль управляемого приложения, но только в варианте запуска “Толстый клиент обычное приложение”.

Общие модули в 1С

Это объекты метаданных, располагающиеся в ветке “Общие”.

Основное предназначение общих модулей – это структурированное хранение процедур и функций, которые могут быть использованы во многих местах конфигурации, с целью избежать дублирования кода. Процедуры и функции, связанные между собой логически, объединяются в один общий модуль.

Модуль формы

В 1С у каждой формы есть собственный модуль, который прежде всего предназначен для обработки событий при интерактивном взаимодействии пользователя и экранной формы. Это может быть нажатие кнопки, включение переключателя, заполнение полей, и многие другие события. Помимо обработчиков событий, связанных с элементами управления, существуют также события, связанные с

самой формой – например, обработчики открытия и закрытия формы.

Модуль объекта

В модуле объекта реализуется прикладная логика, связанная с обработкой одного конкретного объекта – одного элемента справочника, одного документа, одного счета из плана счетов, и т.п. Состав обработчиков в модуле объекта зависит от типа этого объекта. Например, и у документа, и у справочника есть обработчик “ПриЗаписи”, но только у документа есть обработчик “ОбработкаПроведения”.

Модуль набора записей

Данные модули существуют у объектов конфигурации, имеющих табличную сущность – регистров. Они выполняют ту же роль, что и модули объектов – позволяют проверить заполнение набора записей, выполнить обработчики при записи, и т.п.

Модуль менеджера значения

Модуль менеджера значения — модуль константы, в котором описываются обработчики проверки заполнения, перед записью и при записи.

Модуль менеджера

Модули менеджера доступны у многих объектов конфигурации – документов, констант, регистров, и др. Используя модуль менеджера, можно переопределить стандартное поведение платформы, касающееся сразу всех объектов, а не одного.

Прочие модули в платформе 1С

- Модули HTTP и Web-сервисов
- Модули команд
- Модули ботов (с появлением системы взаимодействия в 1С появилась возможность создавать собственных ботов)
- Модули сервисов интеграции

12) Дайте определение понятию «Тестирование». Перечислите цели тестирования;

Тестирование – процесс, содержащий в себе все активности жизненного цикла, касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатов работ с целью определить, что они соответствуют описанным требованиям, показать, что они подходят для заявленных целей и для определения дефектов.

Цели: оценка рабочих продуктов; проверка, все ли указанные требования выполнены; работает ли объект тестирования так, как ожидают пользователи и заинтересованные лица; создание уверенности в уровне качества объекта тестирования;

предотвращение дефектов; обнаружение отказов и дефектов; предоставление актуальной информации для принятия решений.

13) Дайте определение понятию «Тестировщик». Что должен знать и уметь тестировщик?

Тестировщик – простыми словами, человек, проверяющий работоспособность системы. Именно он ищет ошибки и баги, допущенные при разработке программного обеспечения, которые мешают корректной работе.

Что должен знать и уметь тестировщик?

- Классификация тестирования, методы и инструмент, создание сценариев тестирования
- знания основ программирования, протоколов HTTP, умение работать с базами данных и системами контроля версий
- умение работать с инструментами автоматического тестирования
- обладать следующими качествами: внимательность к мелочам, критическое мышление, умение анализировать информацию.

14) Виды тестирования. Перечислите функциональные виды тестирования;

Виды тестирования:

- функциональные

- нефункциональные
- связанные с изменениями

Функциональные тесты базируются на функциях и особенностях, а также на взаимодействии с другими системами.

Виды функциональных тестов:

- функциональное тестирование
- тестирование безопасности
- тестирование взаимодействия

15) Виды тестирования. Перечислите нефункциональные виды тестирования;

Виды тестирования:

- функциональные
- нефункциональные
- связанные с изменениями

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами.

Виды нефункциональных тестов:

- тестирование производительности
  - нагрузочное тестирование

- стрессовое тестирование
- тестирование стабильности или надежности
- объемное тестирование

- тестирование установки
- тестирование удобства пользования
- тестирование на отказ и восстановление
- конфигурационное тестирование

16) Виды тестирования. Виды тестов, связанные с изменениями;

Виды тестирования:

- функциональные
- нефункциональные
- связанные с изменениями

После проведения необходимых изменений, таких как исправление бага/дефекта, программное обеспечение должно быть пере тестировано для подтверждения того факта, что проблема была действительно решена.

Виды тестов, связанных с изменениями:

- дымовое тестирование
- регрессионное тестирование

17) Уровни тестирования. Модульное тестирование.

Уровень тестирования определяет то, над чем производятся тесты: над отдельным модулем, группой модулей или системой, в целом.

Компонентное (модульное тестирование) проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по-отдельности. Обычно компонентное (модульное) тестирование проводится, вызывая код, который необходимо проверить и при поддержке сред разработки, таких как фреймворки для модульного тестирования или инструменты для отладки.

Один из наиболее эффективных подходов к компонентному (модульному) тестированию – это подготовка автоматизированных тестов до начала основного кодирования (разработки) программного обеспечения.

18) Уровни тестирования. Системное тестирование;

Уровень тестирования определяет то, над чем производятся тесты: над отдельным модулем, группой модулей или системой, в целом.

Основной задачей системного тестирования является проверка как функциональных, так и нефункциональных требований в системе в целом.

Можно выделить два подхода к системному тестированию:

- на базе требований (для каждого требования пишутся тестовые случаи, проверяющие выполнение данного требования)

- на базе случаев использования (на основе представления о способах использования продукта создаются случаи использования системы)

19) Уровни тестирования. Интеграционное тестирование;

Уровень тестирования определяет то, над чем производятся тесты: над отдельным модулем, группой модулей или системой, в целом.

Интеграционное тестирование предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы.

Уровни интеграционного тестирования:

- Компонентный интеграционный уровень (проверяется взаимодействие между компонентами системы после проведения компонентного тестирования)
- Системный интеграционный уровень (проверяется взаимодействие между разными системами после проведения системного тестирования)

Подходы к интеграционному тестированию:

- Снизу вверх
- Сверху вниз
- Большой взрыв

20) Уровни тестирования. Приемочное тестирование;

Уровень тестирования определяет то, над чем производятся тесты: над отдельным модулем, группой модулей или системой, в целом.

Приемочное тестирование или приемо-сдаточное испытание – формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:

- Определения удовлетворяет ли система приемочным критериям
- Вынесения решения заказчиком или другим уполномоченным лицом принимается приложение или нет

Приемочное тестирование выполняется на основании набора типичных тестовых случаев и сценариев, разработанных на основании требований к данному приложению.

21) Дайте определение понятиям «Дефект» и «Отказ». Почему появляются ошибки?

Дефект – возможное следствие ошибки.

Отказ – это событие, при котором компонент или система неспособны выполнять в свои требуемые функции (задачи) в соответствии с нефункциональных требованиями.

Ошибки возникают из-за действий программиста на этапе разработки, которые приводят к тому, что в программном обеспечении содержится внутренний дефект, который в процессе работы программы может привести к неправильному результату.

22) Что такое отчеты об ошибках? Структура отчета об ошибке;

Баг-репорт – это технический документ, который подробно описывает ошибку в работе программы, приложения или другого ПО.

Структура отчета об ошибке:

- ID
- Краткое описание
- Проект
- Версия
- Серьезность бага
- Приоритет
- Статус
- Автор
- Исполнитель
- Шаги к воспроизведению
- Фактический результат
- Ожидаемый результат
- Дополнения

23) Автоматизированное тестирование. Цель автоматизации. Преимущества и недостатки автоматизированного тестирования;

Автоматизированное тестирование – это метод тестирования ПО, который выполняется с использованием специальных программных средств, которые, в свою очередь необходимы для выполнения набора тестовых примеров.

Цель – повысить эффективность, а также увеличить охват и скорость тестирования ПО, когда нужно повторять одни и те же тестовые сценарии.

Плюсы:

- Нагрузка на приложение
- Временной фактор
- Повторяемость

Минусы:

- Отсутствие обратной связи
- Отсутствие тестирования глазами пользователя
- Отсутствие возможности тестирования цвета, дизайна и эргономики
- Надежность
- Стоимость

24) Три подхода к тестированию программного обеспечения. Тестирование белого ящика;

Главная цель тестирования белого ящика – проверка кода, тестирование как внутренней структуры, так и дизайна.

Тестирование белого ящика:

- Шаг 1. Изучение исходного кода
- Шаг 2. Создание и внедрение алгоритмов проверки.

Методы тестирования:

Основной подход – анализ кода. Выборка недоработок в процессе внедрения алгоритмов тестирования. Все алгоритмы запускаются по несколько раз на разных участках кода, помогая изъять возможные проблемные участки.

25) Три подхода к тестированию программного обеспечения. Тестирование черного ящика;

Во время поведенческого тестирования или тестирования черного ящика, специалист не знает наверняка, что за продукт он тестирует. Внутренняя структура, приложение и дизайн остаются неизвестными для тестировщика.

Метод черного ящика применим на следующих уровнях тестирования:

- Тестирование системы
- Тестирование интеграции

- Приемочных испытаниях

26) Три подхода к тестированию программного обеспечения. Тестирование серого ящика;

Тестирование серого ящика предусматривает частичную осведомленность о внутренних процессах. Данный метод – это комбинация двух предыдущих подходов.

Подходы к тестированию:

- Тестирование шаблонов
- Тестирование матрицы
- Регрессионное тестирование
- Тестирование с использованием ортогонального массива

Практические задания к экзамену по модулю ПМ.05

1. Измените код Django таким образом, чтобы на странице отображалась информация: наименование группы, ФИО, электронную почту.

2. Напишите код Django, который осуществляет переход по страницам сайта.

Установка django:

python -m pip install Django

django-admin startproject mysite

cd mysite

Файлы:

Создать файл views.py, где есть файл urls.py

views.py

```
from django.http import HttpResponse


def home(request):
    return HttpResponse('Hello World!')


def about(request):
    return HttpResponse('About')


def contacts(request):
    return HttpResponse('Contacts')
```

urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
```

```
urlpatterns = [
    path('', views.home, name='home'),
    path('about/', views.about, name='about'),
            path('contacts/',    views.contacts,
name='contacts'),
    path('admin/', admin.site.urls),
]
```

Запуск проекта:

python manage.py runserver

3. Установите библиотеку django-bootstrap/аналог и продемонстрируйте его работу.

4. Создайте меню навигации используя django-bootstrap/аналог и продемонстрируйте её работу.

5. Создайте простую модель и сделайте вывод данных из этой модели

6. Создайте простую модель и запишите туда данные. Не используйте для выполнения работы панель-администратора.

7. Создайте форму регистрации, соблюдая валидацию данных. Добавьте запись данных в модель.

8. Создайте супер-пользователя, использовав консоль, а также обычного пользователя, использовав панель администратора.

9. Продемонстрируйте работу ограничения прав для администраторов и обычных пользователей, используйте панель администратора.

10. Создайте форму для подсчёта периметра и площади треугольника, используя модель Django.

11. Создайте форму для подсчета среднего заработка за квартал, используя модель Django

12. 1. Прочитать из текстового файла строку "Hellow word!". Удалить каждый второй символ и записать в этот же файл. Выполнить рекурсивно. Составить блок-схему

```
def deleteSecondChar(string, index):
    if index >= len(string):
        return string
    else:
                    nString = string[:index] +
string[(index+1):]
         return deleteSecondChar(nString, index +
1)


with open('text.txt', 'r+') as f:
    n = deleteSecondChar(f.read(), 1)
    f.write(n + "\n\n")
```

13. 2. Дан текстовый файл, в котором хранится список группы. Информация о каждом студенте сохранена в отдельной строке. Составить блок-схему. Информация хранится в следующем формате: Фамилия Имя

14. Составить программу, которая:

а) подсчитывает сколько студентов в группе имеют одинаковое с вами имя.

б) определяет есть ли в вашей группе однофамильцы.

Все найденные сведения вывести в файл. В формате:

Отчет составлен Ивановым Иваном.

Количество студентов в группе с именем - Иван - 3 студента.

В группе есть однофамилец - Иванов Петр.

```
def func(IMA):
    with open('Fio', 'r+', encoding='utf-8') as f:
        k = 0
        for string in f:
            if IMA in string:
                k += 1
        return(k)


def OdnoFamilia(one):
```

```
   with open('Fio', 'r+', encoding='utf-8') as f:
       kolvo = [line.split(None, 1)[0] for line in f]
       if kolvo.count(one) > 1:
           return('есть')
       else:
           return('нет')


all = open('itog.txt', 'w+',encoding='utf-8')
FAM = input('Введите кто составил отчет фамилия= ')
IMA = str(input('Имя = '))
all.write('Отчет составлен ' + FAM + ' ' + IMA +'\n')
all.write('Количество студентов в группе с именем- ' +
IMA + ' ' + str(func(IMA)))
all.write((' студент' if str(func(IMA)) == 1 or '
студента' else str(func(IMA)) > 1) + '\n')
all.write('В группе однофамильцев- '+ FAM + ' ' +
OdnoFamilia(FAM)+'\n')
all.close()
```

15. Создать файл, содержащий содержащим следующие сведения о торговом предприятии: код товара, наименование товара, фирма-поставщик, цена, количество, проданное за день. Составить блок-схему. Поиск осуществлять по следующим параметрам:

а) фирма-поставщик;

б) наименование товара и цена (диапазон);

в) фирма-поставщик, при этом определить общую стоимость проданных товаров данной фирмы и самый дорогой товар.

(ОТВЕТ НЕ СОВСЕМ ВЕРНЫЙ, ТОЛЬКО ОБЩАЯ ЛОГИКА ПО ЗАПИСИ ДАННЫХ В СЛОВАРЬ!!!)

```
A = dict()
file = open('sklad.txt','r')
x = 0
for line in file:
     if line[len(line) - 1] == '\n':
          line = line[:-1]
          l = line.rsplit()
          B = dict()
          B['id'] = l[0]
          B['company'] = l[1]
          B['operation'] = l[2]
          B['count'] = l[3]
          A[x] = B
          x += 1
print('Выберите необходимую операцию:'+'\n'+'1 - Найти
материал по коду'+'\n'+
```

```
    '2- список материалов от фирмы'+'\n'+'3 - Наличие на
складе заданного материала'+'\n'+'4 - Завершить.')
menu = int(input())
print()
if menu == 1:
    name = input('Введите код материала: ')
    f = False
    for i in A:
        if name == A[i]['id']:

print(A[i]['id'],A[i]['company'],A[i]['operation'],A[i]['count
'])
            f = True
    if f == False:
    print('Информация не найдена')
    print()
elif menu == 2:
    comp = input('Введите название фирмы: ')
    f = False
    for i in A:
        if comp == A[i]['company']:
            print(A[i]['id'],A[i]['company'])
            f = True
    if f == False:
    print('Информация не найдена')
    print()
elif menu == 3:
    mat = input('Введите код материала - ')
    s = 0
    f = False
    for i in A:
        if mat == A[i]['id']:
            print(A[i]['count'])
            s += int(A[i]['count'])
```

```
            f = True
    if f == True:
    print('Общее кол-во материала = ',s)
    else:
    print('Информация не найдена')
elif menu == 4:
 print('Завершение работы')
else:
 print('Вы ввели некорректные данные')
```

16. Реализовать метод DFS. Используя возможности графических библиотек языка Python написать метод рисующий граф. Ребра, образующие лес обхода в глубину закрасить в красный цвет. Составить блок-схему.

```
from PIL import Image, ImageDraw, ImageFont
from random import randint

imgDimensions = (1280, 720)
img = Image.new("RGB", imgDimensions, color="white")
imgDrawer = ImageDraw.Draw(img)
font = ImageFont.truetype("arial.ttf", 12)

file = open('graph2.txt')
f = True
for line in file:
    if f:
        N = int(line) + 1
        A = [0] * N
        for i in range(N):
```

```
            A[i] = [0] * N
        f = False
    else:
        x = int(line.split()[0])
        y = int(line.split()[1])
        A[x][y] = 1
        A[y][x] = 1

def coord(x, y):
    for i in range(len(x)):
        x[i] = randint(80, 1200)
        y[i] = randint(80, 740)

def new_coord_1(x, y):
    n = len(x)
    ni = int(n ** 0.5)
    nj = ni + 1
    if ni ** 2 == n:
        nj = ni
    elif ni * nj < n:
        ni = nj + 1

    a = (imgDimensions[0] - 100) // nj
    b = (imgDimensions[1] - 100) // ni

    x = 50
    y = 50
    z = 1
    for i in range(ni):
        for j in range(nj):
            X[z] = randint(x, x + a)
            Y[z] = randint(y, y + b)
            x = (x + a) % (imgDimensions[0] - 100)
            z = z + 1
```

```
            if z >= n:
                return x, y
        y = y + b

def draw_Graph(a, x, y):

    for i in range(len(a)):
        for j in range(i + 1, len(a)):
            if a[i][j] == 1:
                imgDrawer.line([(x[i], y[i]), (x[j], y[j])],
"black", 2)
    for i in range(1, len(x)):
        imgDrawer.ellipse([(x[i] - 10, y[i] - 10), (x[i] + 10,
y[i] + 10)], 'white', 'black', 2)
        imgDrawer.text((x[i] - 4, y[i] - 4), str(i), (0, 0,
0), font)

def dfs(A, v, color, p, q):
    color[v] = 1
    q.append(v)
    for u in range(1, len(A)):
        if (A[v][u] != 0) and (color[u] == 0):
            p[u] = v
            dfs(A, u, color, p, q)
    color[v] = 2

def draw_dfs(p):
    for u in range(1, len(p)):
        if p[u] != 0:
            imgDrawer.line([(X[u], Y[u]), (X[p[u]], Y[p[u]])],
"red", 3)

    for i in range(1, len(X)):
```

```
        imgDrawer.ellipse([(X[i] - 10, Y[i] - 10), (X[i] + 10,
Y[i] + 10)], "white", "black", 2)
        imgDrawer.text((X[i] - 4, Y[i] - 4), str(i), (0, 0,
0), font)

X = [0] * N
Y = [0] * N
file.close()
new_coord_1(X, Y)
draw_Graph(A, X, Y)
p = [0] * N
color = [0] * N
q = []

for v in range(1, len(A)):
    if color[v] == 0:
        dfs(A, v, color, p, q)
draw_dfs(p)

img.show()
```

17. Решить задачу топологической сортировки для неориентированного невзвешенного графа, используя метод обхода в глубину DFS. Составить блок-схему.

```
file = open('text.txt')
f = True
for line in file:
    if f:
        N = int(line) + 1
        A = [0] * N
        for i in range(N):
```

```
            A[i] = [0] * N
        f = False
    else:
        x = int(line.split()[0])
        y = int(line.split()[1])
        A[x][y] = 1
        A[y][x] = 1

def top_sort(A, v, color, p, q):
    color[v] = 1
    q.append(v)
    for u in range(1, len(A)):
        if (A[v][u] != 0) and (color[u] == 0):
            top_sort(A, u, color, p, q)
    k.insert(0, v)
    color[v] = 2

file.close()
p = [0] * N
color = [0] * N
q = []
k = []

for v in range(1, len(A)):
    if color[v] == 0:
        top_sort(A, v, color, p, q)
print(q)
print(k)
```

18. Реализовать волновой алгоритм, используя метод обхода в ширину BFS. Составить блок-схему.

```
file = open('text.txt')
f = True
for line in file:
    if f:
        N = int(line) + 1
        A = [0] * N
        for i in range(N):
            A[i] = [0] * N
        f = False
    else:
        x = int(line.split()[0])
        y = int(line.split()[1])
        A[x][y] = 1
        A[y][x] = 1

for i in range(N):
    for j in range(i, N):
        A[j][i] = A[i][j]

for i in range(N):
    for j in range(N):
        print("%3d" % A[i][j], end='')
    print()
Q = [1]
i = 0
color = [True] * N
color[Q[i]] = False
while i < len(Q):
    for j in range(N):
        if A[Q[i]][j] == 1 and color[j]:
            Q.append(j)
```

```
            color[j] = False
    i += 1
print(Q)
```

19. Создать словарь, состоящий из N элементов по следующему правилу: ключи – числа от 1 до N, значения – первые N простых чисел. Величина N вводиться с клавиатуры. Например, при N = 7 программа формирует словарь из 7-ми простых чисел – {1:2, 2:3, 3:5, 4:7, 5:11, 6:13, 7:17}. Составить блок-схему

```
def is_prime(num):
  if num < 2:
      return False
  for i in range(2, int(num**0.5) + 1):
      if num % i == 0:
          return False
  return True



def generate_prime_dictionary(N):
```

```
    primes_dict = {}
    num = 2  # начинаем с первого простого числа
    i = 1



    while i <= N:
        if is_prime(num):
            primes_dict[i] = num
            i += 1
        num += 1



    return primes_dict



N = int(input("Введите значение N: "))
prime_dict = generate_prime_dictionary(N)
print("Словарь из первых N простых чисел:")
print(prime_dict)
```

20. Даны два списка одинаковой длины. Создать словарь, так чтобы элементы первого списка были ключами словаря, а элементы второго списка значениями. Составить блок-схему.

```
a = [1,2,3,4,5]

b = ['a', 'b', 'c', 'd', 'e']



dct = {i:j for i, j in zip(a,b)}



print(dct)
```

21. Построить диаграмму последовательности для любого процесса по теме …

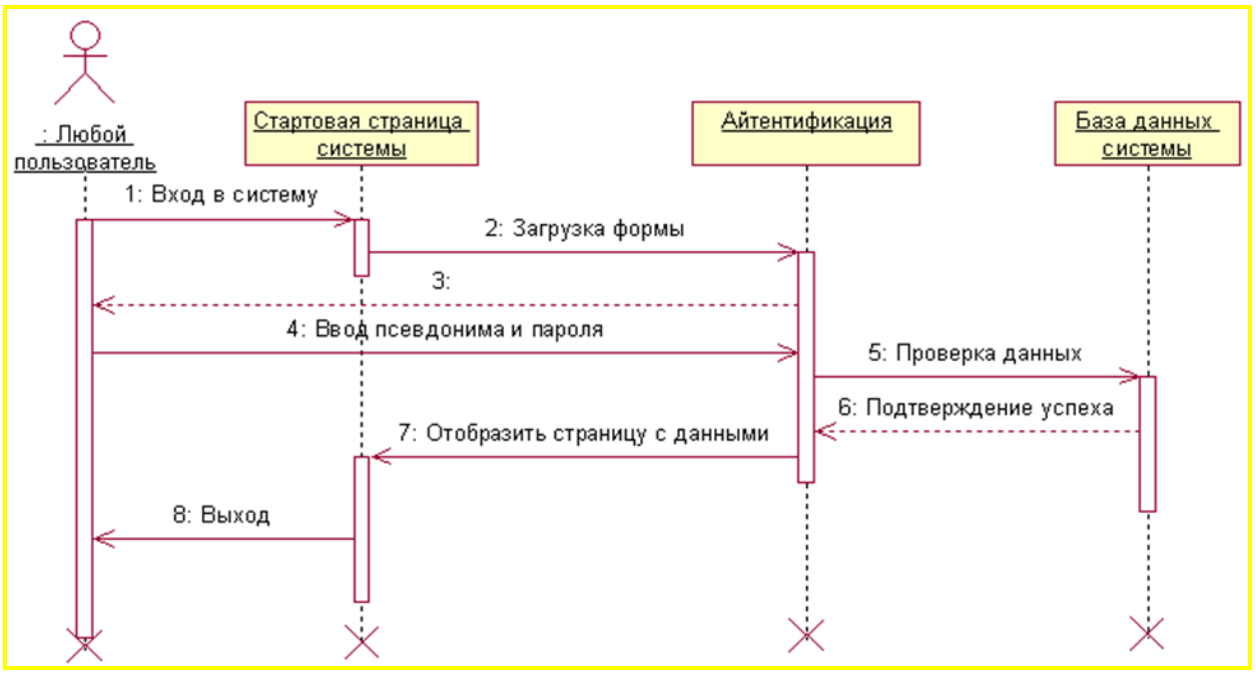


Рис.1 Диаграмма последовательности.

22. Построить диаграмму DFD (контекстный уровень и декомпозицию первого уровня) по теме …

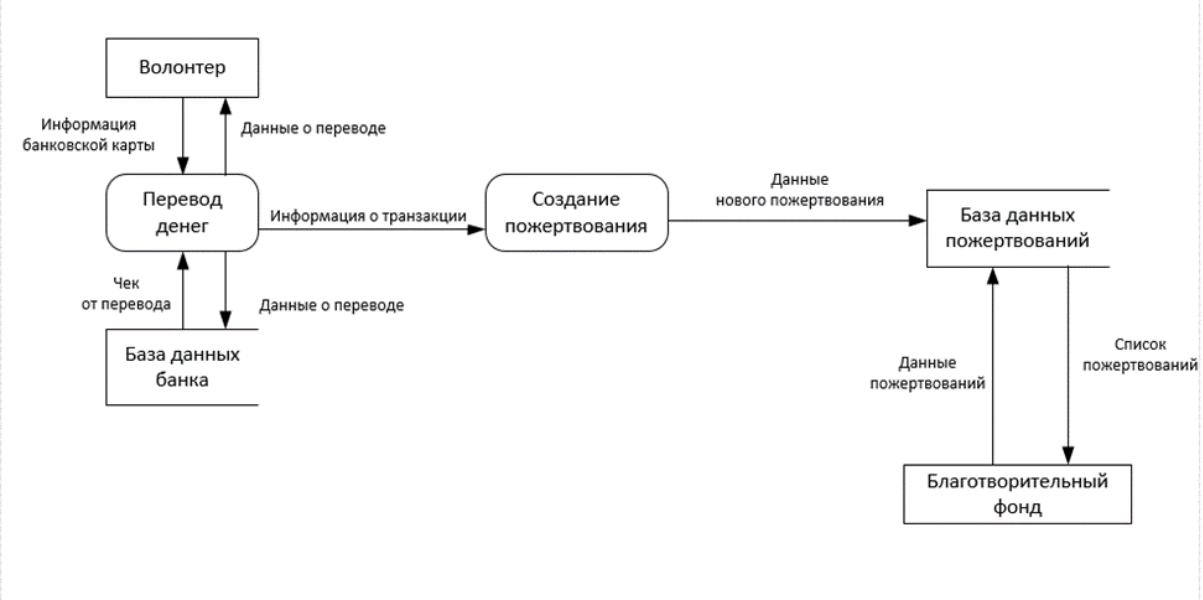
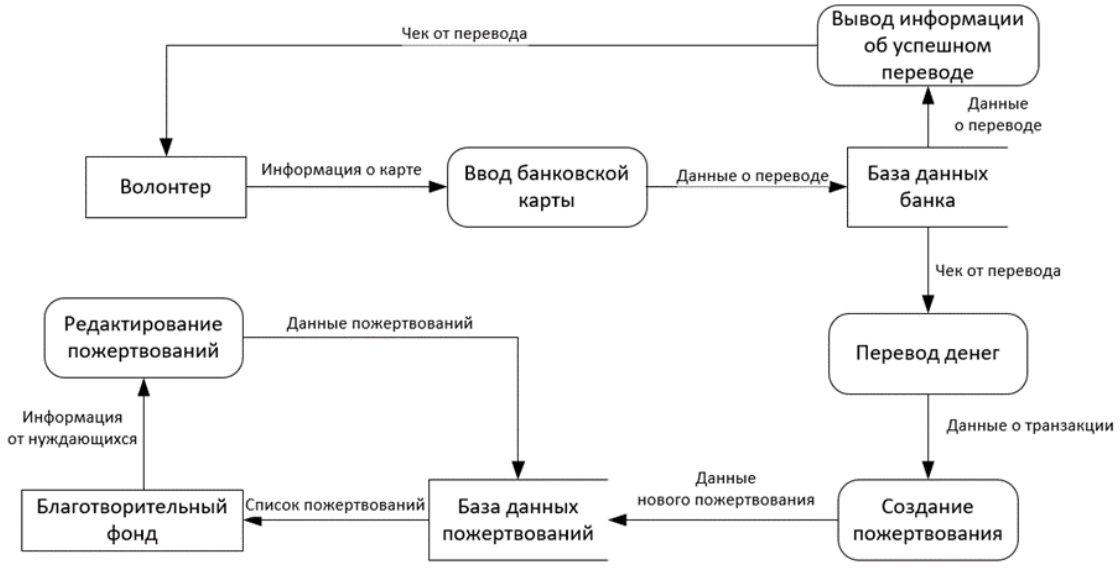
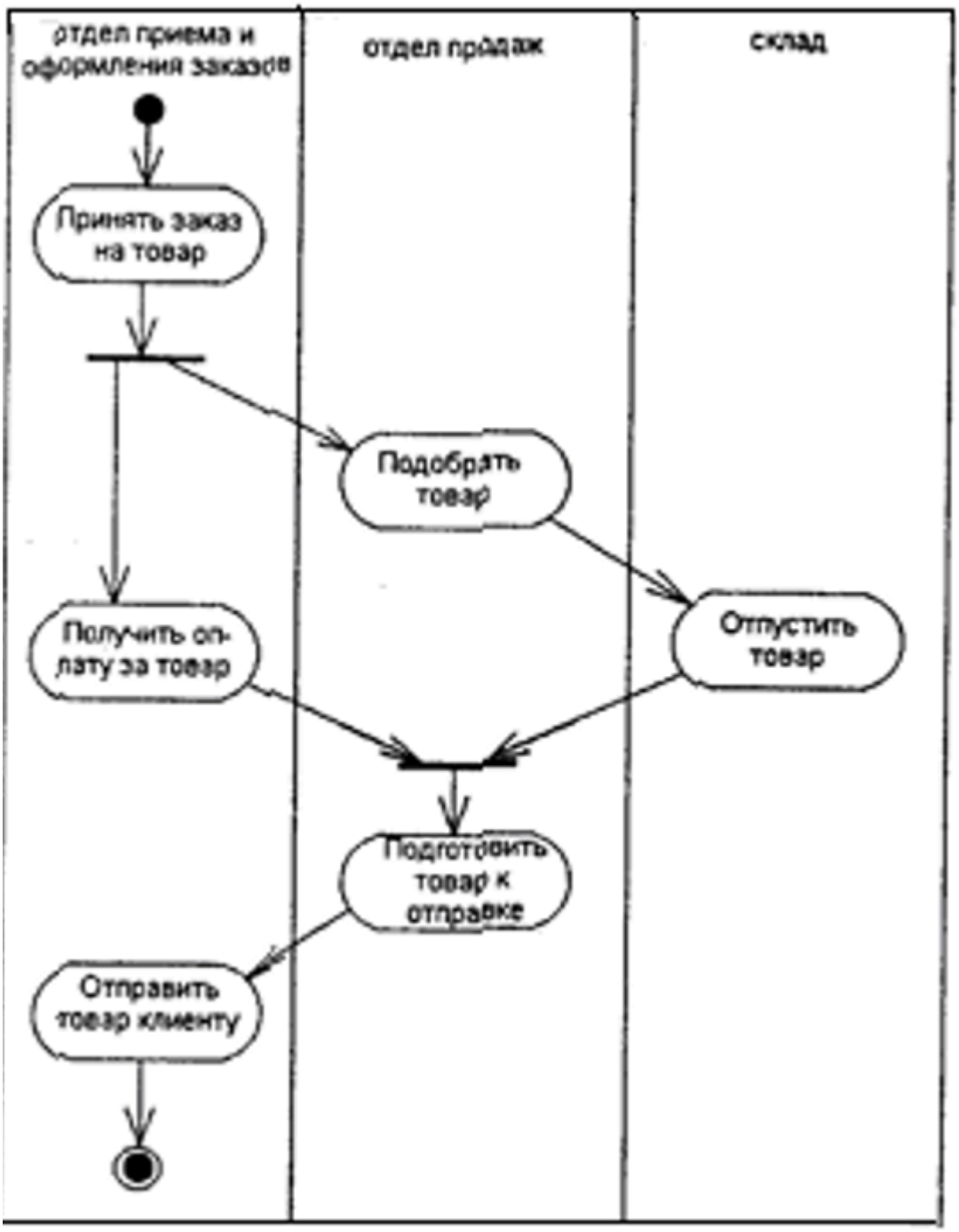


Рис.1 DFD диаграмма

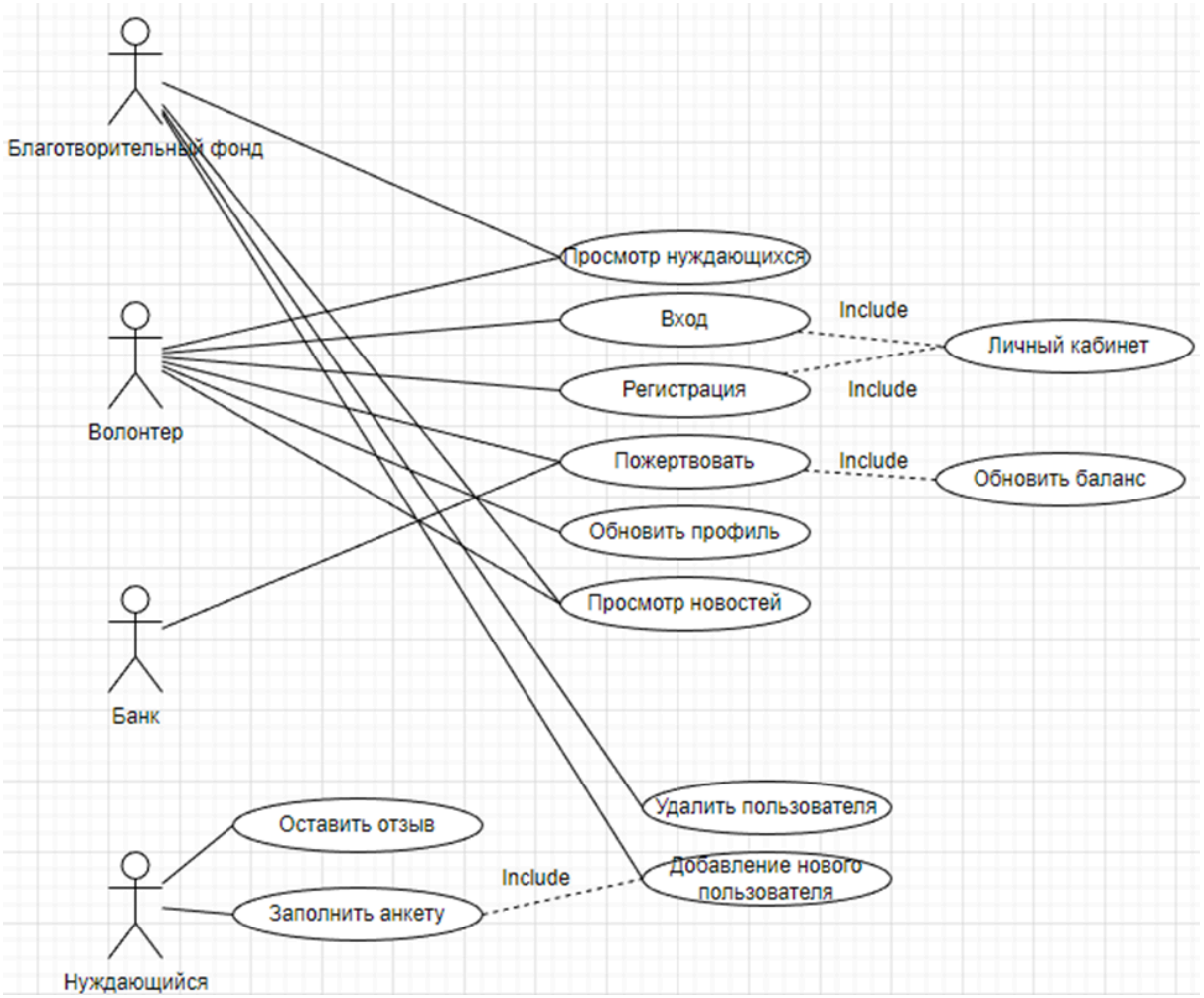


Построить декомпозицию 1 уровня.

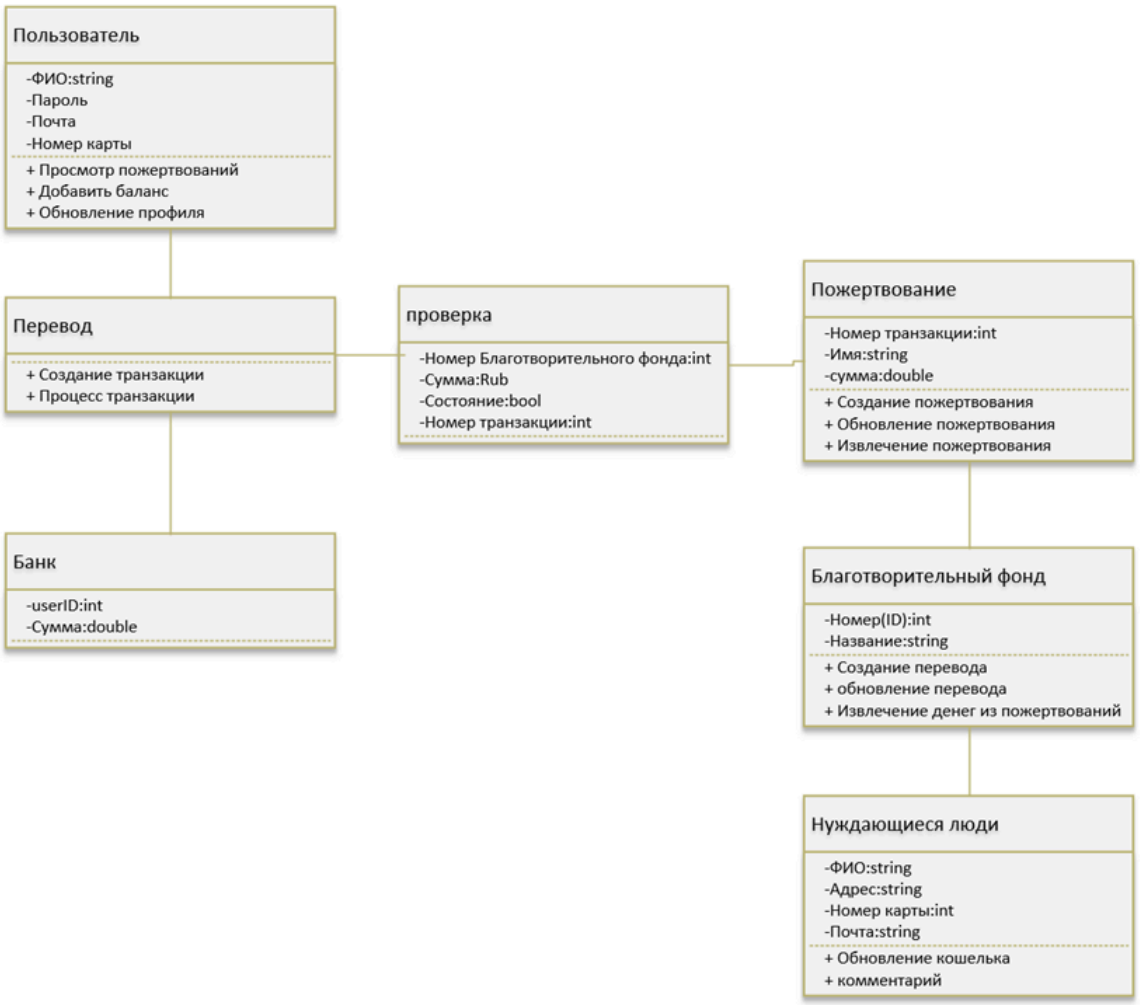
23. Построить диаграмму деятельности для любого процесса по теме …



24. Построить диаграмму вариантов использования по теме …

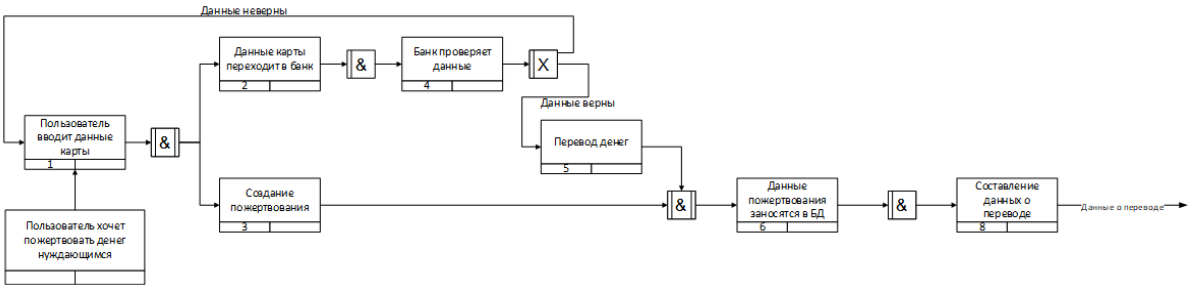


25. Построить диаграмму классов по теме …



26. Построить диаграмму объектов по теме …

27. Построить диаграмму IDEF3 для любого процесса из темы …



28. Построить диаграмму состояний любого объекта, который меняется в течение работы системы, по теме …

29. Построить диаграмму IDEF0 по теме …