Министерство науки и высшего образования
Российской Федерации
Братский педагогический колледж
федерального государственного бюджетного
образовательного учреждения высшего
образования
«Братский государственный университет»

МДК 02.01. ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

методические указания по выполнению лабораторных работ

для студентов IV курса
очной формы обучения
специальности
09.02.07 Информационные системы и программирование

Автор: А.В. Конаков

Братск, 2020

МДК 02.01. Технология разработки программного обеспечения. Методические рекомендации по выполнению лабораторных работ. / Сост. А.В. Конаков. - Братск, 2020. - 90 с.

Содержат указания к выполнению лабораторных работ по дисциплине «Технология разработки программного обеспечения». В лабораторных работах содержатся основные теоретические сведения, касающиеся техники выполнения работ по созданию и дальнейшему сопровождению, программного обеспечения.

Предназначены для студентов специальности 09.02.07 Информационные системы и программирование.

Печатается по решению научно-методического совета
Братского педагогического колледжа ФГБОУ ВО «БрГУ»
665709, г. Братск, ул. Макаренко 40

СОДЕРЖАНИЕ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Целью преподавания дисциплины "Технологии разработки программного обеспечения" является развитие у обучаемых знаний умений и навыков в области выбора, проектирования, реализации, оценки качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях, изучение типовых приемов организации и конструирования пакетов программ сложной структуры, этапов процесса проектирования программного обеспечения, создание прикладных программ с высокой степенью автоматизации управления. В результате изучения дисциплины студент должен освоить основные понятия, методы и технологии, необходимые для решения задач при разработке ПО, уметь применять на практике методы и подходы информационных технологий.

Задачи дисциплины:

- обучение студентов основным подходам к проектированию, разработке и использованию программ;
- дать обучающимся знание технологий разработки программного обеспечения с использованием универсальных языков программирования;
- рассмотреть использование объектно-ориентированного подхода в проектировании программ;
- получение практических навыков использования технологию обобщенного программирования, использования стандартных библиотек классов и шаблонов.
- ознакомить студентов с принципами функционирования и управления специальными средствами WINDOWS–программирования (реализация многозадачности и многопоточности, работа с файловой системой).

Дисциплина «Технологии разработки программного обеспечения» базируется на знаниях, полученных в ходе изучения дисциплин «Структуры и алгоритмы компьютерной обработки данных», «Объектно-ориентированное программирование», «Алгоритмы и технологии параллельного программирования». Знания и умения, практические навыки, приобретенные студентами в результате изучения дисциплины, будут использоваться при выполнении курсовых и дипломных работ.

Тема 1. Основы методологии создания программного обеспечения (ПО)

Модели жизненного цикла. Выбор модели жизненного цикла ПО

1. Цель работы: ознакомиться с понятием жизненного цикла программного продукта, этапами проектирования АСУ, получить практические навыки моделирования функций автоматизированных систем управления для заданной предметной области.

2. Основные теоретические сведения

2.1. Жизненный цикл программного изделия и его критичные этапы

В основе деятельности по созданию и использованию программного обеспечения (ПО) лежит понятие *жизненного цикла* (ЖЦ). ЖЦ является моделью создания и использования ПО, отражающей его различные состояния, начиная с момента возникновения необходимости в данном программном изделии и заканчивая моментом его полного выхода из употребления у всех пользователей.

Традиционно выделяются следующие основные этапы ЖЦ ПО:

- ✓ **Анализ требований,**
- ✓ **Проектирование,**
- ✓ **Кодирование (программирование),**
- ✓ **Тестирование и отладка,**
- ✓ **Эксплуатация и сопровождение.**

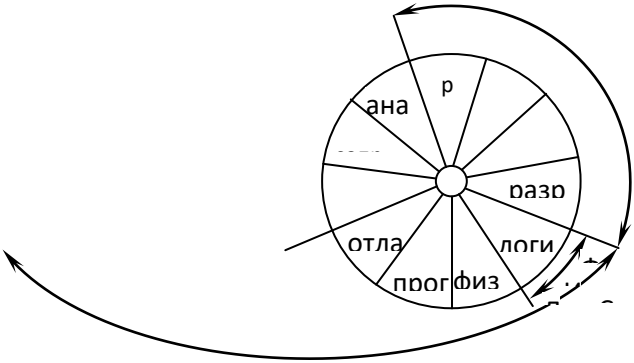
Обобщенно ЖЦ можно представить в виде схемы:

ЖЦ образуется в соответствии с принципом нисходящего проектирования и, как правило, носит итерационный характер: реализованные этапы, начиная с самых ранних, циклически повторяются в соответствии с изменениями требований и внешних условий, введением ограничений и т.п. На каждом этапе ЖЦ порождается определенный набор документов и технических решений, при этом для каждого этапа исходными являются документы и решения, полученные на предыдущем этапе.

Существующие модели ЖЦ определяют порядок исполнения этапов в ходе разработки, а также критерии перехода от этапа к этапу. В соответствии с этим наибольшее распространение получили три следующие *модели ЖЦ*:

1) **КАСКАДНАЯ МОДЕЛЬ** (70-80г.г.) – предполагает переход на следующий этап после полного окончания работ по предыдущему этапу.

2) **ПОЭТАПНАЯ МОДЕЛЬ С ПРОМЕЖУТОЧНЫМ КОНТРОЛЕМ** (80-85 г.г.) – итерационная модель разработки ПО с циклами обратной связи между этапами. Преимущество такой



модели заключается в том, что межэтапные корректировки обеспечивают меньшую трудоемкость по сравнению с каскадной моделью; с другой стороны, время жизни каждого из этапов растягивается на весь период разработки.

3) **СПИРАЛЬНАЯ МОДЕЛЬ** (86-90 г.г.) – делает упор на начальные этапы ЖЦ: анализ требований, проектирование спецификаций, предварительное и детальное проектирование. На этих этапах проверяется и обосновывается реализуемость технических решений путем создания прототипов. Каждый виток спирали соответствует поэтапной модели создания фрагмента или версии программного изделия, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы следующего витка спирали. Таким образом, углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.

АНАЛИЗ ТРЕБОВАНИЙ является первой фазой разработки ПО, на которой требования заказчика уточняются, формализуются и документируются. Фактически на этом этапе дается ответ на вопрос: «Что должна делать будущая система?». Именно здесь лежит ключ к успеху всего проекта. В практике создания больших систем ПО известно немало примеров неудачной реализации проекта именно

из-за неполноты и нечеткости определения системных требований. Список требований к разрабатываемой системе должен включать:

- совокупность условий, при которых предполагается эксплуатировать будущую систему (аппаратные и программные ресурсы, предоставляемые системе; внешние условия ее функционирования; состав людей и работ, имеющих к ней отношение);
- описание выполняемых системой функций;
- ограничения в процессе разработки (директивные сроки завершения отдельных этапов, имеющиеся ресурсы, организационные процедуры и мероприятия, обеспечивающие защиту информации).
- целью анализа является преобразование общих, неясных знаний о требованиях к будущей системе в точные (по возможности) определения.

На этом этапе определяются:

1. архитектура системы, ее функции, внешние условия, распределение функций между аппаратурой и ПО;
2. интерфейсы и распределение функций между человеком и системой;
3. требования к программным и информационным компонентам ПО, необходимые аппаратные ресурсы, требования к базам данных, физические характеристики компонент ПО, их интерфейсы.

ЭТАП ПРОЕКТИРОВАНИЯ дает ответ на вопрос: «**Как (каким образом) система будет удовлетворять предъявленным к ней требованиям?**». Задачей этого этапа является исследование структуры системы и логических взаимосвязей ее элементов, причем здесь не рассматриваются вопросы, связанные с реализацией на конкретной платформе. Проектирование определяется как «(итерационный) процесс получения логической модели системы вместе со строго сформулированными целями, поставленными перед нею, а также написания спецификаций физической системы, удовлетворяющей этим требованиям». Обычно этот этап разделяют на 2 подэтапа:

- *проектирование архитектуры ПО*, включающее разработку структуры и интерфейсов компонент, согласование функций и технических требований к компонентам, методам и стандартам проектирования, производство отчетных документов;

o *детальное проектирование*, включающее разработку спецификаций каждой компоненты, интерфейсов между компонентами, разработку требований к тестам и плана интеграции компонент.

В результате деятельности на этапах анализа и проектирования должен быть получены проект системы, содержащий достаточно информации для реализации системы, содержащий достаточно информации для реализации системы на его основе в рамках бюджета выделенных ресурсов и времени.

2.2. Принципы структурного анализа

Анализ требований разрабатываемой системы является важнейшим среди этапов ЖЦ. Он оказывает существенное влияние на все последующие этапы, являясь в то же время наименее изученным и понятным процессом. На этом этапе, *во-первых*, необходимо понять, что предполагается сделать, а *во-вторых*, задокументировать это, т.к. если требования не зафиксированы и не сделаны доступными для участников проекта, то они вроде бы и не существуют. При этом язык, на котором формулируются требования, должен быть достаточно прост и понятен заказчику.

Конечно, применение известных аналитических методов снимает некоторые из перечисленных проблем анализа, однако эти проблемы могут существенно быть облегчены за счет применения современных структурных методов, среди которых центральное место занимают методологии структурного анализа и проектирования.

Структурным анализом принято называть метод исследования системы, которое начинается с ее общего обзора и затем детализируется, приобретая иерархическую структуру со все большим числом уровней. Для таких методов характерно разбиение на уровни абстракции с ограничением числа элементов на каждом из уровней (обычно от 3 до 6-7); ограниченный контекст, включающий лишь существенные на каждом уровне детали; использование строгих формальных правил записи; последовательное приближение к конечному результату.

Все методологии структурного анализа базируются на ряде общих принципов, часть из которых регламентирует организацию работ на начальных этапах ЖЦ, а часть используется при выработке рекомендаций по организации работ. В качестве базовых принципов используются: **принцип «разделяй и властвуй»** и **принцип**

иерархического упорядочивания. Первый является принципом решения трудных проблем путем разбиения их на множество меньших независимых подзадач, легких для понимания и решения. Второй принцип в дополнение к тому, что легче понимать проблему, если она разбита на части, декларирует, что устройство этих частей также существенно для понимания. Понимаемость проблемы резко повышается при организации ее частей в древовидные иерархические структуры, т.е. система может быть понята и построена по уровням, каждый из которых добавляет новые детали. Выделение двух базовых принципов инженерии программного обеспечения вовсе не означает, что остальные принципы являются второстепенными, игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и неуспеху всего проекта). Перечислим основные из этих принципов: *принцип абстрагирования, принцип формализации, принцип упрятывания, принцип концептуальной общности, принцип полноты, принцип непротиворечивости, принцип логической независимости, принцип независимости данных, принцип структурирования данных, принцип доступа конечного пользователя.*

2.3. Средства структурного анализа и проектирования

Для целей моделирования систем вообще и структурного анализа в частности используются 3 группы средств, иллюстрирующих:

- функции, которые система должна выполнять;
- отношения между данными;
- зависящее от времени поведение системы (аспекты реального времени).

Среди многообразия средств решения данных задач в методологиях структурного анализа наиболее часто и эффективно применяются следующие:

Функциональные модели: SADT (IDEF0)-модели (реализуются с использованием пакетов IDEF, AllFusion Process Modeler).

Информационные модели: ERD (IDEF1X) – пакеты IDEF, AllFusion ERWin Data Modeler.

Динамические модели: IDEF/CPN – пакет IDEF, IDEF3 – пакет AllFusion Process Modeler.

2.4. Основные сведения по методологии IDEF0

Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Цель моделирования (Purpose). Модель не может быть построена без четко сформулированной цели. Пример цели: «Описать функциональность предприятия с целью написания спецификаций ИС».

Точка зрения (Viewpoint). Точку зрения можно представить как взгляд человека, который видит систему в нужном для моделирования аспекте. Как правило, выбирается точка зрения человека, ответственного за моделируемую работу в целом. Цель и точка зрения документируются.

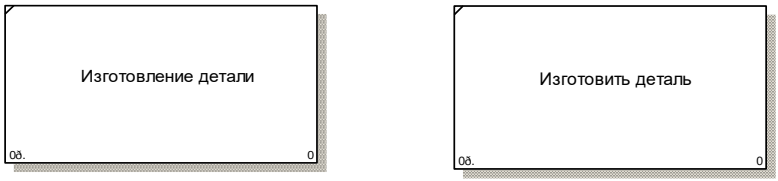
Основные элементы IDEF0-модели

В основе методологии IDEF0 лежат 4 основных понятия:

- функциональный блок;
- интерфейсная дуга (стрелка);
- декомпозиция;
- глоссарий.

1. Функциональный блок

Функциональные блоки обозначают *поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты*. Графически функциональные блоки *изображаются в виде прямоугольников*. Все блоки должны быть названы и определены. *Имя функционального блока* должно быть выражено сочетанием *отглагольного существительного, обозначающего процесс, или глаголом* (рис. 2.1):



а) б)

Рисунок 2.1 – Примеры работ

Определение функционального блока заносится в глоссарий или словарь работ (Activity Dictionary).

Все функциональные блоки модели нумеруются. Номер состоит из префикса и числа. Может использоваться префикс любой

длины, но обычно используется префикс А. Контекстная (корневая) работа (функциональный блок) имеет номер А0.

2. Интерфейсная дуга (стрелка - Arrow)

Взаимодействие функциональных блоков с внешним миром и между собой описывается в виде интерфейсных дуг (стрелок). Стрелки представляют собой *некую информацию и обозначаются существительными* (например, «Заготовка», «Изделие») или *именуемыми сочетаниями* (например, «Готовое изделие»). Все стрелки должны быть определены. Определения заносятся в словарь стрелок – глоссарий (Arrow Dictionary).

В IDEF0 различают 4 типа стрелок (рис.2.2).

Каждая стрелка имеет свое расположение относительно функционального блока.

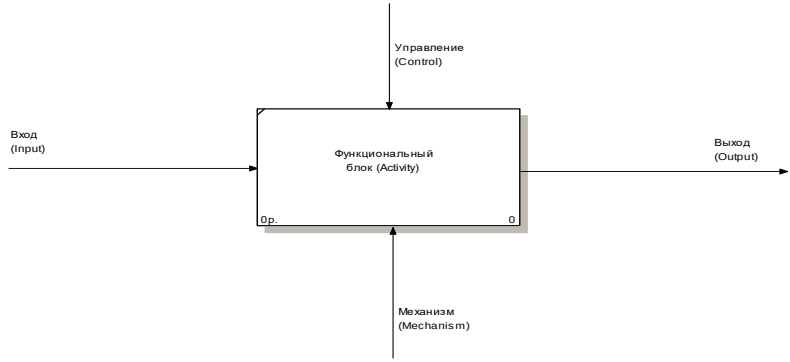


Рисунок 2.2 – Типы стрелок

Вход (Input) – материал или информация, которые используются или преобразуются работой для получения результата (выхода). Стрелка *Input* рисуется входящей в левую грань работы.

Управление (Control) – правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Рисуется как входящая в верхнюю грань работы.

Выход (Output) – материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Работа без результата не имеет смысла и не должна моделироваться. Изображается исходящей из правой грани работы.

Механизм (Mechanism) – ресурсы, которые выполняют работу, например, персонал предприятия, станки, устройства и т.д. Рисуется как входящая в нижнюю грань работы.

3. Глоссарий *– набор определений, ключевых слов* и т.д., *которые характеризуют каждый объект модели*.

4. Декомпозиция – это *разбиение системы на крупные фрагменты – функции, функции – на подфункции и т.д. до конкретных процедур.*

Модель может содержать 4 типа диаграмм:

- контекстную (в каждой модели может быть только 1 контекстная диаграмма);
- декомпозиции;
- дерева узлов;
- только для экспозиции (FEO).

Контекстная диаграмма является вершиной древовидной структуры диаграмм и *представляет собой общее описание системы и ее взаимодействия с внешней средой*.

После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов – *диаграммами декомпозиции*. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и т.д., до достижения нужного уровня подробности описания.

Диаграмма дерева узлов показывает *иерархическую зависимость работ*, но не взаимосвязи между работами.

Диаграммы для экспозиции (FEO) строятся для *иллюстрации отдельных фрагментов модели,* для иллюстрации альтернативной точки зрения либо для специальных целей.

Все диаграммы имеют нумерацию. Контекстная диаграмма имеет номер A-0, декомпозиция контекстной диаграммы – номер A), остальные диаграммы-декомпозиции – номера по соответствующему узлу (например, A1, A2, A21 и т.д.).

3. Методика выполнения лабораторной работы

В качестве примера рассматривается процесс выполнения студентом курсовой работы (курсового проекта).

3.1. Создание контекстной диаграммы.

1. Запустите AllFusion Process Modeler.
2. Если появится диалог ModelMart Connection Manager, нажмите кнопку Cancel.
3. *Заполнение реквизитов мастерской страницы*. Щелкните по кнопке  или воспользуйтесь меню (File→New). Появится диалог

I would like to.. Введите имя модели: «Курсовая работа» и выберите Type – Business Process (IDEF0). Нажмите OK. Появится окно Properties for new Models. Во вкладке General необходимо ввести имя автора модели, инициалы. Нажмите кнопку OK.

Автоматически создается контекстная диаграмма.

Перейдите в меню Model/Model Properties. Во вкладке General диалога Model Properties следует внести имя проекта «Модель выполнения курсовой работы», а также тип модели – Time Frame: AS-IS.

Во вкладке Definition внесите определение «Это учебная модель, описывающая процесс выполнения студентом курсовой работы» и цель Scope: «Изучение процесса и выявление недостатков».

На вкладке Status отметьте галочкой Working.

Нажатием кнопки OK закройте диалоговое окно Model Properties. При этом автоматически будет создана мастерская страница.

4. *Определение цели и точки зрения.* Перейдите в меню Model/Model Properties. Во вкладке Purpose внесите цель – «Purpose: Моделировать процесс выполнения курсовой работы» и точку зрения – «Viewpoint: Студент».

С помощью кнопки T внесите текст в поле диаграммы – точку зрения и цель (рисунок 3.1).

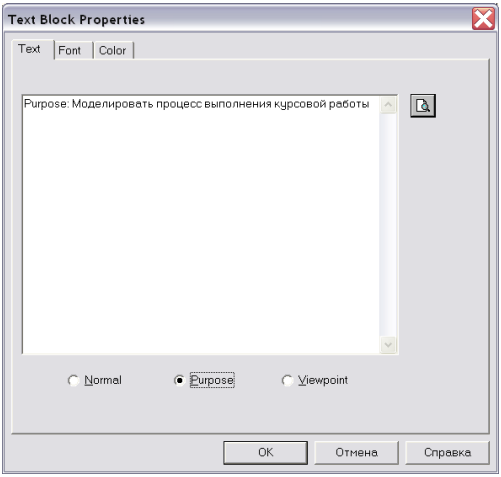


Рисунок 3.1 – Внесение текста в поле диаграммы с помощью редактора Text Block Editor

5. *Определение настроек параметров текста*

Для правильного отображения элементов модели, необходимо задать настройки шрифта, которые будут использоваться по умолчанию. Для этого выберите пункт меню Model/ Default Fonts (рис. 3.2) и во всплывающем меню для всех пунктов установите следующие параметры: Font: Arial, Size: 11, Script: Кириллический. Проставьте галочку в пункте Change all occurrences of this font in the model.

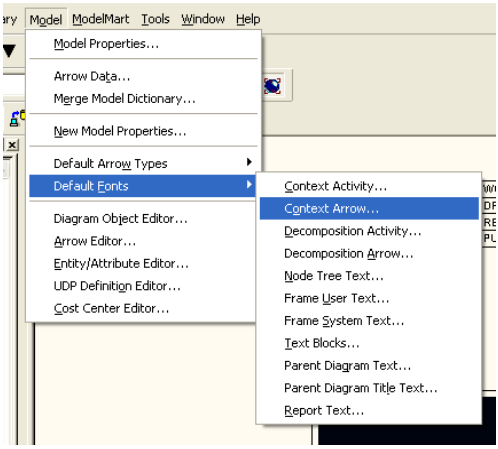


Рисунок 3.2 – Настройка параметров текста

При внесении элементов на диаграммы модели настройки шрифта можно изменить нажатием правой кнопки мыши и выбором пункта Font в контекстном меню.

6. Обратите внимание на кнопку  на панели инструментов. Эта кнопка включает и выключает инструмент просмотра и навигации – Model Explorer (появляется слева). Model Explorer имеет 3 вкладки – Activities, Diagrams, Objects. Во вкладке Activities щелчок правой кнопкой по объекту позволяет редактировать его свойства.

7. Перейдите на контекстную диаграмму и правой кнопкой мыши щелкните по работе. В контекстном меню выберите Name. Во вкладку Name внесите имя: «Выполнить курсовую работу».

8. Во вкладке Definition внесите определение: «Текущие процессы выполнения курсовой работы».

9. Нажатием копки  создайте стрелки на контекстной диаграмме (Таблица 3.1).

Таблица 3.1 – Стрелки контекстной диаграммы

Имя стрелки (Arrow Name)	***Определение стрелки (Arrow Definition)***	***Тип стрелки (Arrow Type)***
График	График консультаций и сроки сдачи	Input
Список литературы	Источники информации для выполнения курсовой работы	Input
Варианты заданий	Список заданий на курсовую работу, подлежащий распределению между студентами	Input
Методические указания	Документ, содержащий указания по выполнению курсовой работы, описывающий содержание ее частей и основные требования	Control
Положение о курсовом проектировании	Документ, отражающий организационные требования по выполнению и сдаче курсовой работы	Control
Курсовая работа	Документ, являющийся основанием для получения оценки	Output
Оценка за курсовую работу	Результат выполнения курсовой работы	Output
Студент	Тот, кто выполняет курсовую работу	Mechanism

10. Результат выполнения предыдущих пунктов представлен на рисунке 3.3.

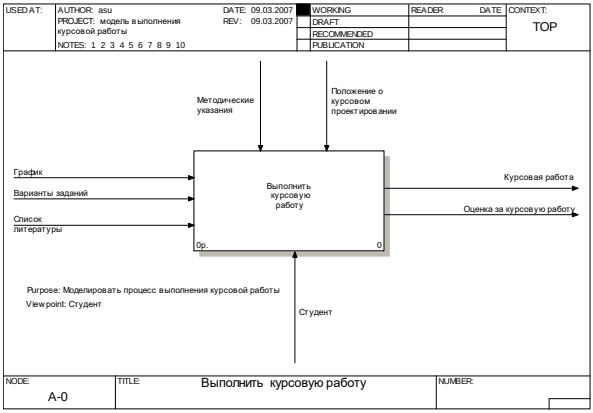


Рисунок 3.3 – Контекстная диаграмма

3.2. Создание диаграммы декомпозиции

1. Выберите кнопку перехода на нижний уровень на панели инструментов ▼ (или щелкните правой кнопкой мыши по работе в Model Explorer на вкладке Activities, выберите в контекстном меню Decompose) и в диалоге Activity Box Count установите число работ на диаграмме нижнего уровня – 7 (рис. 3.4). Нажмите кнопку OK.

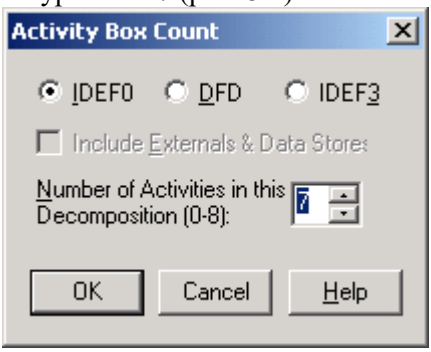


Рисунок 3.4 – Диалог Activity Box Count

2. Автоматически будет создана диаграмма декомпозиции (рис. 3.5).

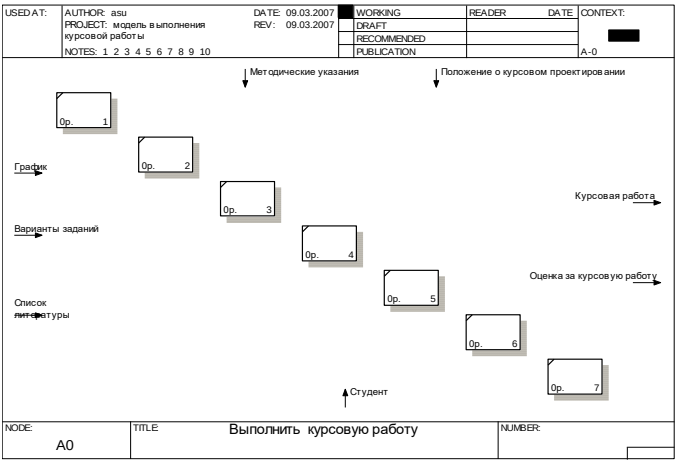


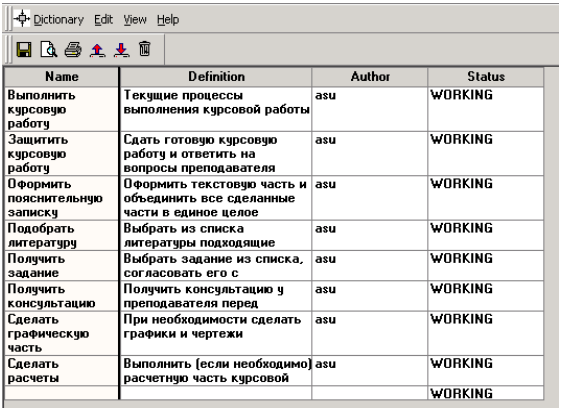
Рисунок 3.5 – Диаграмма декомпозиции

Правой кнопкой мыши щелкните по работе, выберите Name, внесите имя работы и определение. Повторите операцию для всех 7 работ согласно таблице 3.2.

Таблица 3.2 – Работы диаграммы декомпозиции A0

Имя работы (Activity Name)	**Определение (Definition)**
Получить задание	Выбрать задание из списка, согласовать его с преподавателем
Подобрать литературу	Выбрать из списка литературы подходящие источники
Сделать расчеты	Выполнить (если необходимо) расчетную часть курсовой работы согласно заданию
Сделать графическую часть	При необходимости сделать графики и чертежи
Оформить пояснительную записку	Оформить текстовую часть и объединить все сделанные части в единое целое
Получить консультацию	Получить консультацию у преподавателя перед защитой, выявить неточности и недостатки
Защитить курсовую работу	Сдать готовую курсовую работу и ответить на вопросы преподавателя

3. Просмотреть свойства работ и внести изменения можно, воспользовавшись словарем работ. Вызов словаря – меню Dictionary/Activity (рис. 3.6).



Name	Definition	Author	Status
Выполнить курсовую работу	Текущие процессы выполнения курсовой работы	asu	WORKING
Защитить курсовую работу	Сдать готовую курсовую работу и ответить на вопросы преподавателя	asu	WORKING
Оформить пояснительную записку	Оформить текстовую часть и объединить все сделанные части в единое целое	asu	WORKING
Подобрать литературу	Выбрать из списка литературы подходящие	asu	WORKING
Получить задание	Выбрать задание из списка, согласовать его с	asu	WORKING
Получить консультацию	Получить консультацию у преподавателя перед	asu	WORKING
Сделать графическую часть	При необходимости сделать графики и чертежи	asu	WORKING
Сделать расчеты	Выполнить (если необходимо) расчетную часть курсовой	asu	WORKING
			WORKING

Рисунок 3.6 – Словарь Activity Dictionary

4. Добавить функциональный блок на диаграмму декомпозиции можно нажатием кнопки  . Удалить работу можно нажатием клавиши Del, предварительно выделив ее. Нумерация работ изменится автоматически.

Примечание. Если работа удаляется из диаграммы, из словаря она не удаляется. Имя и описание такой работы может быть использовано в дальнейшем. Для удаления всех имен работ, не использующихся в модели, щелкните по кнопке  (см. рис. 3.6).

5. Для связывания граничных, не связанных с работами стрелок (рис. 3.5) необходимо щелкнуть по наконечнику стрелки и по соответствующему сегменту работы. Не связанные стрелки рассматриваются как синтаксическая ошибка. Свяжите граничные стрелки с работами, как показано на рисунке 3.7.

Разветвление стрелок. Расписание необходимо для того, чтобы прийти на консультацию и на защиту, т.е. необходимо подвести одноименную стрелку к 2 работам. Для разветвления стрелки необходимо *в режиме редактирования* стрелок щелкнуть по фрагменту стрелки и по соответствующему сегменту работы.

Слияние стрелок. Для слияния двух стрелок выхода необходимо *в режиме редактирования* стрелок щелкнуть по

сегменту выхода работы, а затем по соответствующему фрагменту стрелки.

6. Чтобы сделать видимыми ICOM-метки (I1, I2, C1, C2, M1, O1) граничных стрелок, воспользуйтесь пунктом меню Model/Model Properties. В закладке Display необходимо отметить галочкой ICOM codes.

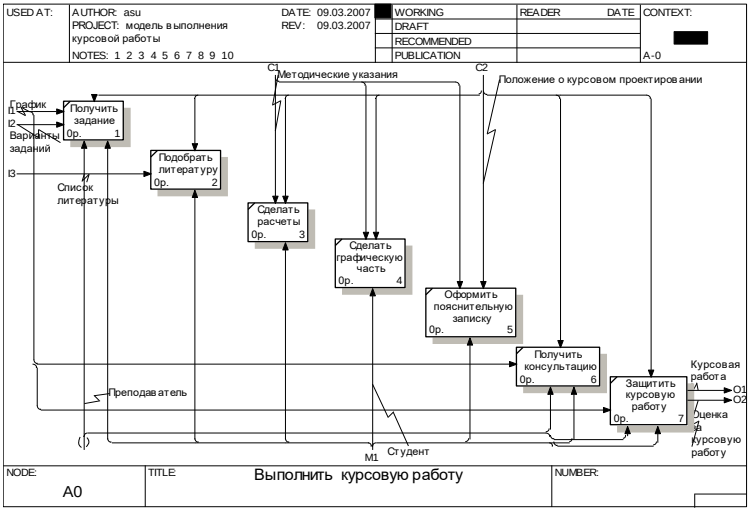


Рисунок 3.7 – Связанные граничные стрелки на диаграмме декомпозиции A0

6. Создайте новую граничную стрелку механизма «Преподаватель» для работ «Получить задание», «Получить консультацию», «Получить консультацию», «Защитить курсовую работу» (см. рис. 3.7). Эта запрещенная стрелка, она автоматически не попадает на диаграмму верхнего уровня и имеет квадратные скобки на конце.

Туннелирование стрелок. Щелкните правой кнопкой мыши по квадратным скобкам и выберите пункт меню Arrow Tunnel. В появившемся диалоговом окне выберите пункт Change it to resolved rounded tunnel. Стрелка станет туннельной (с круглыми скобками на конце). Это означает, что она будет присутствовать только на данной диаграмме. Если вы выберете пункт меню Resolve it to border arrow, стрелка появится на всех диаграммах модели.

7. Создайте внутренние стрелки, как это показано на рисунке 3.8 и дайте им определения (см. табл. 3.3).

Результат выполнения предыдущих пунктов представлен на рисунке (рис. 3.8).

Таблица 3.3 – Внутренние стрелки диаграммы декомпозиции

Название стрелки (Arrow Name)	**Определение (Definition)**
Задание	Выдается на консультации преподавателем, чтобы студенты могли выбрать самостоятельно, или назначается в зависимости от варианта
Литература	Выбранные источники, необходимые для выполнения курсовой работы
Расчеты	Выполненная расчетная часть курсовой работы
Графическая часть	Выполненная графическая часть курсовой работы
Пояснительная записка	Теоретическая часть + расчеты + графическая часть, оформленные в соответствии ГОСТ
Замечания, дополнения	Замечания преподавателя, полученные на консультации, которые необходимо исправить до защиты

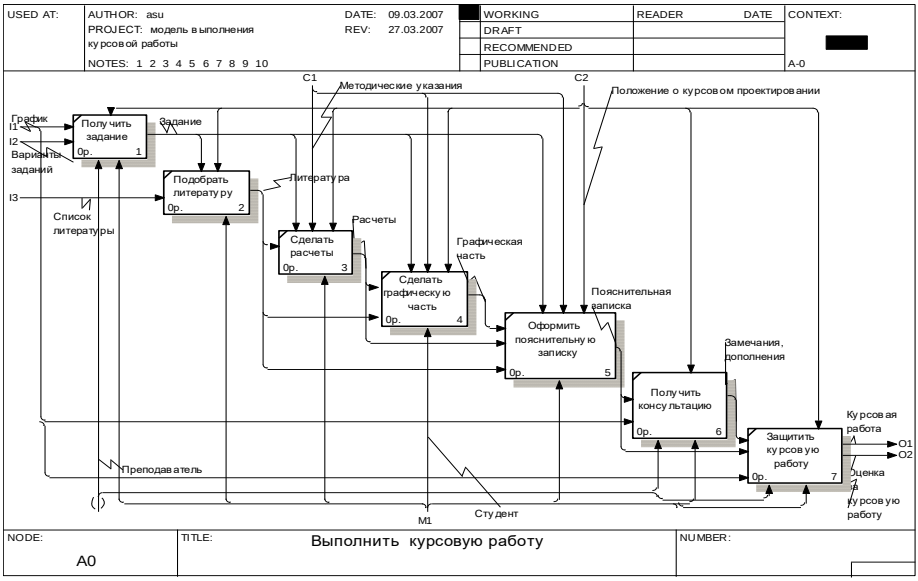


Рисунок 3.8 – Диаграмма декомпозиция блока A0

3.3. Создание дерева узлов

1. Выберите меню Diagram/Add Node Tree. В первом диалоге Node Tree Wizard внесите имя диаграммы, укажите диаграмму корня дерева и количество уровней (рис. 3.9). Нажмите кнопку «Далее».

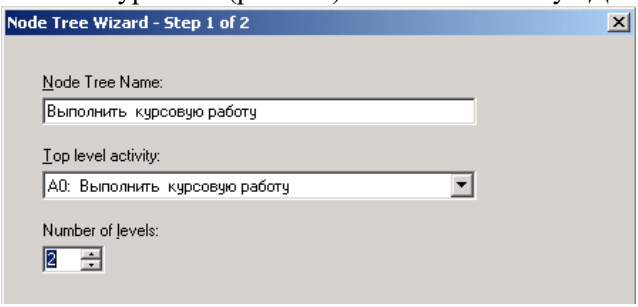


Рисунок 3.9 – Первый диалог Node Tree Wizard

2. Во втором диалоге все параметры можно оставить без изменения, данные опции описывают внешний вид дерева (рис. 3.10):

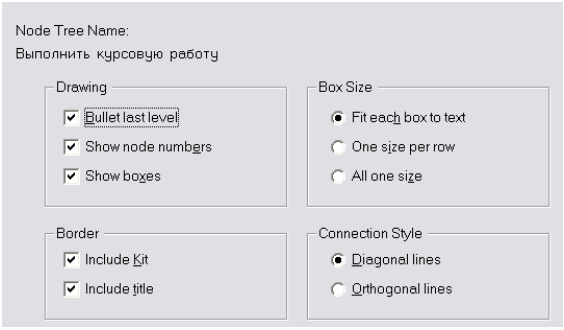


Рисунок 3.10 – Параметры настройки диаграммы Node Tree

3. Нажмите кнопку «Готово».

4. Автоматически будет сформирована диаграмма, представленная на рисунке 3.11. Просмотреть и отредактировать Дерево узлов можно на вкладке Diagrams в Model Explorer.

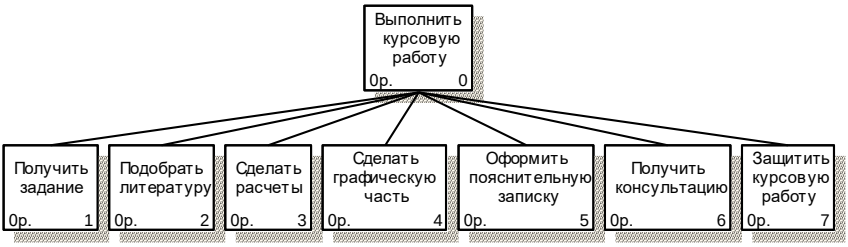
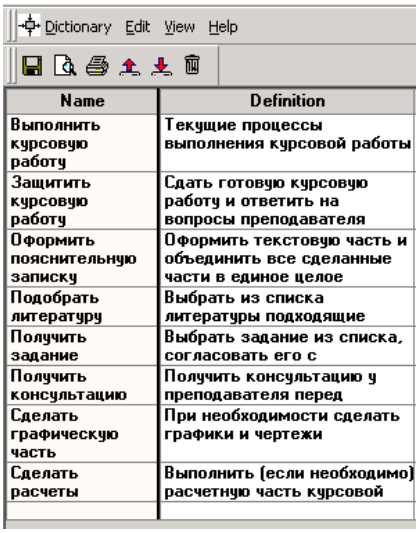


Рисунок 3.11 – Диаграмма узлов

3.4. Создание глоссария

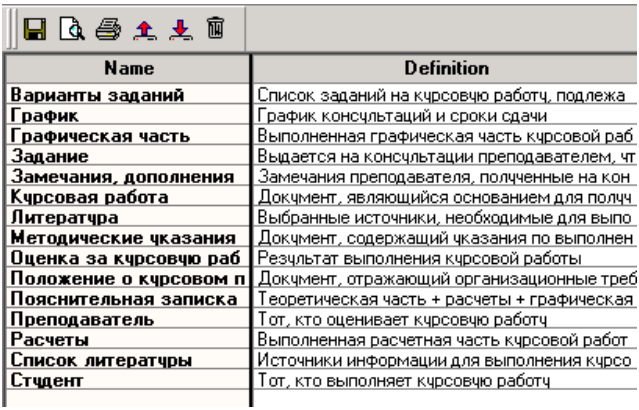
Вызов словаря работ: меню Dictionary/ Activity. Заголовки столбцов таблицы словаря можно отредактировать (меню View/ Customize) – добавив или удалив столбцы. Словарь работам представлен на рисунке 3.12.



Name	Definition
Выполнить курсовую работу	Текущие процессы выполнения курсовой работы
Защитить курсовую работу	Сдать готовую курсовую работу и ответить на вопросы преподавателя
Оформить пояснительную записку	Оформить текстовую часть и объединить все сделанные части в единое целое
Подобрать литературу	Выбрать из списка литературы подходящие
Получить задание	Выбрать задание из списка, согласовать его с
Получить консультацию	Получить консультацию у преподавателя перед
Сделать графическую часть	При необходимости сделать графики и чертежи
Сделать расчеты	Выполнить (если необходимо) расчетную часть курсовой

Рисунок 3.12 – Словарь работ

Вызов словаря стрелок: меню Dictionary/ Arrow (рис. 3.13).



Name	Definition
Варианты заданий	Список заданий на курсовую работу, подлежа
График	График консультаций и сроки сдачи
Графическая часть	Выполненная графическая часть курсовой раб
Задание	Выдается на консультации преподавателем, чт
Замечания, дополнения	Замечания преподавателя, полученные на кон
Курсовая работа	Документ, являющийся основанием для получ
Литература	Выбранные источники, необходимые для выпо
Методические указания	Документ, содержащий указания по выполнен
Оценка за курсовую раб	Результат выполнения курсовой работы
Положение о курсовом п	Документ, отражающий организационные треб
Пояснительная записка	Теоретическая часть + расчеты + графическая
Преподаватель	Тот, кто оценивает курсовую работу
Расчеты	Выполненная расчетная часть курсовой работ
Список литературы	Источники информации для выполнения курсо
Студент	Тот, кто выполняет курсовую работу

Рисунок 3.13 – Словарь стрелок

4. Задание

Создать функциональную модель предметной области с использованием пакета AllFusion Process Modeler в соответствии с вариантом задания. Вариант задания определяет преподаватель. Функциональная модель должна содержать 3 уровня декомпозиции.

5. Порядок выполнения работы

Для выполнения работы необходимо:

1) повторить правила техники безопасности при работе с вычислительной техникой;

2) изучить теоретическую часть настоящего методического указания;

3) получить у преподавателя вариант задания;

4) выполнить лабораторную работу согласно описанной в пункте 3 методике;

5) в соответствии с требованиями, приведенными в разделе 6, оформить отчет по лабораторной работе;

6) защитить лабораторную работу.

6. Требования к отчету

Отчет по лабораторной работе должен содержать:

а) титульный лист;

б) название лабораторной работы, цель работы;

в) основные определения;

г) краткое описание предметной области;

д) функциональную модель (контекстная диаграмма, диаграммы декомпозиции, дерево узлов) предметной области в соответствии с вариантом задания;

е) выводы по проделанной работе.

7. Контрольные вопросы

1. Каковы цели функционального моделирования?
2. Чем модель отличается от диаграммы?
3. Назовите основные компоненты функциональной модели.
4. Какие виды интерфейсных дуг различают в IDEF0?
5. Для чего нужна цель и точка зрения?
6. Что такое функциональный блок?
7. Что такое глоссарий? Каким образом он формируется в AllFusion Process Modeler?
8. Какие виды отчетов по модели можно сформировать в AllFusion Process Modeler?
9. Какие виды диаграмм может содержать функциональная модель?
10. Что представляет собой туннельная стрелка?
11. Дайте определение ЖЦ ПО.
12. Перечислите модели ЖЦ. В чем их отличие.
13. Назовите основные стадии ЖЦ ПО.
14. Назовите принципы, лежащие в основе структурного анализа и проектирования.

Тема 2. Структурный подход к разработке ПО

Реализация основ структурного программирования в языках программирования.

***Цель работы*:** составить и проанализировать требования к программе и разработать техническое задание на разработку программного средства.

Подготовка к лабораторной работе

Ознакомиться с лекционным материалом по теме «Модели ЖЦ ПО. Этапы ЖЦ в соответствии с ГОСТ 19.102-77. Постановка задачи» учебной дисциплины «Разработка и стандартизация ПС и ИТ».

Изучить соответствующие разделы в изданиях [1 - 3].

Теоретическая часть. Разработка технического задания

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.

1. Общие положения

1.1.Техническое задание оформляют в соответствии с ГОСТ 19.106—78 на листах формата А4 и А3 по ГОСТ 2.301—68, как правило, без заполнения полей листа. Номера листов (страниц) проставляют в верхней части листа над текстом.

1.2.Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 19.104—78. Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать.

1.3.Для внесения изменений и дополнений в техническое задние на последующих стадиях разработки программы или программного изделия выпускают дополнение к нему. Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

1.4.Техническое задание должно содержать следующие разделы:

•введение;

•наименование и область применения;

•основание для разработки;

•назначение разработки;

•технические требования к программе или программному изделию;

•технико-экономические показатели;

•стадии и этапы разработки;

•порядок контроля и приемки;

•приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них. При необходимости допускается в техническое задание включать приложения.

2. Содержание разделов

2.1.Введение должно включать краткую характеристику области применения программы или программного продукта, а также объекта (например, системы), в котором предполагается их использовать. Основное назначение введения — продемонстрировать актуальность данной разработки и показать, какое место эта разработка занимает в ряду подобных.

2.2.В разделе «Наименование и область применения» указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие.

2.3.В разделе «Основание для разработки» должны быть указаны:

• документ (документы), на основании которых ведется разработка. Таким документом может служить план, приказ, договор и т. п.;

• организация, утвердившая этот документ, и дата его утверждения;

• наименование и (или) условное обозначение темы разработки.

2.4. В разделе «Назначение разработки» должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.

2.5. Раздел «Технические требования к программе или программному изделию» должен содержать следующие подразделы:

•требования к функциональным характеристикам;

•требования к надежности;

•условия эксплуатации;

•требования к составу и параметрам технических средств;
•требования к информационной и программной совместимости;
•требования к маркировке и упаковке;
•требования к транспортированию и хранению;
•специальные требования.

2.5.1.В подразделе «Требования к функциональным характеристикам» должны быть указаны требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т. п.

2.5.2.В подразделе «Требования к надежности» должны быть указаны требования к обеспечению надежного функционирования (обеспечение устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа и т. п.).

2.5.3.В подразделе «Условия эксплуатации» должны быть указаны условия эксплуатации (температура окружающего воздуха, относительная влажность и т. п. для выбранных типов носителей данных), при которых должны обеспечиваться заданные характеристики, а также вид обслуживания, необходимое количество и квалификация персонала.

2.5.4.В подразделе «Требования к составу и параметрам технических средств» указывают необходимый состав технических средств с указанием их технических характеристик.

2.5.5.В подразделе «Требования к информационной и программной совместимости должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования. При необходимости должна обеспечиваться защита информации и программ.

2.5.6.В подразделе «Требования к маркировке и упаковке» в общем случае указывают требования к маркировке программного изделия, варианты и способы упаковки.

2.5.7.В подразделе «Требования к транспортированию и хранению» должны быть указаны для программного изделия условия транспортирования, места хранения, условия хранения, условия складирования, сроки хранения в различных условиях.

2.5.8. В разделе «Технико-экономические показатели» должны быть указаны: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

2.6.В разделе «Стадии и этапы разработки» устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а также как правило, сроки разработки и определяют исполнителей.
2.7.В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.
2.8.В приложениях к техническому заданию при необходимости приводят:
•перечень научно-исследовательских и других работ, обосновывающих разработку;
•схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые могут быть использованы при разработке;
•другие источники разработки.

В случаях, если какие-либо требования, предусмотренные техническим заданием, заказчик не предъявляет, следует в соответствующем месте указать «Требования не предъявляются».

Примеры разработки технического задания приведены в приложениях Б и В.

Порядок выполнения работы

1. Разработать техническое задание на программный продукт согласно своему варианту (см. варианты в приложении А) в соответствии с ГОСТ 19.106-78. При разработке технического задания не ограничиваться требованиями, приведенными условии задачи приложения А, добавить своих требования , выработанные на предыдущем этапе после анализа бизнес-модели.
2. Оформить отчет по лабораторной работе.
3. Представить отчет по лабораторной работе для защиты.

Требования к результатам выполнения лабораторной работы

При формировании технического задания обратить внимание на

- Требования для пункта 2.5.1 –это набор пользовательских требований четко описывающий функционал разрабатываемого программного средства (не мене 20) (п 2.5.1)

- Требования для пунктов 2.5.2-2.5.5 –это нефункциональные требования к структуре и эксплуатации программного средства.
- Требования для пунктов 2.5.6-2.5.8. технического задания не формируются
- Требования для пункта 2.6 представляются в виде диаграммы Ганта.
- Требования для пункта 2.7 формируются обобщенно и будут уточнены в процессе разработки.

Защита отчета по лабораторной работе

Отчет по лабораторной работе должен быть оформлен согласно требований СТО ВГУЭС и состоять из следующих структурных элементов:

- титульный лист;
- текстовая часть;
- приложение: разработанное технического задания на программное средство.

Текстовая часть отчета должна включать пункты:

- условие задачи;
- порядок выполнения.

Зашита отчета по лабораторной работе заключается в предъявлении преподавателю полученных результатов в виде файла и демонстрации полученных навыков при ответах на вопросы преподавателя.

Контрольные вопросы

1. Что такое жизненный цикл программного продукта?
2. Дайте определение модели жизненного цикла ПО.
3. Приведите этапы разработки программного средства.
4. Какие этапы включает в себя модель ЖЦ ПС согласно ГОСТ 19.102-77?
5. Что включает в себя этап предпроектного исследования?
6. Перечислите функциональные требования к программному продукту.
7. Перечислите эксплуатационные требования к программному продукту.
8. Перечислите правила разработки технического задания.
9. Назовите основные разделы технического задания.

В каких отношениях находятся заказчик и разработчик при выработке требований к программному средству?

Графическое представление структурированных схем алгоритмов.

Краткие теоретические сведения

Алгоритмом называется точное предписание, определяющее последовательность действий исполнителя, направленных на решение поставленной задачи. В роли исполнителей алгоритмов могут выступать люди, роботы, компьютеры.

Понятие алгоритма в программировании является фундаментальным. Для алгоритма важен не только набор определенных действий, но и то, как они организованы, т.е. в каком порядке они выполняются.

Свойства алгоритма:

- ***понятность*** – все действия должны входить в систему команд исполнителя, т.е. быть понятны ему;
- ***дискретность*** - алгоритм делится на отдельные элементарные шаги;
- ***определенность*** - каждая команда однозначно определяет действие исполнителя;
- ***конечность(результативность***) - алгоритм должен завершаться за конечное число шагов.
- ***массовость*** – алгоритм позволяет решать целый класс похожих задач.

Способы записи алгоритма:

1. Словесно-формульный

Пример.

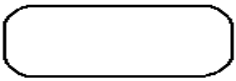
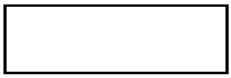
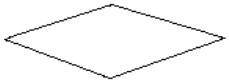
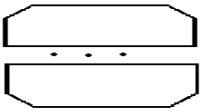
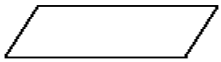
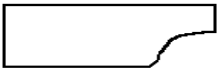
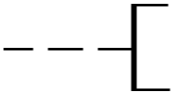
Алгоритм деления обыкновенных дробей

1. Числитель первой дроби умножить на знаменатель второй;
2. Знаменатель второй дроби умножить на числитель второй;
3. Записать дробь, числитель которой есть результат выполнения пункта 1, а знаменатель - результат выполнения пункта 2.

2. Графический способ (в виде блок-схемы *)*

Блок схема – это графическое представление алгоритма при помощи стандартных обозначений. Блок схемы составляются в соответствии с ГОСТами. ГОСТы алгоритмов: ГОСТ 19.002-80, ГОСТ 19.003-80. На схемах алгоритмов выполняемые действия изображаются в виде отдельных блоков, которые соединяются между собой линиями связи в порядке выполнения действий. На линиях связи могут ставиться стрелки, причем, если направление связи *слева направо или сверху вниз*, то стрелки *не ставятся*. Блоки нумеруются. Внутри блока дается информация о выполняемых действиях.

Таблица 1 – Основные блоки, используемые при составлении алгоритмов

Название	**Обозначение**	**Назначение**
Пуск, Останов		Начало-конец алгоритма
Процесс		Любое вычислительное действие
Решение		Проверка условия
Модификатор		Цикл
Ввод-вывод		Ввод-вывод данных
Документ		Вывод на печатающее устройство
Соединитель		Используется на линиях разрыва
Комментарий		Комментарий

3. Запись алгоритма в виде последовательности команд для ЭВМ

Алгоритм, записанный на одном из языков программирования называется ***программой***.

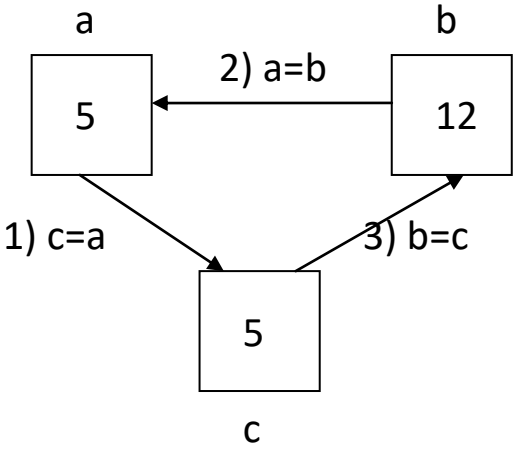
Типы вычислительных процессов

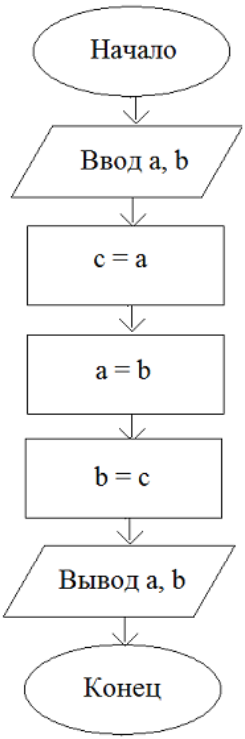
Вычислительные процессы могут быть: линейные, разветвляющиеся и циклические.

Линейные алгоритмы

Линейный алгоритм – алгоритм, в котором все команды выполняются последовательно друг за другом.

Пример 1: составить алгоритм обмена значений переменных a и b.

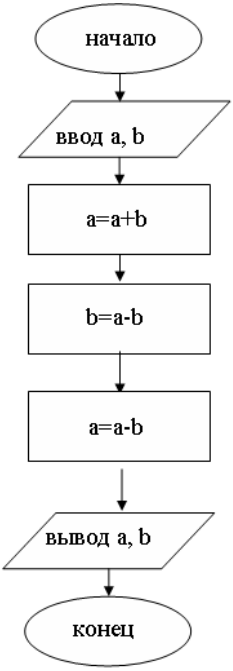




Команды	a	b	c
a=5, b=12	5	12	-
c=a	5	12	5
a=b	12	12	5
b=c	12	5	5

Пример 2: Составить алгоритм обмена значений переменных a и b без использования дополнительной переменной.

Команды	a	b
a=3, b=7	3	7
a=a+b	10	7
b=a-b	10	3
a=a-b	7	3



Пример 2: составить алгоритм вычисления a^8, используя не более 3х действий умножения (возведение в степень не использовать)

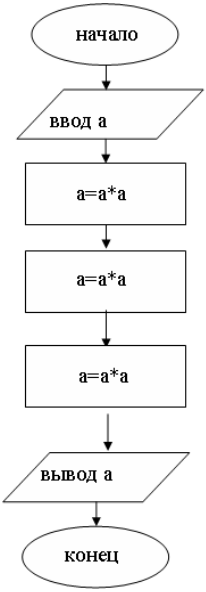
a: 2 ->4 ->16 -> 256

a=a*a | a^2

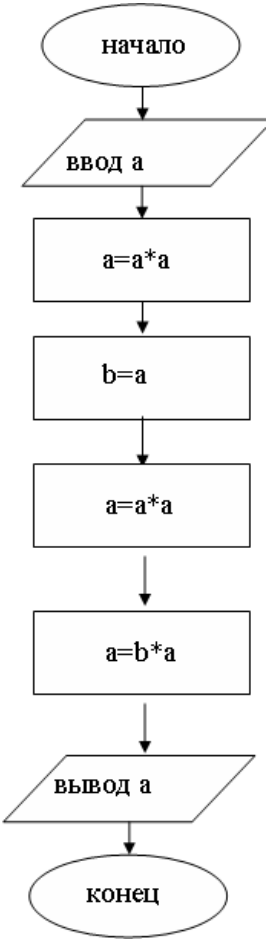
a=a*a | a^4

a=a*a | a^8

Команды	a
a=2	2
a=a*a	4
a=a*a	16
a=a*a	256



Пример 3: Составить алгоритм вычисления a^6, используя не более трех команд умножения.

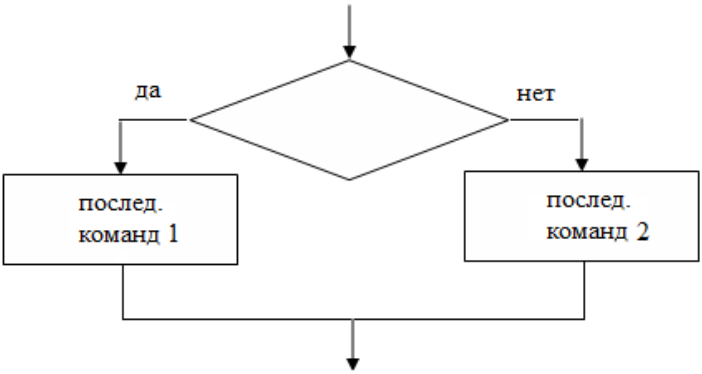


Команды	a	b
a=2	2	-
a=a*a	2^2	-
b=a	2^2	2^2
a=a*a	2^4	2^2
a=b*a	2^6	2^2

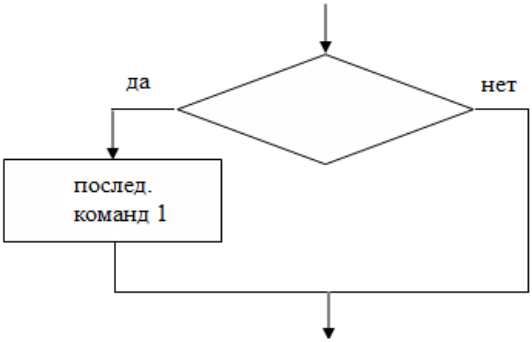
Алгоритмы с ветвлением

Часто при выполнении алгоритма должны предлагаться различные действия в зависимости от выполнения или невыполнения некоторого условия. Такие алгоритмические структуры называют ветвлением.

Полное ветвление



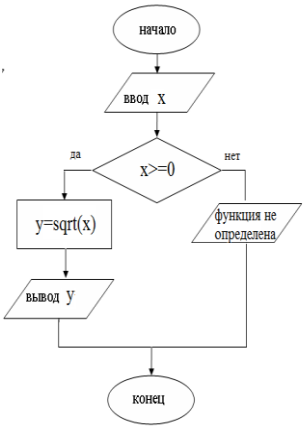
Неполное ветвление



Пример 4. Вычислить выражение $y = \sqrt{x}$ для введенного *x*.

Исходные данные: x.

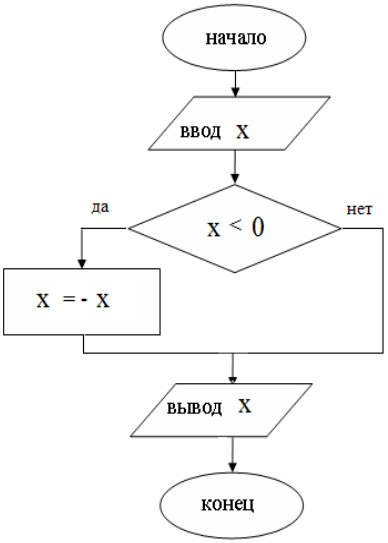
Результат: y, или ‘функция не определена’

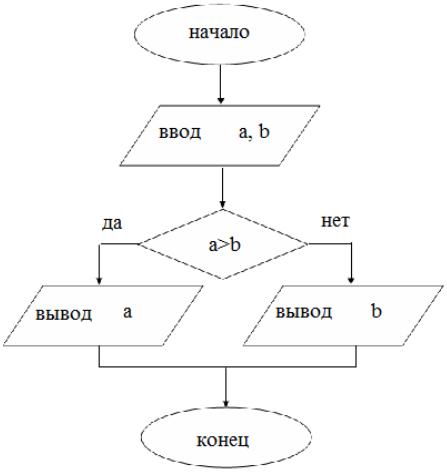


проверяемый случай	x>=0	y	результат
x=9	9>=0 да	y= $\sqrt{9}=\pm 3$	y= ± 3
x=-9	-9>=0 нет	-	функция не определена

Пример 5. Вычислить выражение $y=|x|$ для введенного *x*.

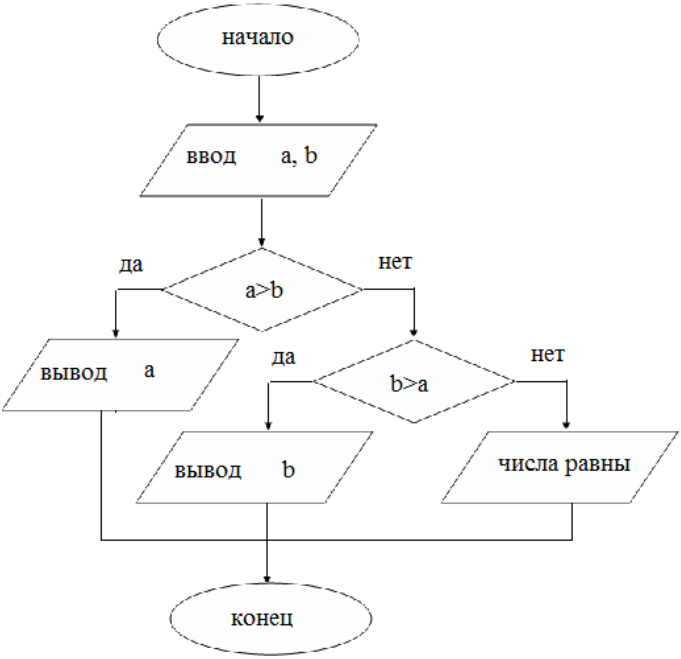
$$|x|=\begin{cases} x, x \geq 0 \\ -x, x < 0 \end{cases}$$



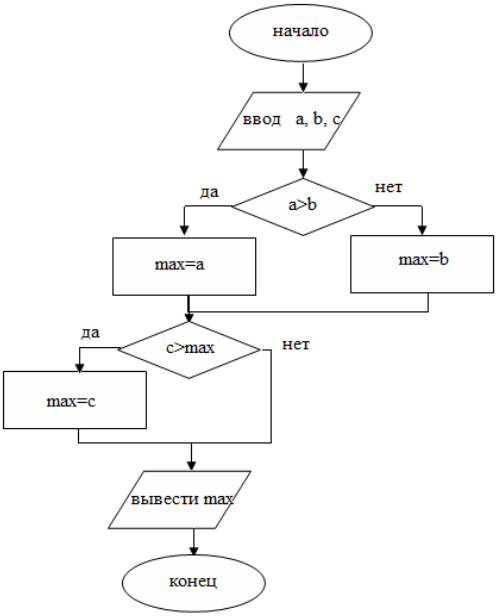


Пример 6. Выбрать максимальное из 2х чисел a и b.

1 вариант



2 вариант



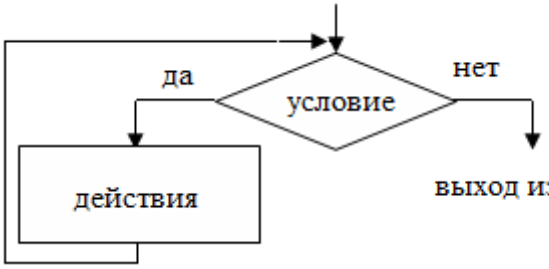
Пример 7. Выбрать максимальное из 3х чисел a, b, c.

проверяемый случай	x	max	y	результат
a=9 b=15 c=2	9>15 нет	15	2>15 нет	15
a=8 b=3 c=22	8>3 да	8	22>8 да	22
a=12 b=9 c=1	17>9 нет	17	1>17 нет	17

Алгоритмы с циклами

Цикл – многократное повторение одних и тех же действий.

1. Цикл с предусловием



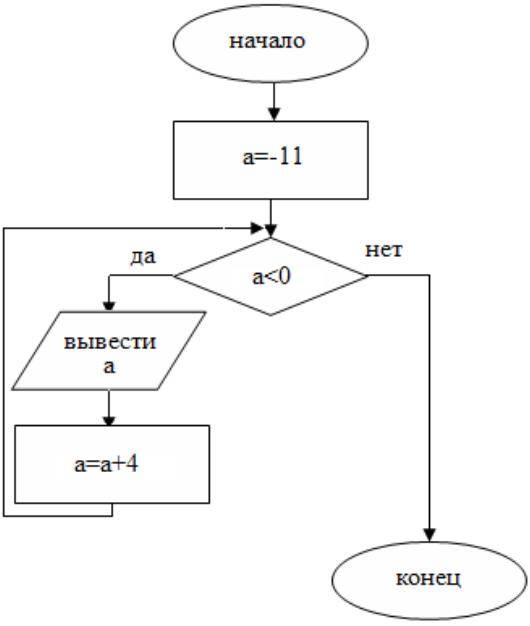
Такой цикл называют «пока». Механизм его работы: пока условие истинно, повторять…

Пример 8. *Вывести все «-» члены арифметической прогрессии -11; -7…*

Пусть a – очередной член прогрессии.

$a=a_1+4$ – следующий член прогрессии.

Пока a<0, повторять $a=a_1+4$.



Цикл с предусловием может не выполниться ни разу, если условие сразу оказалось ложным.

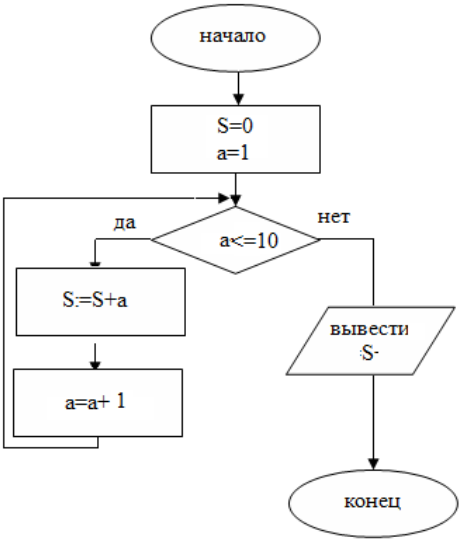
Пример 9. *Найти сумму первых десяти натуральных чисел.*

S=1+2+…+10

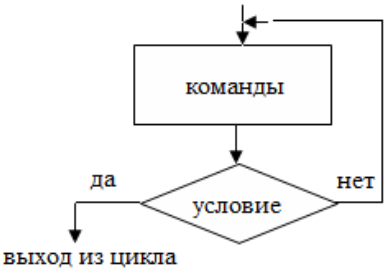
S – сумма

a – очередное слагаемое.

S:=S+a a:=a+1	Выполнять, пока a<=10



2. Цикл с постусловием.

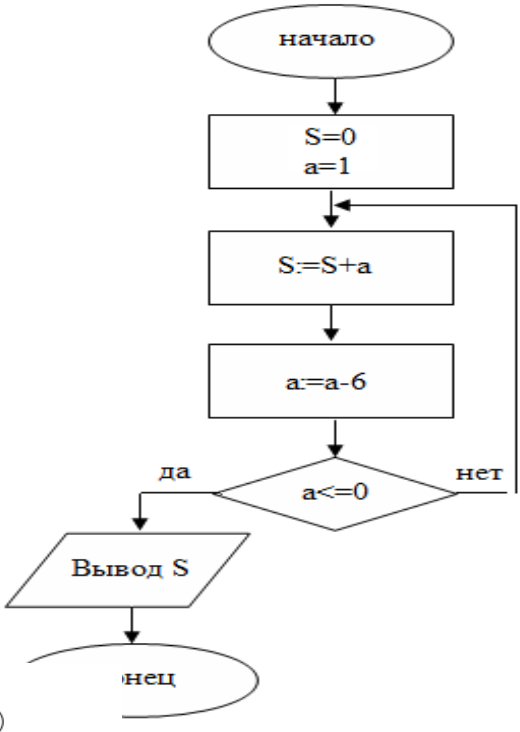


Механизм работы: повторять, пока условие не станет истинным.

Этот цикл всегда выполняется хотя бы 1 раз.

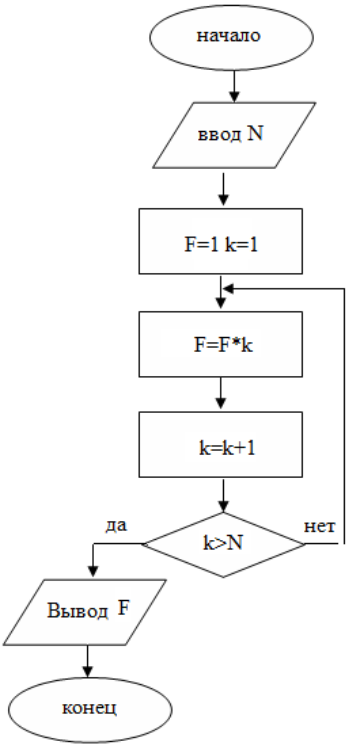
Пример 10. *Найти сумму положительных членов арифметической прогрессии: 17; 11 ...*

S =a+S a: =a-6	тело цикла повторять до тех пор, пока не выполнится условие a<=0



Пример 11. *Вычислить n!*

F=F *k k=k +1	тело цикла повторять до тех пор, пока не выполнится условие k>N



Задания для самостоятельной работы

1. Составить блок-схему алгоритма решения задачи:

а). По длине ребра куба найти площадь грани, площадь полной поверхности и объем куба.

б). Найти площадь кольца с радиусами r1 и r2.

в). Вычислить площадь треугольника по трем сторонам(по формуле Герона).

г). Вычислить площадь параллелограмма по двум сторонам и углу между ними, заданному в градусах.

д). Вычисление суммы цифр введенного натурального двухзначного числа.

е). По координатам трёх вершин некоторого треугольника найти его площадь и периметр.

ж). Вводятся X и Y. Если X больше Y, то произвести их обмен.

з). Из чисел A, B, C, D выбрать максимальное.

и). Введено четырехзначное число. Найти количество четных цифр.

к). Введено трехзначное число. Найти сумму четных цифр.

л). Введено четырехзначное число. Найти среднее арифметическое нечетных цифр.

м). Определить, существует ли треугольник с заданными сторонами a, b, c.

н). Найти сумму делителей числа n.

о). Найти сумму 1 + 1/3 + 1/5 +...(N слагаемых).

п). Подсчитать сумму двухзначных чисел, сумма цифр которых не превышает 7.

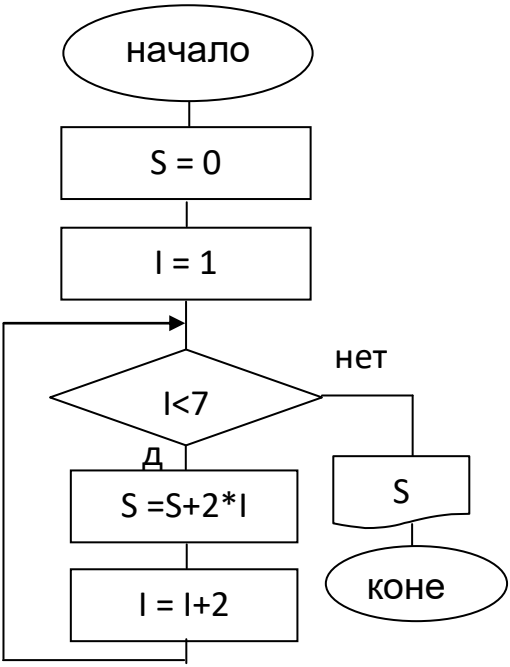
р). Задана арифметическая прогрессия 2; 5; … . Определите наименьшее количество членов прогрессии, начиная с первого, сумма которых превышает 50.

с). Вывести таблицу значений функции $y= \sin^2 x - \cos x$ на интервале $[-\pi,\pi]$ с шагом $\pi/10$.

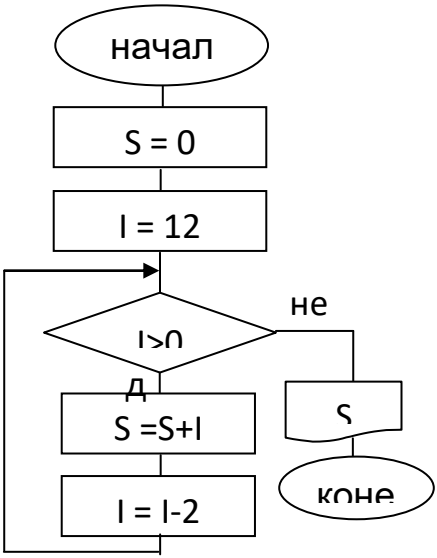
т). Найти сумму: S = x+2x+3x... (n слагаемых)

2. Выполнить ручную трассировку и определить результат выполнения алгоритма

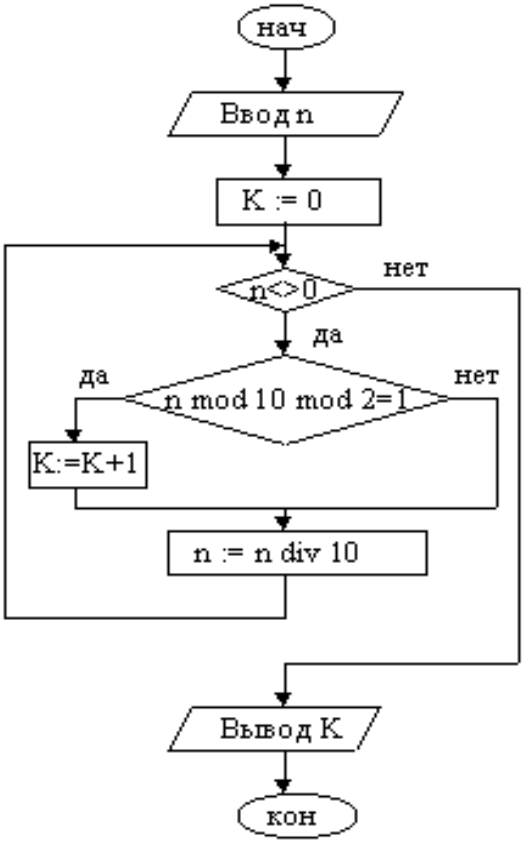
а)



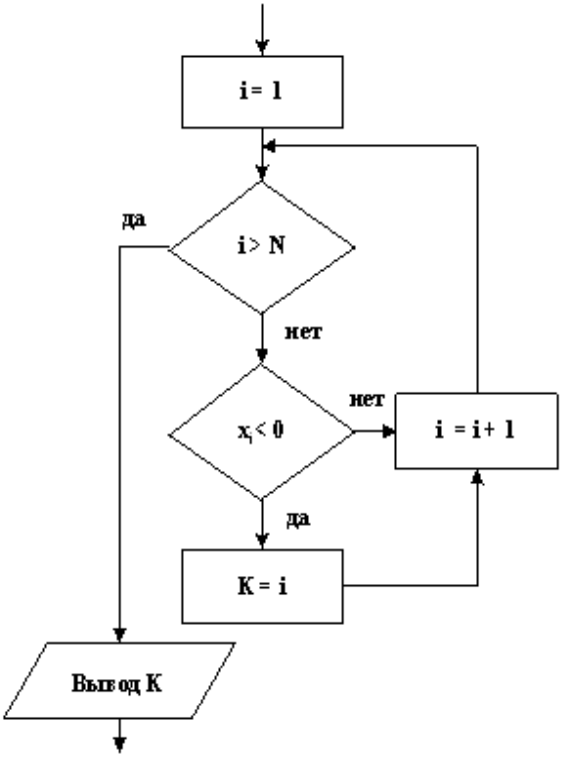
б)



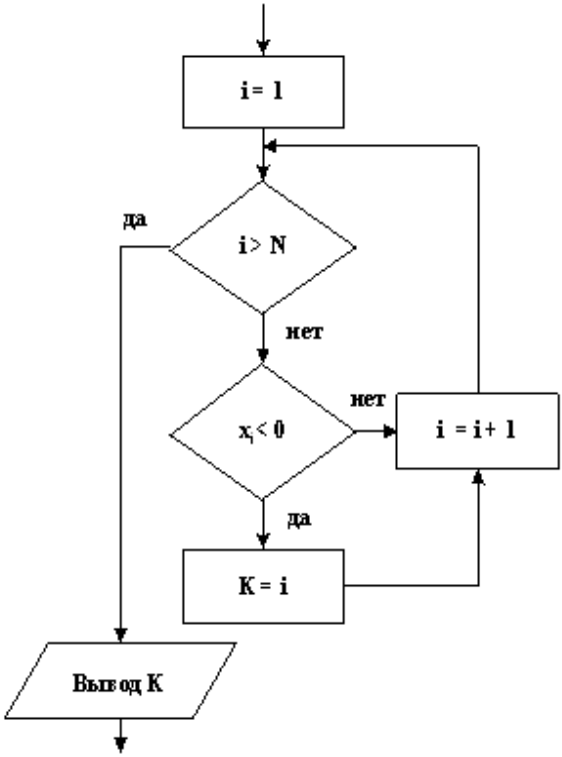
д) определить значение К при n=3275



Е) Для массива X = (-8, 9, 10, -2, 4, -5, 3, 2) найти значение переменной, которая является результатом работы алгоритма



Ж) Для массива X = (8, 0, -10, -8, 4, -9, -3, 7) найти значение переменной, которая является результатом работы алгоритма



Модульное проектирование программные средств.

Задание:

1. Разработать ПС.
2. Построить иерархическую схему ПС.
3. Оценить связность и сцепление модулей входящих в вашу программную систему.
4. Построить схему информационных связей.
5. Построить схему Варнье-Орра или схему HIPO.

Сделать возможным корректировку, добавление и удаление записей. Файл должен содержать не менее 5 записей.

В отчетных формах возможен поиск, отчет за период и т.д.

Разработанное ПС должно быть качественным, написанным в хорошем стиле. Сделайте вывод о проделанной работе.

Вариант №1

Создать БД «Библиотека», включающую следующую информацию: код книги, автор, название, год издания, Ф.И.О. читателя, год рождения читателя, образование и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №2

Создать БД «Студент», включающую следующую информацию: Ф.И.О. студента, год рождения, домашний адрес, факультет, специальность, курс, какое среднее учебное заведение закончил и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №3

Создать БД «Наряд», включающую следующую информацию: шифр наряда, дата (год, месяц, число), номер цеха, табельный номер, код операции, норма времени, количество изготовленных деталей и др.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №4

Создать БД «Продуктовый магазин», включающую следующую информацию: код продукта, название, вид упаковки, дата поступления, срок хранения, объем закупки, объем продажи и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №5

Создать БД «Бухгалтерия», включающую следующую информацию: Ф.И.О. работника, образование, должность, дата поступления на работу, оклад и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №6

Создать БД «Сотрудники НИИ», включающую следующую информацию: № отдела, табельный номер, Ф.И.О., № темы над которой работает сотрудник, продолжительность работы в месяцах, код должности, размер заработной платы и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №7

Создать БД «Реализованный товар», включающую следующую информацию: номер магазина, номер секции, номер чека, наименование товара, артикул товара, цена товара, дата продажи и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №8

Создать БД «Сессия», включающую следующую информацию: Ф.И.О. студента, факультет, специальность, курс, оценки по 4 предметам и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №9

Создать БД «Аэрофлот», включающую следующую информацию: номер рейса, пункт назначения, время вылета, время прибытия, количество свободных мест в салоне и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №10

Создать БД «Аптека», включающую следующую информацию: код лекарства, название, дата выпуска, срок хранения, форма изготовления, объем партии и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №11

Создать БД «Абонент телефона», включающую следующую информацию: Ф.И.О. абонента, адрес, номер телефона, год установки телефона, количество ремонтных работ, вид ремонта, ФИО мастера и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №12

Создать БД «Железнодорожный вокзал», включающую следующую информацию: номер поезда, станция назначения, время отправления, время в пути, время прибытия, наличие билетов, количество вагонов и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Вариант №13

Создать БД «Подписка», включающую следующую информацию: индекс издания, наименование, цена подписки с доставкой, цена подписки без доставки, количество подписчиков, на какой срок подписался (1 месяц, …1 год), наличие льгот на подписку и т.д.

Разработать 2 формы отчетности с возможностью подключения 3-й формы.

Тема 3. Разработка ПО на основе объектно-ориентированного программирования

Средства объектного программирования языка С++.

Учебная цель:

Цель работы состоит в приобретении навыков

Краткие теоретические и учебно-методические материалы по теме практической работы

Сущность объектно-ориентированного подхода к программированию заключается в том, что основные идеи объектно-ориентированного подхода опираются на следующие положения:

- программа представляет собой модель некоторого реального процесса, части реального мира;
- модель реального мира или его части может быть описана как совокупность взаимодействующих между собой объектов;
- объект описывается набором параметров, значения которых определяют состояние объекта, и набором операций (действий), которые может выполнять объект;
- взаимодействие между объектами осуществляется посылкой специальных сообщений от одного объекта к другому. Сообщение, полученное объектом, может потребовать выполнения определенных действий, например, изменения состояния объекта;
- объекты, описанные одним и тем же набором параметров и способные выполнять один и тот же набор действий представляют собой класс однотипных объектов.

С точки зрения языка программирования класс объектов можно рассматривать как тип данного, а отдельный объект - как данное этого типа. Определение программистом собственных классов объектов для конкретного набора задач должно позволить описывать отдельные задачи в терминах самого класса задач (при соответствующем выборе имен типов и имен объектов, их параметров и выполняемых действий).

Таким образом, объектно-ориентированный подход предполагает, что при разработке программы должны быть определены классы используемых в программе объектов и построены их описания, затем созданы экземпляры необходимых объектов и определено взаимодействие между ними.

Классы объектов часто удобно строить так, чтобы они образовывали иерархическую структуру. Например, класс «Студент», описывающий абстрактного студента, может служить основой для построения классов «Студент 1 курса», «Студент 2 курса» и т.д., которые обладают всеми свойствами студента вообще и некоторыми дополнительными свойствами, характеризующими студента конкретного курса. При разработке интерфейса с пользователем программы могут использовать объекты общего класса «Окно» и объекты классов специальных окон, например, окон информационных сообщений, окон ввода данных и т.п. В таких иерархических структурах один класс может рассматриваться как базовый для других, производных от него классов. Объект производного класса обладает всеми свойствами базового класса и некоторыми собственными свойствами, он может реагировать на те же типы сообщений от других объектов, что и объект базового класса и на сообщения, имеющие смысл только для производного класса. Обычно говорят, что объект производного класса наследует все свойства своего базового класса.

Некоторые параметры объекта могут быть локализованы внутри объекта и недоступны для прямого воздействия извне объекта. Например, во время движения объекта-автомобиля объект-водитель может воздействовать только на ограниченный набор органов управления (рулевое колесо, педали газа, сцепления и тормоза, рычаг переключения передач) и ему недоступен целый ряд параметров, характеризующих состояние двигателя и автомобиля в целом.

Очевидно, для того, чтобы продуктивно применять объектный подход для разработки программ, необходимы языки

программирования, поддерживающие этот подход, т.е. позволяющие строить описание классов объектов, образовывать данные объектных типов, выполнять операции над объектами. Одним из первых таких языков стал язык SmallTalk в котором все данные являются объектами некоторых классов, а общая система классов строится как иерархическая структура на основе предопределенных базовых классов.

Опыт программирования показывает, что любой методический подход в технологии программирования не должен применяться слепо с игнорированием других подходов. Это относится и к объектно-ориентированному подходу. Существует ряд типовых проблем, для которых его полезность наиболее очевидна, к таким проблемам относятся, в частности, задачи имитационного моделирования, программирование диалогов с пользователем. Существуют и задачи, в которых применение объектного подхода ни к чему, кроме излишних затрат труда, не приведет. В связи с этим наибольшее распространение получили объектно-ориентированные языки программирования, позволяющие сочетать объектный подход с другими методологиями. В некоторых языках и системах программирования применение объектного подхода ограничивается средствами интерфейса с пользователем (например, Visual FoxPro ранних версий).

Наиболее используемыми в настоящее время объектно-ориентированными языками являются Паскаль с объектами и Си++, причем наиболее развитые средства для работы с объектами содержатся в Си++.

Объектно-базирующееся программирование - это методология разработки программ, основанная на использовании совокупности объектов, каждый из которых является реализацией определенного класса. Программный код и данные структурируются так, чтобы имитировалось поведение фактически существующих объектов. Содержимое объекта защищено от внешнего мира посредством инкапсуляции. Благодаря наследованию уже запрограммированные функциональные возможности можно использовать и для других объектов. Объекты являются программным представлением физических и/или логических сущностей реального мира. Они необходимы для моделирования поведения физических или логических объектов, которые они представляют. Для изменения поведения и состояния элементов управления используются их свойства, методы, поля и события. Классы задают структуру

объектов. При программировании создаются объекты - представители классов. С другой стороны, классы составляют группы одноименных объектов. Внутренняя структура класса в Visual Basic передается объекту с использованием модуля класса. С использованием команды Project →Add Class Module модуль класса можно добавить в проект. После добавления модуля класса выводится окно кода, в котором можно реализовать компоненты (свойства, поля, методы, события) класса.

Пример использования методики объектно–ориентированного программирования

Задание. Создать в предметной области «Автомобили» класс с требуемой функциональностью (использовать компоненты класса: методы, поля и т.д.). Создать объект - экземпляр класса. Создать пример использования объектом компонентов класса.

Реализация задания

Приводится проект, дающий справку желающим приобрести автомобиль. Создан класс Class1, содержащий компоненты, определяющие название фирмы-изготовителя, модель автомобиля, его стоимость, изображение автомобиля и следующие технические характеристики:

- тип двигателя (бензин/дизель),
- число цилиндров/рабочий объём,
- система питания (карбюратор/впрыскивание),
- мощность (л.с),
- максимальная скорость (км/час),
- разгон 0 - 100 (км/час)/сек,
- привод (передний/задний/4х4).

Далее создаётся экземпляр класса: Dim av As New Class1, использующий компоненты класса.

Пользователю предлагается решить вопрос о необходимости покупки, выбрать фирму-изготовителя, ответить на вопрос о выводе изображения покупаемого автомобиля, либо его технических характеристик, либо обеих категорий одновременно (используются процедуры Property Get и Property Let, созданные в классе Class1), после чего программа адекватно реагирует: либо выводятся вышеперечисленные данные, либо выводится некоторое сообщение.

Для реализации проекта нужно выполнить следующую последовательность действий:

1. добавить в стандартный проект модуль класса (Project ⟶ Add Class Module ⟶Class Module ⟶Открыть),
2. создать:
   - методы класса. Четыре метода создаются в процедурах: Public Function Met1(), Public Function Met2(), Public Function Met3(), Public Function Met4() (Tools ⟶ Add Procedure ⟶ввести имя { Met1, Met2, Met3, Met4} ⟶ выбрать Function ⟶выбрать Public ⟶OK),
   - свойства класса. Свойства задаются с использованием процедур Property Get и Property Let (Tools ⟶ Add Procedure ?ввести имя (здесь - varian) ⟶ выбрать Property ⟶выбрать Public ⟶OK),
   - поля класса - avto, firma, model, stoim, pict, var, см. ниже.
3. создать на форме:
   - два элемента управления ComboBox с именами Combo1 и Combo2,
   - два элемента управления CommandButton с именами Command1 и Command2; значению свойства Caption объекта Command1 присвоить значение "OK", Command2 - "Exit",
   - элементы управления Label1 - Label4, значениям свойств Caption присвоить: Label1 - "Хотите ли Вы купить машину?", Label2 - "Выберите фирму-изготовитель", Label3 - "Хотите ли Вы увидеть изображение выбранного автомобиля или его технические характеристики ?", Label4- "", свойству Visible объекта Label4 присвоить False,
   - массив элементов управления OptionButton (присвоить значения свойствам - Option1(0).Caption= "да", Option1(1).Caption= "нет"),
   - массивы элементов управления PictureBox: Picture1(0) - Picture1(12) и Picture2(0) - Picture(12). Свойству Visible всех элементов управления присвоить значение False. Свойству Picture каждого элемента управления присвоить значение изображения соответствующего автомобиля и списка технических характеристик (эти списки создаются в приложении Excel, далее таблицы

передаются в приложение Paint и сохраняются как рисунки).

4. ввести код в область класса (см. ниже "область проекта Class1"),
5. ввести код, данный ниже, в области:
   - General Declarations формы,
   - Combo1, событие Click,
   - Combo2, событие Click,
   - Command1, событие Click,
   - Command, событие Click,
   - Form, событие Load,
   - Form, событие Unload,
6. стартовать проект, получить справку о предполагаемой покупке.

```
//////////////////////////область проекта Class1////////////////////////////
Public avto As Boolean
Public firma As String
Public model As String
Public stoim As String
Public pict As String
Dim var As String
Private Sub Class_Initialize() ' инициализация полей класса
avto = False: firma = "": model = "": stoim = "": var = ""
End Sub
Public Function Met1()
' Если пользователь нажал кнопку (OptionButton) - Да, то выполнить процедуры
' Met2, Met3, Met4, результатом выполнения которых является вывод данных:
' марка, стоимость, изображение и технические характеристики, иначе
' Met1 = False и выводится сообщение "Приносим свои извинения, мы даем
' информацию для желающих купить автомобиль"
If avto = True Then
model = Met2()
stoim = Met3()
pict = Met4() ' поле pict определяет номера элементов массивов
' PictureBox, см. Met4
Met1 = True
```

```
Else
Met1 = False
End If
End Function
' после щелчка на кнопках Да/Нет (два переключателя OptionButton) и выбора
' фирмы из списка ComboBox с именем Combo1 определить марку автомобиля
Public Function Met2()
Select Case firma
Case "AUDI": Met2 = "A6"
Case "CITROEN": Met2 = "C5"
Case "FORD": Met2 = "Focus"
Case "HONDA": Met2 = "Accord"
Case "HYUNDAI": Met2 = "Elanta"
Case "JEEP": Met2 = "Grand Cherokee LTD"
Case "LAND ROVER": Met2 = "Land Rover Discovery"
Case "LEXSUS": Met2 = "RX330"
Case "MITSUBISHI": Met2 = "Pajero III"
Case "NISSAN": Met2 = "Primera(1.8)"
Case "PEUGEOT": Met2 = "307 XR"
Case "PORSCHE": Met2 = "Cayenne Turbo"
Case "RENAULT": Met2 = "Laguna II"
End Select
End Function
' определить стоимость автомобиля в долларах США
Public Function Met3()
Select Case firma
Case "AUDI": Met3 = "41500"
Case "CITROEN": Met3 = "20100"
Case "FORD": Met3 = "12430"
Case "HONDA": Met3 = "33900"
Case "HYUNDAI": Met3 = "13790"
Case "JEEP": Met3 = "41690"
Case "LAND ROVER": Met3 = "40850"
Case "LEXSUS": Met3 = "65500"
Case "MITSUBISHI": Met3 = "56640"
Case "NISSAN": Met3 = "25100"
Case "PEUGEOT": Met3 = "13808"
Case "PORSCHE": Met3 = "140500"
```

```
Case "RENAULT": Met3 = "22900"
End Select
End Function
Public Function Met4()
' при выборе данных из списка ComboBox с именем Combo2
' (после щелчка на кнопке "ОК" ) определяется номер элемента массива
' PictureBox, соответствующий выбранной фирме-изготовителю и
' на экран позднее выводится соответствующая фотография
' и/или технические характеристики автомобиля
Select Case firma
Case "AUDI": Met4 = "0"
Case "CITROEN": Met4 = "1"
Case "FORD": Met4 = "2"
Case "HONDA": Met4 = "3"
Case "HYUNDAI": Met4 = "4"
Case "JEEP": Met4 = "5"
Case "LAND ROVER": Met4 = "6"
Case "LEXSUS": Met4 = "7"
Case "MITSUBISHI": Met4 = "8"
Case "NISSAN": Met4 = "9"
Case "PEUGEOT": Met4 = "10"
Case "PORSCHE": Met4 = "11"
Case "RENAULT": Met4 = "12"
End Select
End Function
' процедура Property Let используется для задания значения свойства,
' Property Get - для считывания значения свойства
Public Property Get varian() As String
Select Case var
Case Is = 0: varian = "pict"
Case Is = 1: varian = "texn"
Case Is = 2: varian = "all"
End Select
End Property
Public Property Let varian(ByVal vNewValue As String)
Select Case vNewValue
Case "изображение": var = 0
```

```
Case "технические параметры": var = 1
Case Else: var = 2
End Select
End Property
////////////////////////////////область проекта Form1/////////////////////////////
Dim av As Class1 ' av - экземпляр класса
Dim v As String
Dim i As Integer, j As Integer
Private Sub Combo1_Click()
' сделать невидимыми элементы управления Label и Picture
' (формирующие фотографии, технические характеристики, фирму,
' марку и стоимость), для того, чтобы впоследствии на форму
' выводились только те из них, которые определяет своими
' действиями покупатель
Label5.Visible = False
For i = 0 To 12
Picture1(i).Visible = False
Picture2(i).Visible = False
Next
Dim ot As String ' переменная для хранения сообщения программы
av.firma = Combo1.Text ' значение поля firma объекта av взять из
' списка ComboBox с именем Combo1
av.avto = Option1(0).Value ' значение поля avto объекта av взять
' из поля массива OptionButton
If av.Met1 = True Then
ot = " " & CStr(av.firma) & vbCrLf: ot = ot & " " & vbCrLf
ot = ot & " модель " & CStr(av.model) & vbCrLf: ot = ot & " " & vbCrLf
ot = ot & " цена в $ " & CStr(av.stoim) & vbCrLf: ot = ot & " " & vbCrLf
ot = ot & "Для получения более полной информации обращайтесь_
по телефону 7077888"
MsgBox Title:="Мы можем предложить", Prompt:=ot
Else
Label5.Visible = False
Picture1(Val(av.pict)).Visible = False ' аргумент Picture1: (av.pict)
' определяет индекс элемента массива PictureBox
```

```
ot = "Приносим свои извинения, мы даем информацию для желающих_
купить автомобиль"
MsgBox Title:="Автосалон START", Prompt:=ot
End If
End Sub
Private Sub Combo2_Click()
av.varian = Combo2.Text ' см. процедуру Property Let. Присваиваем
' свойству varian значение выбранные из списка ComboBox с именем Combo2
End Sub
Private Sub Command1_Click()
Label5.Visible = False
Label5.Caption = ""
For i = 0 To 12
Picture1(i).Visible = False
Picture2(i).Visible = False
Next
v = av.varian ' см. процедуру Property Get. Переменной v присваиваем
' значение свойства varian объекта av
av.avto = Option1(0).Value
If av.Met1 = True Then
Select Case v
Case "pict"
Picture1(Val(av.pict)).Visible = True
Case "texn"
Picture2(Val(av.pict)).Visible = True ' технические характеристики
' хранятся как изображения в соответствующих элементах
' массива PictureBox2
Case "all"
Picture1(Val(av.pict)).Visible = True
Picture2(Val(av.pict)).Visible = True
Label5.Visible = True
Label5.Caption = CStr(av.firma) & " " & CStr(av.model) & _
vbCrLf & "цена в $ " & CStr(av.stoim)
End Select
Else
```

```
Picture1(Val(av.pict)).Visible = False
Picture2(Val(av.pict)).Visible = False
MsgBox Title:="Автосалон START", Prompt:="Приносим свои_
извинения, мы даем информацию для желающих купить автомобиль"
End If
End Sub
Private Sub Command2_Click()
MsgBox Title:="Автосалон START", Prompt: = "Мы всегда рады помочь!_
Будем рады новой встрече!"
End ' выход из программы после сообщения MsgBox.
End Sub
Private Sub Form_Load()
Set av = New Class1 ' описание объектной переменной дано выше
' заполнение списка ComboBox с именем Combo1 названиями фирм
Combo1.AddItem "AUDI"
Combo1.AddItem "CITROEN"
Combo1.AddItem "FORD"
Combo1.AddItem "HONDA"
Combo1.AddItem "HYUNDAI"
Combo1.AddItem "JEEP"
Combo1.AddItem "LAND ROVER"
Combo1.AddItem "LEXSUS"
Combo1.AddItem "MITSUBISHI"
Combo1.AddItem "NISSAN"
Combo1.AddItem "PEUGEOT"
Combo1.AddItem "PORSCHE"
' заполнение списка ComboBox с именем Combo2 предложениями для
' выбора данных в процедурах Property Get и Property Let
Combo2.AddItem "изображение"
Combo2.AddItem "технические параметры"
Combo2.AddItem "все данные"
End Sub
Private Sub Form_Unload(Cancel As Integer)
Set av = Nothing ' удалить объект из памяти
End Sub
```

Инструкция пользователя

После старта проекта при отрицательном ответе на вопрос «Хотите ли Вы купить машину?» выводится сообщение: «Приносим свои извинения, мы даем информацию для желающих купить автомобиль».

При положительном ответе на вопрос и выборе фирмы-изготовителя из списка на экран выводится сообщение о фирме, марке и стоимости автомобиля.

Далее покупатель может просмотреть или внешний вид, или технические характеристики, или одновременно обе категории, выбрав соответствующую строку во втором списке и сделав щелчок на кнопке ОК (используются процедуры Property Get и Property Let), где дан результат работы программы при выборе строки «все данные».

При щелчке на кнопке Exit выводится сообщение: «Мы всегда рады помочь! Будем рады новой встрече!» и проводится выход из программы.

Заключение (выводы)

Созданный программный продукт позволяет клиенту получить справочные данные при покупке автомобиля. Представленная программа является лишь небольшим примером использования классов, в реальности же сфера применения свойств объектно-базирующегося программирования гораздо шире.

Задания для выполнения:

1. Для предметной области (заданной преподавателем к практической работе №2) выполнить объектно-ориентированное проектирование программного продукта.

2. Оформить отчет.

Отчет по практической работе должен быть оформлен на основании требований и состоять из следующих структурных элементов:

- Анализа предметной области
- Определения функций предметной области
- Схемы документопотока
- Выделенных сущностей, атрибутов и установленных связей
- Концептуальной модели
- Описания выходных и входных данных

3. Ответить на контрольные вопрсы

Вопросы для закрепления теоретического материала к практическому занятию:

1. В чем заключается сущность объектно-ориентированного подхода при разработке программного продукта?
2. На что направлен объектно-ориентированный анализ?
3. Перечислите основные достоинства объектно-ориентированной методологии по сравнению со структурными методами.
4. Перечислите принципы объектного подхода. Дайте им краткие характеристики
5. Назовите основные методики объектно-ориентированного анализа.
6. Какие понятия, необходимые для проектирования системы включает концептуальная модель?

Порядок выполнения отчета по практической работе

Защита отчета по практической работе заключается в предъявлении преподавателю полученных результатов, демонстрации полученных навыков и ответах на вопросы преподавателя.

Тема 4. Качество программного обеспечения

Оценка качества процесса разработки.

Процесс оценки качества программного обеспечения осуществляется для каждой фазы его жизненного цикла и включает:

• выбор совокупности (номенклатуры) показателей качества оцениваемого программного средства;

• определение значений этих показателей;

• сравнение полученных значений с базовыми значениями показателей.

Под жизненным циклом программного обеспечения понимается период времени с момента начала предпроектного обследования до момента полного выхода программы из употребления пользователями.

Весь период жизненного цикла программного обеспечения делится на следующие временные промежутки или фазы.

1. Анализ - этап определения требований к программному обеспечению, спецификация требований и формирования технического задания на проектирование программы.

2. Проектирование - этап разработки технического проекта.

3. Реализация - этап разработки программного обеспечения, средств тестирования и документации.

4. Тестирование - этап испытания программного обеспечения и устранение недостатков.

5. Изготовление - этап преобразования программного обеспечения в форму, готовую для поставки; завершение формирования документации.

6. Внедрение - этап подтверждения стабильной работы программного

обеспечения; ввод в стадию активного применения.

7. Эксплуатация - этап применения программного обеспечения по назначению.

8. Сопровождение - этап устранения дефектов в процессе эксплуатации; усовершенствование, оптимизация и модификация используемого программного обеспечения при условии сохранности целостности программного продукта.

Оценка качества программного обеспечения на всех фазах жизненного цикла осуществляется на основе чётырехуровневой системы показателей.

Показатели первого уровня (факторы качества) характеризуют потребительски-ориентированные свойства программных средств, которые соответствуют потребностям пользователей. Факторы качества, собственно, определяют наиболее значимые (с точки зрения использования) свойства программ. Для оценки качества программного обеспечения используют следующие факторы:

- надежность;
- сопровождаемость;
- удобство применения;
- эффективность;
- универсальность;
- корректность.

Каждый фактор представляет собой интегральную оценку, которой соответствует несколько критериев качества (комплексных показателей второго уровня).

В качестве примера в табл. 6.1 представлен состав фактора универсальность по критериям и метрикам для различных фаз жизненного цикла программных средств. Здесь символом «+» отмечены фазы жизненного цикла, на которых определяются значения указанных метрик для расчета соответствующих критериев и факторов. Пустые ячейки таблицы, соответствующие метрикам и

фазам жизненного цикла, означают, что указанные метрики на указанных в таблице фазах не определяются.

Таблица 1. Состав и соответствие показателей качества программных систем на различных фазах их жизненного цикла

Факторы	Критерии	Метрики	Фаза жизненного цикла					
			Анализ	Проектирование	Реализация	Тестирование	Изготовление	Сопровождение
Универсальность	Гибкость	Широта охвата функций	+	+	+	+	+	+
		Простота архитектуры проекта		+	+	+	+	+
		Сложность архитектуры проекта		+	+	+	+	+
		Сложность структуры кода программы			+	+	+	+
		Применение стандартных протоколов связи			+	+	+	+
		Применение стандартных интерфейсных программ			+	+	+	+

Окончание табл. 1

Факторы	Критерии	Метрики	Фаза жизненного цикла					
			Анализ	Проектирование	Реализация	Тестирование	Изготовление	Сопровождение
		Сложность архитектуры проекта		+	+	+	+	+
		Сложность структуры кода программы			+	+	+	+
		Применение стандартных протоколов связи			+	+	+	+
		Применение стандартных интерфейсных программ			+	+	+	+
Универсальность	Мобильность	Зависимость от используемого комплекса технических средств	+		+	+	+	+
		Зависимость от базового программного обеспечения	+		+	+	+	+
		Изоляция немобильности	+		+	+	+	+
	Модифицируемость	Простота кодирования			+	+	+	+
		Число комментариев			+	+	+	+
		Качество комментариев			+	+	+	+
		Использование описательных средств языка			+	+	+	+
		Независимость модулей			+	+	+	+

Оценить качество программного обеспечения по фактору универсальность для фазы жизненного цикла сопровождение по результатам оценки программы экспертами. Результаты проведенной экспертизы представлены в табл. 2. При проведении расчетов число экспертов окончательно определено в индивидуальном задании (см. табл. 3). Базовые значения критериев универсальности - 0,5.

Таблица 2. Значения оценочных элементов универсальности

Критерии	Метрики	Оценочные элементы	Оценки				
			1	2	3	4	5
1. Гибкость	1. Широта охвата функций	1. Оценка числа потенциальных пользователей	0,32	0,54	0,89	0,88	0,5
		2. Оценка числа функций программы	0,78	0,14	0,43	0,92	0,87
1. Гибкость	1. Широта охвата функций	3. Насколько набор функций удовлетворяет требованиям пользователя	0,2	0,91	0,14	0,98	0,6
		4. Насколько возможности программ охватывают область решаемых пользователем задач	0,71	0,64	0,31	0,94	0,05
		5. Возможность настройки формата входных данных для конкретных пользователей	0,99	0,04	0,57	0,94	0,31
	2. Простота архитектуры проекта	1. Наличие схемы иерархии модулей программ	0,3	0,4	0,1	0,48	0,97
		2. Оценка независимости модулей	0,25	0,06	0,35	0,8	0,57
		3. Оценка числа уникальных элементов	0,41	0,28	0,77	0,74	0,27
		4. Используется ли в текущем вызове модуля информация, полученная в предыдущем вызове	0,85	0,55	0,91	0,25	0,99
		5. Оценка организации точек входа и выхода модуля	0	0,01	0,21	0,9	0,84
		6. Наличие описания атрибутов модуля	0,56	0,2	0,21	0,12	0,51
	3. Сложность архитектуры проекта	1. Оценка программ по числу переходов и точек ветвления	0,71	0,1	0,9	0,89	0,92
	4. Сложность структуры кода программы	1. Использование метода пошагового уточнения	0,36	0,35	0,69	0,91	0,65
		2. Наличие описания структуры программы	0,02	0,21	0,67	0,73	0,38
		3. Наличие описания связей между элементами структуры программы	0,23	0,4	0,17	0,09	0,47
		4. Наличие в программе повторного выполнения функций (подпрограмм)	0,36	0,04	0,4	0,8	0,6

Продолжение табл.2

Критерии	Метрики	Оценочные элементы	Оценки				
			1	2	3	4	5
1. Гибкость	5. Применение стандартных протоколов связи	1. Использование стандартных протоколов связи	0,36	0,19	0,52	0,06	0,64
	6. Применение стандартных интерфейсных программ	1. Использование стандартных интерфейсных подпрограмм	0,86	0,96	0,16	0,54	0,91
2. Мобильность	1. Зависимость от используемого комплекса технических средств	1. Оценка зависимости программ от емкости оперативной памяти	0,14	0,31	0,96	0,19	0,68
		2. Оценка зависимости временных характеристик программы от скорости вычислений ЭВМ	0,53	0,9	0,83	0,98	0,73
		3. Оценка зависимости функционирования программы от числа внешних запоминающих устройств и их общей емкости	0,36	0,54	0,96	0,53	0,85
		4. Оценка зависимости функционирования программы от специальных устройств ввода-вывода	0,75	0,97	0,24	0,33	0,71
		5. Оценка зависимости программ от емкости оперативной памяти	0,49	0,5	0,67	0,81	0,78
	2. Зависимость от базового программного обеспечения	1. Применение специальных языков программирования	0,72	0,07	0,24	0,02	0,79
		2. Оценка зависимости программы от программ операционной системы	0,36	0,41	0,49	0,88	0,91
		3. Зависимость от других программных средств	0,12	0,54	0,43	0,16	0,52
	3. Изоляция немобильности	1. Оценка локализации переносимой части программы	0,63	0,9	0,83	0,41	0,58

Продолжение табл.2

Критерии	Метрики	Оценочные элементы	Оценки				
			1	2	3	4	5
3. Модифицируемость	1. Простота кодирования	1. Оценка использования отрицательных или булевых выражений	0,36	0,91	0,86	0,14	0,98
		2. Оценка программы по использованию условных переходов	0,52	0,86	0,26	0,31	0,94
		3. Оценка программы по использованию безусловных переходов	0,86	0,12	0,94	0,57	0,94
		4. Оформление процедур входа и выхода из циклов	0,99	0,25	0,15	0,1	0,48
		5. Ограничения на модификацию переменной индексации в цикле	0,5	0,05	0,23	0,35	0,8
		6. Оценка модулей по направлению потока управления	0,35	0,89	0,03	0,77	0,74
		7. Оценка программы по использованию локальных переменных	0,8	0,96	0,07	0,91	0,25
	2. Число комментариев	1. Оценка программы по числу комментариев	0,83	0,42	0,6	0,21	0,9
	3. Качество комментариев	1. Наличие заголовка в программе	0,01	0,41	0,21	0,21	0,12
		2. Комментарии к точкам ветвления	0,49	0,72	0,42	0,9	0,89
		3. Комментарии к машинозависимым частям программы	0,35	0,89	0,84	0,69	0,91
		4. Комментарии к машинозависимым операторам программы	0,24	0,57	0,3	0,67	0,73
		5. Комментарии к операторам объявления переменных	0,17	0,82	0,14	0,17	0,09
		6. Оценка семантики операторов	0,37	0,19	0,17	0,4	0,8
		7. Наличие соглашений по форме представления комментариев	0,61	0,04	0,16	0,52	0,06
		8. Наличие общих комментариев к программам	0,33	0,61	0,67	0,16	0,54

Окончание табл. 2

Критерии	Метрики	Оценочные элементы	Оценки				
			1	2	3	4	5
	4. Использование описательных средств языка	1. Использование языков высокого уровня	0,2	0,66	0,11	0,96	0,19
		2. Семантика имен используемых переменных	0,81	0,63	0,48	0,83	0,98
		3. Использование отступов, сдвигов и пропусков при формировании текста	0,9	0,42	0	0,96	0,53
		4. Размещение операторов по строкам	0,94	0,04	0,55	0,24	0,33
	5. Независимость модулей	1. Передача информации для управления по параметрам	0,16	0,75	0,15	0,67	0,81
		2. Параметрическая передача входных данных	0,59	0,63	0,87	0,24	0,02
		3. Наличие передачи результатов работы между модулями	0,64	0,17	0,86	0,49	0,88
		4. Наличие проверки правильности данных, получаемых модулями от вызываемого модуля	0,64	0,95	0,21	0,43	0,16
		5. Использование общих областей памяти	0,32	0,54	0,42	0,83	0,41

Тема 5. Технологии коллективной разработки программного обеспечения

Программные средства планирования и управления процессом разработки

Цель работы состоит в приобретении навыков визуализации информации с помощью диаграмм; расширения навыков построения нестандартных диаграмм в среде электронных таблиц применении средств деловой графики.

Краткие теоретические и учебно-методические материалы по теме практической работы

Диаграммы Ганта (англ.Gantt chart, также ленточная диаграмма) - это популярный тип столбчатых диаграмм, который используется для иллюстрации плана, графика работ по какому-либо проекту. Является одним из методов планирования проектов, представляет собой изображение календарного графика задач в проекте. График Ганта является своеобразным стандартом в области

управления проектами, ведь именно с его помощью появляется возможность показать структуру выполнения всех этапов проекта наглядно.

Диаграммы Ганта позволяет:

- визуально оценить последовательность задач, их относительную длительность и протяженность проекта в целом;
- сравнить планируемый и реальный ход выполнения задач;
- детально проанализировать реальный ход выполнения задач.

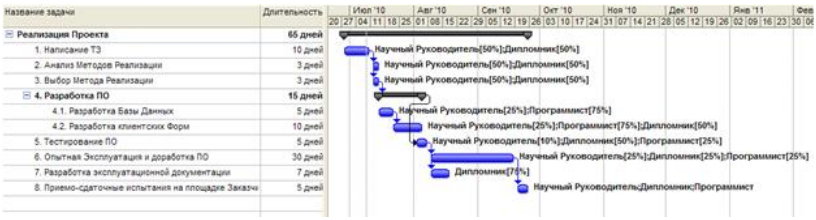
Диаграмма даёт возможность решить одну из основных задач и показать персоналу, над чем следует работать, какие ресурсы применять в процессе и с какой скоростью выполнять те или иные задачи. Вся информация подаётся в сжатом виде, без использования запутанных таблиц и огромного количества текста. При этом суть ясна и понятна всем, без исключения, участникам проекта.

Использование диаграммы значительно упрощает управление проектами небольших масштабов и даёт возможность всегда держать деятельность сотрудников под контролем.

Диаграмма Ганта состоит из полос, ориентированных вдоль оси времени. Каждая полоса на диаграмме представляет отдельную задачу в составе проекта (вид работы), её концы - моменты начала и завершения работы, её протяженность - длительность работы. Вертикальной осью диаграммы служит перечень задач. Кроме того, на диаграмме могут быть отмечены совокупные задачи, проценты завершения, указатели последовательности и зависимости работ, метки ключевых моментов (вехи), метка текущего момента времени «Сегодня» и др.

Диаграмма может использоваться для представления текущего состояния выполнения работ: часть прямоугольника, отвечающего задаче, заштриховывается, отмечая процент выполнения задачи; показывается вертикальная линия, отвечающая моменту «сегодня».

Часто диаграмма Ганта соседствует с таблицей со списком работ, строки которой соответствуют отдельно взятой задаче, отображенной на диаграмме, а столбцы содержат дополнительную информацию о задаче. Пример такой таблицы представлен ниже.



В электронном виде диаграмма Ганта строится с помощью таких средств, как: MS Project, MS Visio, Primavera и прочее.

Разумеется, существует множество других, более совершенных программ, облегчающих управление проектами. Диаграмма Ганта любой сложности может быть легко построена с помощью таких приложений, как: SchedRoll; Gantt Designer; Mindjet JCVGantt Pro; и многих других.

Кроме того, существует целый ряд онлайн-сервисов, которые предоставляют своим пользователям возможность не только планировать свои дела, но и получать регулярные отчёты, уведомления о текущем статусе задач по электронной почте.

Рассмотрим основные принципы построения диаграммы:

Графически задачи отображаются сверху вниз слево направо.

Задача может состоять из нескольких подзадач, но не менее 2-х (на графике Задача 4. Разработка ПО и ее подзадачи 4.1. и 4.2.).

Задачи могут выполняться последовательно (Задачи 1. и 2.) и параллельно (Задачи 2. и 3.).

Загруженность исполнителей ставится с учетом выполнения работ во времени (Обратите внимание на Задачи 2. и 3.).

Этапы календарного планирования

1) Определение основных этапов проекта. Как правило, выделяется минимум три этапа проекта. Этапы можно обозначить, руководствуясь следующими принципами:

a. По времени – делить на примерно равные временные промежутки;

b. По характеру работ – делить проект на различные блоки работ, отличные друг от друга по содержанию и характеру;

c. По результатам – выделить значимые, измеряемые результаты.

2) Декомпозиция этапов на меньшие и конкретные работы (задачи). Если возможно, то максимально подробно, если нет – руководствуйтесь принципами из пункта 1, то есть, разделите каждый этап как минимум еще на 3 задачи.

3) Определение последовательности работ.

4) Определение логических связей. Могут быть ограничения, например, невозможно начать выполнять задачу, не закончив предыдущую, либо не закончив три предыдущих задачи одного блока.

5) Спланировать сроки выполнения задач с построением диаграммы Ганта, диаграмма будет рассмотрена ниже.

6) Определение потребности в ресурсах:

a. людские ресурсы - определить ответственных по каждой задаче;

b. машины и механизмы;

c. помещения;

d. материалы и так далее.

7) Определение стоимости ресурсов и трудозатрат.

8) Оптимизация диаграммы Ганта с учетом загрузки ресурсов. Например, первоначально вы не знали, кого назначите выполнять две параллельные задачи, а потом оказалось, что их может выполнить одно должностное лицо и никак иначе, соответственно, эти задачи станут последовательными.

Оптимизация диаграммы Ганта:

1) Рассмотрите возможность выполнения задач параллельно.

2) Определите крайние точки проекта и отдельных задач.

3) Установите связи для последовательных задач.

4) Задачи, которые можно сделать позже передвиньте в конец.

5) Определите критичный путь – последовательность задач, которая определяет длительность проекта, и рассмотрите каждую на возможность сокращения по времени.

Основные достоинства и недостатки описанного метода планирования и управления.

Главным преимуществом является графическая подача материала. Удобство работы с графиками Ганта – возможность чётко выделить и обозначить этапы работы над проектом и одновременное отображение мероприятий и сроков их выполнения. За счёт представления заданий в виде различных цветных полос все члены команды могут буквально с первого взгляда определить свои задачи.

Диаграммы Ганта являются отличным презентационным инструментом, который способен продемонстрировать ключевые приоритеты проекта. То есть, как только руководящие лица выделяют и распределяют каждый из имеющихся в наличии

ресурсов, команда моментально узнаёт об этом и следует дальнейшим указаниям. Данное свойство графика Ганта крайне полезно для руководителей высшего звена – используя его, можно намного проще подготовить подробный, ёмкий отчёт о состоянии выполнения различных проектов.

Тем не менее, как и у любого другого метода планирования, у диаграммы Ганта есть свои недостатки.

Ключевым понятием диаграммы Ганта является «Веха» - метка значимого момента в ходе выполнения работ, общая граница двух или более задач. Вехи позволяют наглядно отобразить необходимость синхронизации, последовательности в выполнении различных работ. Вехи, как и другие границы на диаграмме, не являются календарными датами. Сдвиг вехи приводит к сдвигу всего проекта. Поэтому диаграмма Ганта не является, строго говоря, графиком работ. И это один из основных её недостатков. Кроме того, диаграмма Ганта не отображает значимости или ресурсоемкости работ, не отображает сущности работ (области действия). Для крупных проектов диаграмма Ганта становится чрезмерно тяжеловесной и теряет всякую наглядность.

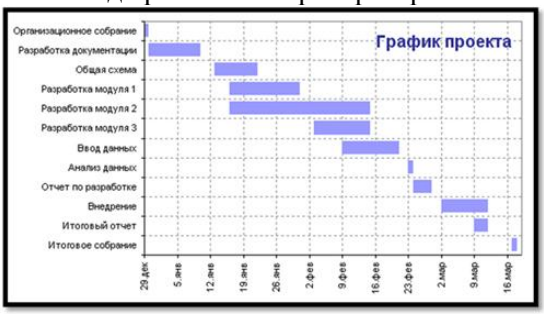
Недостатком является зависимость задач. Довольно часто в процессе презентации проектов у руководителей возникает необходимость показать, какие из указанных заданий связаны друг с другом. Но, к сожалению, сам формат диаграммы не позволяет сделать этого. Для того чтобы обойти это ограничение, менеджеры прибегают к различным хитростям: например, добавляют в график специальные вертикальные линии, которые демонстрируют ключевые зависимости. Однако это лишь временное решение, не способное передать информацию в полном объёме.

Ещё одним минусом графиков Ганта можно назвать их негибкость. В наши дни проекты не являются статичными – в них постоянно происходят какие-то изменения, сдвиги, учесть которые в диаграмме просто невозможно. Прежде чем приступать к построению графика, руководителям приходится просчитывать всё до мелочей, ведь при малейшем изменении оценки нужно перерисовывать «с нуля» всю диаграмму. И это не говоря уже о том, что возможность проиллюстрировать несколько разных способов планирования за один раз также отсутствует.

Вне зависимости от того, зачем вам нужна диаграмма Ганта, программа (даже самая «продвинутая») не сможет отобразить значимость и ресурсоёмкость тех или иных работ, их сущность. А

потому для особо масштабных проектов она используется крайне редко.

Указанные выше недостатки и ограничения серьёзно ограничивают область применения диаграммы. Тем не менее, в настоящее время диаграмма Ганта является стандартом де-факто в теории и практике управления проектами. В практике управления проектами данный метод чрезвычайно распространён.



Календарный план, план-график или диаграмма Ганта после построения становится реальным управленческим инструментом, во-первых, возможно видеть весь проект в виде одной схемы взаимоувязанных задач и не нужно держать в голове, во-вторых, на этом же графике вы отмечаете выполнение задач и видите отклонение от графика.

Задания для практического занятия:

1. Выполнить упражнения. Сохранить результаты на диске Е: в папке СТУДЕНТЫ.

2. Ответить на контрольные вопросы. Оформить в отчет.

Упражнения

Задание 1. Построение диаграммы Ганта. Стрелочная диаграмма.

1.Нарисуйте таблицу, в левый столбец которой занесите наименования выполняемых мероприятий. Наименования мероприятий следует расставлять сверху вниз в порядке их выполнения.

2.Выберите удобную периодичность контроля над выполнением занесенных в таблицу мероприятий и проставьте ее в верхней строке нарисованной таблицы. В качестве периодичности выполнения работ могут выступать недели, месяцы, кварталы и т.д.

3.В строке каждого мероприятия следует нарисовать стрелку, которая начинается в столбце запланированного срока начала выполнения этого мероприятия, а заканчивается в столбце

запланированного срока завершения выполнения рассматриваемого мероприятия.

Пример диаграммы Ганта:

Задание 2.

1. Сбор данных.

Для того чтобы построить график, понадобятся следующие данные:

- координаты всех наборов данных (откуда должен начинаться каждый из столбиков);
- название каждого этапа;
- продолжительность каждого этапа.

Для удобства вписываем их сразу в соответствующие поля таблицы. После того как Вы ввели всю требуемую информацию, можно переходить к созданию самой диаграммы.

Важно: проследите за тем, чтобы все форматы данных были указаны правильно: в частности, это касается дат.

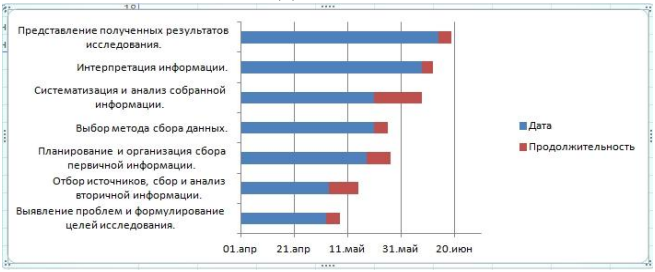
Особенность диаграммы Ганта в том, что ее столбики начинаются в произвольном месте, в то время как наиболее похожая на нее диаграмма – линейчатая диаграмма начинается с оси. Поэтому мы используем один из самых распространенных трюков с диаграммами – скроем какие-то данные.

Создадим таблицу.

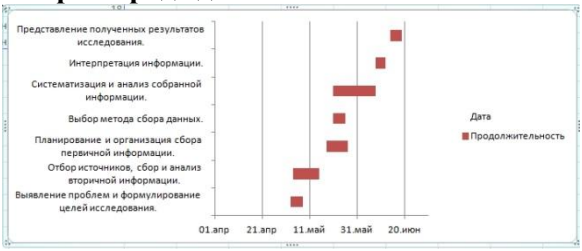
	A	B	C
1		Дата	Продолжительность
2	Выявление проблем и формулирование целей исследования.	03.май	5
3	Отбор источников, сбор и анализ вторичной информации.	04.май	11
4	Планирование и организация сбора первичной информации.	18.май	9
5	Выбор метода сбора данных.	21.май	5
6	Систематизация и анализ собранной информации.	21.май	18
7	Интерпретация информации.	08.июн	4
8	Представление полученных результатов исследования.	14.июн	5
9			

Обратите внимание, что не озаглавлены названия этапов, т.е. ячейка А1 пустая. Этот прием позволяет с первого раза в диаграмме определить, где находятся данные, а где подписи к ним.

Используем линейчатую диаграмму. Надо брать ее с накоплением, так как нужен второй ряд данных, который и используем в качестве основных данных.



Второй необходимый шаг – это скрытие первого ряда. Для этого делаем его невидимым. Щелкаем на синих данных, правой кнопкой мыши **Формат ряда данных/Заливка/Нет Заливки**



По умолчанию данные расположены в порядке снизу вверх. Исправляем это положение - кликаем на ось категорий (подписей) правой кнопкой мыши - **Формат оси/Параметры оси/Обратный порядок категорий**. Все, основная диаграмма готова, нужно теперь сделать еще кое-какое полезное форматирование:

- **Убираем легенду.** Она просто уже неактуальна для диаграммы Ганта. Выделяем и кнопкой Delete удаляем.
- **Определяем границы.** Excel сам определяет откуда начинаются значения, т.е. на рисунке сверху шкала начинается с первого апреля. Необходимо установить другую дату. Поэтому правой кнопкой мыши на оси дат **Формат оси/Параметры оси/Максимальное и минимальное значения**. На диаграмме это 3 мая (точнее, 28 апреля, понедельник) и 20 июня. Правда, есть нюанс, в Excel 2007 надо ставить числа в числовом формате.
- **Ставим недельные шкалы**. Там же в параметрах оси, ставим 7 в окне **цена основных делений**. Вот для чего потребовалось, чтобы первая дата стояла понедельником.

Форматируем, добавляем названия диаграммы:

Задание 3. Создание диаграммы Ганта в MS Visio

При помощи диаграммы Ганта можно составить расписание задач проекта, а затем отследить его ход.

1.В меню Файл последовательно выберите команды Создать, Расписание, а затем – команду Диаграмма Ганта.

2.На вкладке Дата введите количество задач для начала работы, единицы времени, которые нужно отобразить, и диапазон дат для проекта.

ПРИМЕЧАНИЕ. Параметры форматирования можно изменить в любой момент. В меню Диаграмма Ганта выберите команду Параметры, а затем перейдите на вкладку Формат.

3.Замените имена задач по умолчанию именами задач, соответствующими проекту, а также замените даты начала и окончания задач и длительность.

4.В диаграмме Ганта в столбце Название задачи выделите ячейку, содержащую задачу, которую необходимо переименовать, а затем введите новое имя.

5.В диаграмме Ганта выделите ячейку, содержащую дату, которую необходимо изменить, а затем введите новую дату.

ПРИМЕЧАНИЕ. Дату для итоговой задачи изменить нельзя. Даты итоговых задач изменяются только при изменении даты одной или нескольких задач, расположенных уровнем ниже итоговой задачи.

6.В диаграмме Ганта выделите ячейку, содержащую длительность, которую нужно изменить, а затем введите новое значение длительности. Используйте следующие сокращения: м - минуты, ч - часы, д - дни и н - недели.

ПРИМЕЧАНИЕ. Между числом и сокращением не должно быть пробела. Например, для указания длительности в 5 дней введите 5д.

СОВЕТ. Длительность можно также изменить, выделив область задач и перетащив маркер выделения ■ с правой стороны области задач на новую дату окончания на временной шкале.

7.Добавьте дополнительные задачи в диаграмму Ганта.

- Выделите рамку проекта, щелкнув сплошную линию вокруг диаграммы Ганта.

- Перетащите вниз маркер выделения ■, расположенный по центру нижней части рамки. Будут созданы новые строки задач, заполняющие пространство.

- Выделите ячейку Название задачи одной из новых задач, а затем введите имя задачи.

СОВЕТ. Положение задач в диаграмме Ганта можно изменить, перетащив строки задач внутри рамки диаграммы.

10.Добавьте вехи в диаграмму Ганта.

- Из набора элементов Фигуры диаграммы Ганта перетащите на страницу документа фигуру Веха и поместите ее между задачами, которые предваряет или за которыми следует веха.

ПРИМЕЧАНИЕ. При добавлении фигуры Веха в диаграмму Ганта автоматически добавляется строка со значением длительности, равным 0 (нулю).

- Введите имя и дату для вехи.

СОВЕТ. Задачу в вехе можно изменить, задав значение ее длительности равным 0. Подобным образом можно изменить веху в задаче, задав положительное значение длительности.

11.Создайте зависимости между задачами в диаграмме Ганта.

- Сначала выделите область задач или веху, из которых нужно установить связь, а затем нажмите клавишу SHIFT и выделите зависимую задачу или веху.

- Щелкните правой кнопкой мыши одну из фигур, а затем в контекстном меню выберите команду Связать задачи.

Задание 4. Имитация диаграммы Ганта

Этапы имитации диаграммы Ганта:

1. Для имитации диаграммы Ганта необходимо ввести данные на листе рабочей книги Microsoft Office Excel (например, в соответствии с таблицей 3)

	A	B	C
1	Мероприятия	Начало, день	Длительность, дней
2	Мероприятие 1	0	2
3	Мероприятие 2	2	3
4	Мероприятие 3	2	5
5	Мероприятие 4	4	2
6	Мероприятие 5	5	3
7	Мероприятие 6	8	2
8			

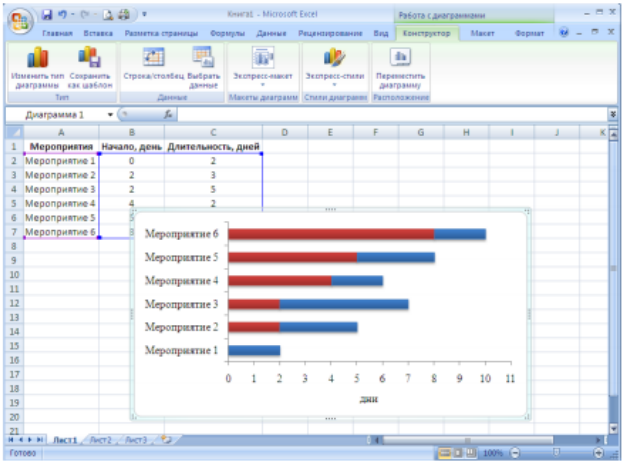
Таблица 3 – Исходные данные для построения диаграммы Ганта

№	Мероприятия	Начало, день	Длительность, дней
1	Мероприятие 1	0	2
2	Мероприятие 2	2	3
3	Мероприятие 3	2	5
4	Мероприятие 4	4	2
5	Мероприятие 5	5	3
6	Мероприятие 6	8	2

2. Выберите данные, которые нужно показать на диаграмме Ганта.

3. На вкладке Вставка в группе Диаграммы щелкните Линейчатая.

4. В группе Плоская линейчатая диаграмма выберите Линейчатая диаграмма с накоплением.



5. Щелкните на диаграмме область диаграммы, при этом появится панель Работа с диаграммами с вкладками Конструктор, Макет и Формат.

6. Выберите на диаграмме значения интервалов относящие к группе значений первого столбца, а далее измените его Заливку выберите вариант Нет заливки.

7. Таким образом, получаем имитацию диаграммы Ганта.

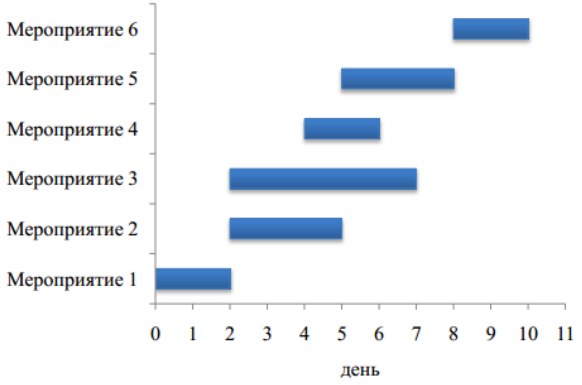


Диаграмма реализации мероприятий

Вопросы для закрепления теоретического материала к практическому занятию:

1. Для чего нужна диаграмма Ганта?
2. История появления первого графика
3. Диаграмма Ганта в современном мире
4. Другие программы для создания графика
5. основные принципы построения диаграммы
6. Преимущества и недостатки метода
7. Что такое календарное планирование
8. Где применимо календарное планирование

Порядок выполнения отчета по практической работе

1. Изучить теоретический материал. Ответить на контрольные вопросы.

2. Оформить работу в соответствии с шаблоном (Приложение 1). При оформлении использовать MS Office.

3. Сохранить результаты работы в соответствующей папке на диске E:.

Коллективное создание программного продукта.

Цель работы состоит в приобретении навыков работы в составе бригады при разработке программного продукта

Краткие теоретические и учебно-методические материалы по теме практической работы

В настоящее время сложность промышленных приложений и систем такова, что процесс их разработки стал практически неуправляемым. Кроме того, их развертывание на сотнях компьютеров, расположенных в разных местах, значительно раздвигает границы процесса разработки.

Один человек не способен создать приложение масштаба предприятия. Ни один разработчик просто не удержит в голове все требования к системе и варианты проекта. Поэтому сегодня разработкой промышленных систем занимаются проектные группы, и все обязанности распределяются среди членов группы.

Существует две основные модели организации коллектива при разработке ПО:

1) иерархическая модель;

2) модель группы.

Работа в коллективе отличается определенными сложностями. Иерархическая модель организации, определяет начальников и подчиненных. Однако если в современных производственных средах один менеджер проекта отвечает за все тонкости разработки и принимает все важные решения, возникает множество проблем, ведущих к провалу проекта. Опыта одного человека чаще всего недостаточно для быстрого решения задачи и для интеграции приложения в существующую инфраструктуру.

В организациях, построенных на основе иерархической модели, затруднен обмен информацией - в этой модели он, по определению, осуществляется через посредников. Вся информация иерархических групп «фильтруется» тремя или четырьмя менеджерами, что значительно повышает вероятность утери самого важного. Часто такое отсеивание идей происходит при прохождении сообщения от разработчика, непосредственно занимающегося проектом, к высшему руководству. Естественно, некоторые участники «выпадают» из процесса, что снижает эффективность их труда и повышает вероятность провала проекта.

Чтобы сгладить недостатки иерархической модели, в проектной группе предусматривается распределение обязанностей руководителя между членами коллектива. При этом за проект отвечает не один человек, а все члены группы - каждый за свой участок.

Модель группы не определяет структуру коллектива с точки зрения отдела кадров. Ведь в такую разностороннюю группу привлечены ресурсы из разных отделов организации. Задача модели проектной группы - определить цели проекта и распределить обязанности. Руководители каждого направления с помощью выделенных им ресурсов выполняют возложенную на них часть работы. Обязанности ролей определяются работой над проектом, а не деятельностью «штатной единицы». При этом руководители направлений выполняют свои обычные функции: составляют график выплаты премий, распределяют отпуска и контролируют эффективность работы сотрудников. Начальник может оценить степень участия и эффективность работы сотрудников в проектной группе, но это - прерогатива менеджера конкретного сотрудника, а не проектной группы.

Для достижения целей в модели проектной группы выполняемые задачи распределяются по шести ролям: менеджмент продукта, менеджмент программы, разработка, тестирование, обучение пользователей и логистика. Люди, выполняющие конкретную роль, должны обладать необходимой для этого квалификацией.

В этой модели нет руководителя всего проекта - есть группа людей, знающих, что нужно делать и делающих это.

Иерархическая модель.

Недостатки:

- нехватка информации;
- невозможностью учесть все особенности проекта;
- отсутствие полноценной связи между всеми участниками проекта, так как вся информация идет в одном направлении - вверх по иерархии, к главному менеджеру;
- трудность освоения новых технологий, необходимых при создании кроссплатформенных приложений;
- сложность расстановки приоритетов.

Кроме того, опыта одного человека чаще всего недостаточно для быстрого решения задачи и для интеграции приложения в существующую инфраструктуру.

В организациях, построенных на основе иерархической модели, затруднен обмен информацией - в этой модели он, по определению, осуществляется через посредников. Вся информация иерархических групп «фильтруется» тремя или четырьмя менеджерами, что

значительно повышает вероятность утери самого важного. Часто такое отсеивание идей происходит при прохождении сообщения от разработчика, непосредственно занимающегося проектом, к высшему руководству. Естественно, некоторые участники «выпадают» из процесса, что снижает эффективность их труда и повышает вероятность провала проекта.

Чтобы сгладить недостатки иерархической модели, в проектной группе предусматривается распределение обязанностей руководителя между членами коллектива. При этом за проект отвечает не один человек, а все члены группы - каждый за свой участок.

Модель группы не определяет структуру коллектива с точки зрения отдела кадров. Ведь в такую разностороннюю группу привлечены ресурсы из разных отделов организации. Задача модели проектной группы - определить цели проекта и распределить обязанности. Руководители каждого направления с помощью выделенных им ресурсов выполняют возложенную на них часть работы. Обязанности ролей определяются работой над проектом, а не деятельностью «штатной единицы». При этом руководители направлений выполняют свои обычные функции: составляют график выплаты премий, распределяют отпуска и контролируют эффективность работы сотрудников. Начальник может оценить степень участия и эффективность работы сотрудников в проектной группе, но это - прерогатива менеджера конкретного сотрудника, а не проектной группы.

Коллективный подход.

Недостатки:

- разрозненная связь с внешними источниками информации;
- несогласованное представление о разных сторонах проекта;
- несогласованность личных планов членов группы;
- отсутствие опыта, снижающее эффективность коллективной работы.

Обязанности членов группы. Предлагается подход MFS.

MSF - не готовое решение, а каркас, который можно адаптировать для нужд любой организации. Один из элементов этого каркаса - *модель проектной группы.* Она описывает структуру группы и принципы, которым надо следовать для успешного выполнения проекта.

Хотя модель группы разработчиков весьма конкретна, при знакомстве с MSF ее нужно рассматривать в качестве отправной точки. Разные коллективы реализуют этот каркас по-разному, в зависимости от масштаба проекта, размеров группы и уровня подготовки ее членов.

Чтобы проект считался удачным, следует решить определенные задачи:

- **удовлетворить требования заказчика** - проект должен выполнить требования заказчиков и пользователей, иначе ни о каком успехе не может быть и речи, возможна ситуация, когда бюджет и график соблюдены, но проект провалился, так как не выполнены требования заказчика;
- **соблюсти ограничения** - разработчики проекта должны уложиться в финансовые и временные рамки;
- **выполнить спецификации, основанные на требованиях пользователей** - спецификации - это подробное описание продукта, создаваемое группой для заказчика; они представляют собой соглашение между проектной группой и клиентом и регулируют вопросы, касающиеся приложения, в основе этого требования лежит принцип «сделать все, что обещано»;
- **выпустить продукт только после выявления и устранения всех проблем** - не существует программ без дефектов, однако группа должна найти и устранить их до выпуска продукта в свет, причем устранением ошибки считается не только ее исправление, но и, например, занесение в документацию способа ее обхода; даже такой способ устранения проблем предпочтительнее, чем выпуск приложения, содержащего не выявленные ошибки, которые в любой момент могут преподнести неприятный сюрприз пользователям и разработчикам;
- **повысить эффективность труда пользователей** - новый продукт должен упрощать работу пользователей и делать ее более эффективной. Поэтому приложение, обладающее массой возможностей применять которые сложно или неудобно, считается провальным;
- **гарантировать простоту развертывания и управления** - эффективность развертывания непосредственно влияет на оценку пользователем качества продукта, например, ошибка в программе установки может создать у пользователей впечатление, что и само приложение небезгрешно, от проектной группы требуется не только подготовить продукт к развертыванию и гладко провести его, но и

обеспечить пользователей поддержкой, организовав сопровождение приложения.

Для достижения этих целей в модели проектной группы выполняемые задачи распределяются по шести ролям: менеджмент продукта, менеджмент программы, разработка, тестирование, обучение пользователей и логистика. Люди, выполняющие конкретную роль, должны обладать необходимой для этого квалификацией.

Шесть ролей модели проектной группы проиллюстрированы в таблице. Все эти цели важны для успеха проекта в целом, и поэтому все роли равноправны. В этой модели нет руководителя всего проекта - есть группа людей, знающих, что нужно делать и делающих это.

Таблица - Цели и роли

Цель	**Роль**
Удовлетворение требований заказчика Соблюдение ограничений проекта Соответствие спецификациям Выпуск только после выявления и устранения проблем Повышение эффективности труда пользователя Простота развертывания и постоянное сопровождение	Менеджер продукта Менеджер программы Разработчик Тестер Инструктор Логистик

Как начать работу над проектом, не зная, сколько времени на это потребуется, сколько проект будет стоить и каких результатов ожидать? Ответить на эти вопросы поможет модель проектной группы MSF и модель процесса разработки. Обе эти модели предлагают составлять расписания и графики «снизу - вверх», в проектировании придерживаться подхода «сверху - вниз» и, кроме этого, распределять ответственность за результаты работы между членами группы. Приняв на этих стадиях правильное решение, можно избежать серьезных изменений в дальнейшем, что намного сократит время выполнения проекта и затраты на него.

Модель проектной группы

В проектную группу должны входить:

- опытные руководители;
- инициативные сотрудники, способные принимать решения и нести ответственность за свое направление работы.

Их задача:

- сконцентрироваться на выпуске продукта;
- выработать общее представление о проекте.

Для эффективной работы проектной группы требуется соблюдение следующих правил в отношениях между людьми:

- **доверие** - делает действия людей согласованными, при этом все
- следуют принципу «мы делаем то, что обещали сделать»;
- **уважение** - люди признают способности других, следуя правилу

Рисунок 1 - Роли в модели проектной группы

- «каждый из нас необходим нашей группе»;
- **согласие** - все должны знать и поддерживать цели проекта и верить в его успех - «мы завершим проект, и точка»;
- **ответственность** - люди должны ясно понимать цели проекта, свои обязанности и чего от них ожидают - «я сделаю свою работу, вы - свою, и к четвергу мы построим наш дом».

Менеджер продукта

Менеджер продукта должен вовремя реагировать на потребности заказчика. Его главная задача - сформировать общее представление о поставленной задаче и о том, как ее решать. Он должен ответить на вопрос «Зачем мы делаем все это?» и убедиться, что все члены группы знают и понимают ответ на него.

Основная цель этой роли - удовлетворение требований заказчика. Для этого менеджер продукта выступает представителем заказчика в группе разработчиков и представителем группы у

заказчика. (На этом этапе важно понимать разницу между заказчиком и пользователем: заказчик платит за создание продукта, а пользователи с ним работают.) Кроме того, менеджер продукта вместе с менеджером программы должны прийти к компромиссному решению относительно функциональных возможностей продукта, сроков его разработки и финансирования проекта.

Менеджер программы

Задача менеджера программы - вести процесс разработки с учетом всех ограничений. Руководитель этого направления должен понимать разницу между понятиями «руководитель» и «начальник». Главная обязанность менеджера программы - выполнить все стадии разработки так, чтобы нужный продукт был выпущен в нужное время. Он координирует деятельность других членов группы. И хотя иногда ему придется подгонять своих сотрудников, он не должен и помышлять о диктаторском стиле управления.

Главный менеджер программы составляет график проекта на основе информации, полученной от остальных членов группы. Он координирует этот график с руководителями всех подгрупп. Буферным временем проекта также управляет менеджер программы. Если отдельные части работы выполняются раньше графика или, наоборот, задерживаются, именно менеджер программы должен выяснить, как это скажется на проекте, и изменить график.

Менеджер программы отвечает и за бюджет проекта, объединяя требования к ресурсам всех членов группы в единый план расходов. Естественно, его задача - не только разобраться в этих требованиях, но и контролировать реальные затраты, сравнивая их с запланированными. Кроме того, менеджер программы должен регулярно сообщать о состоянии работы всем основным участникам проекта.

Разработчик

Разработчики знакомят остальных членов группы с применяемыми технологиями и собственно создают продукт. В качестве консультантов они предоставляют исходные данные для проектирования, проводят оценку технологий, а также разрабатывают прототипы и тестовые системы, необходимые для проверки решений и сокращения рисков на ранних стадиях процесса разработки. Чтобы создать продукт определенного качества, разработчикам не следует замыкаться на создании кода, они должны участвовать и в решении прикладной задачи. Они творят не ради творчества, а для реализации требований заказчика. Часто, чтобы

полностью разобраться в проекте, приходится создавать прототипы, а чтобы протестировать новую технологию, - испытательные системы, помогающие принять окончательное решение относительно архитектуры приложения. Этим также занимаются разработчики.

Разработчики сами оценивают сроки своей работы. Разработчики отвечают и за техническую реализацию проекта - в основном на фазах создания логической и физической модели. На этих стадиях их задача - определить методы реализации функциональных возможностей и заданной архитектуры, а также оценить сроки выполнения этой работы. Заметим, что разработчики не выбирают функции - они только решают, как их реализовать.

Кроме того, на стадии «Планирование» разработчики решают, какое влияние окажет на проект добавление или удаление некоторых функций. Разработчики не участвуют в заключительной стадии проекта - развертывании продукта, однако они должны тесно сотрудничать с логистиками на стадии установки приложения.

Тестер

Задача тестеров - испытание продукта в реальных условиях, дабы определить, что в продукте работает и что не работает, и нарисовать таким образом точный «портрет» приложения. Естественно, для проведения тестов нужно отлично разбираться и в требованиях пользователей, и в том, как их удовлетворить.

Тестеры разрабатывают стратегию, планы, графики и сценарии тестирования, которые позволяют убедиться, что все ошибки выявлены и исправлены до выпуска приложения. Ошибкой называют любую проблему, из-за которой продукт не выполняет свои функции. Ею может оказаться и ошибка в коде, называемая «жучком», и отклонение от спецификаций, заданных менеджером программы, и недоработки в документации, подготовленной группой обучения пользователей.

Нельзя совмещать должности тестера и разработчика. Разделение этих обязанностей:

- гарантирует независимую проверку того, что продукт действительно выполняет все требования;
- повышает качество продукта за счет конкуренции между группами.

Хотя проверяют качества продукта только тестеры, за выпуск хорошего продукта отвечают все члены проектной группы.

Контроль изменений

При работе над проектом необходимо контролировать изменения, им должны заниматься все участники группы, но чаще всего в полном объеме этим приходится заниматься именно группе тестирования. Для управления изменениями необходимо:

- создать эталонный документ;
- определить изменяемые элементы;
- определить влияние изменений на существующие системы, •процессы или документы;
- определить метод реализации изменений;
- назначить человека, который внесет изменения;
- определить влияние изменения на условия выполнения проекта, •его бюджет, график и политику;
- получить одобрение изменений (скажем, у руководителя проекта);
- внести изменения;
- сообщать новый документ, в котором изменение учтено.

Прочие обязанности

Некоторые важные обязанности тестеров часто упускают из виду. К ним относятся:

- **уведомление об ошибках и их отслеживание** - тестовая группа отвечает не только за управление изменениями, но и за систему выявления ошибок и информирования о них;
- **сборка продукта** - в группе должен быть человек, ответственный за сборку (компиляцию) продукта, и часто такой «главный сборщик» является тестером, он может использовать только код, хранящийся в системе управления версиями; эту рутинную работе удается автоматизировать с помощью сценариев, однако необходимо проверять правильность сборки;
- **выявление и контроль рисков** - это обязанность всех членов группы, менеджер программы должен разработать метод контроля.

Инструктор

Цель группы обучения - повысить эффективность труда пользователей. Поэтому инструкторы «принимают сторону» пользователей подобно тому, как менеджеры продукта представляют интересы заказчика. Однако перед пользователями инструкторы выступают в роли представителей проектной группы.

В этом последнем качестве группа обучения отвечает за выпуск удобного, полезного продукта, которому практически не нужна поддержка. Персонал группы тестирует удобство использования продукт та, выявляет проблемы в этой области и проверяет проект пользовательского интерфейса.

Активно участвуя в создании пользовательского интерфейса, инструкторы сокращают затраты на сопровождение продукта и поддержку пользователей.

Логистик

Логистик представляет интересы служб поддержки и сопровождения, справочных служб и других служб канала доставки. Он занимается развертыванием продукта и его сопровождением и контролирует продукт с этой точки зрения в процессе проектирования. Кроме того, его задача - составление графиков развертывания приложения. Логистики, менеджеры продукта и менеджеры программы совместно определяют порядок передачи продукта пользователям и организации, после чего логистики готовят их к развертыванию приложения.

Размер группы логистики определяется графиком развертывания, уровнем автоматизации установки, наличием средств автоматического развертывания приложений и другими характеристиками этого процесса.

Размеры группы и масштаб проекта

В проектной группе за каждое направление должен отвечать как минимум один человек. При реализации крупного проекта возникает затруднение, связанное с эффективным обменом информацией. В небольших организациях или при работе над мелкими проектами роли можно совмещать.

Крупные проекты

Чтобы справиться с крупным проектом, необходимо делить проектную группу на тематические и функциональные подгруппы.

Тематические группы

Это небольшие подгруппы из одного или нескольких человек, роли которых различны. Каждой из таких групп выделяется некий набор функциональных возможностей приложения, за все стороны проектирования и разработки которого она и отвечает (включая составление проекта и графика реализации). Например, какой-либо группе нужно предоставить решать задачу вывода данных на печать.

В такой группе высока ответственность. Ее члены могут обратиться ко всем людям, опыт которых необходим в их работе.

Поэтому, если они так и не найдут оптимального решения, в этом они смогут винить только себя. Такая группа сбалансирована. Вряд ли вы захотите, чтобы окончательные спецификации создавал только отдел разработки, маркетинга или контроля качества. Только решение, принятое группой представителей каждого из этих отделов, будет по-настоящему сбалансировано.

Функциональные группы

Функциональные группы формируются в рамках одной роли. Они нужны в очень крупных проектных группах или при работе над крупномасштабными проектами, когда отдельные роли нуждаются в дополнительном подразделении. Например, в Microsoft отдел менеджмента продукта обычно состоит из групп планирования и маркетинга. Обе они занимаются менеджментом продукта, но первая отвечает за определение действительно необходимых заказчику функций приложения, а вторая - за информирование потенциальных клиентов о достоинствах продукта.

Небольшие проекты

Некоторые должности можно совмещать. Конечно, основной смысл такого разделения в том, чтобы каждую из шести задач решал один из членов группы. Однако, не во всех проектах это возможно.

В небольших группах один человек может играть несколько ролей. При этом мы рекомендуем соблюдать следующие принципы разделения должностей.

- *Нельзя совмещать разработку с другими видами деятельности* - ее создателей приложения не стоит отвлекать от основной задачи. Если «повесить» на разработчиков дополнительные обязанности, то скорее всего график работ будет нарушен, а дату выпуска продукта придется отодвинуть.
- *Конфликт интересов* — нельзя совмещать роли, интересы которых противоположны. Пример - менеджер продукта и менеджер программы. Первый хочет выполнить все требования заказчика, второму же надо уложиться в график и бюджет. Если совместить эти роли, возникает опасность упустить просьбу заказчика о внесении изменений в проект либо, напротив, принять их без должного анализа влияния на график работ. Таким образом, назначение на эти роли разных людей позволяет соблюсти интересы всех участников проекта.

Создание группы

Модель проектной группы описывает структуру группы для работы над проектом создания приложений масштаба предприятия. Однако одной структуры недостаточно - важным фактором успеха является квалификация членов группы.

Поиск руководителей

Главная задача человека, ответственного за создание проектной группы, - подобрать квалифицированных исполнителей. Эта кажущаяся простой (но на самом деле сложная) задача имеет огромное значение для успеха всего проекта.

Найти лидеров - несложная проблема; в любой организации они всем известны. Важно понимать, что говорится именно о лидерах, а не начальниках. Конечно, в любой организации есть менеджеры директора и так далее, но положение в иерархической структуре далеко не всегда гарантирует наличие качеств лидера. Лидеров определяют действия и качества, а не должности.

Руководители должны обладать:

- умением понимать и помогать;
- коммуникабельностью;
- авторитетом внутри организации и за ее пределами;
- чувством ответственности за поставленные цели;
- умением принимать конструктивные решения;
- уверенностью в своих силах;
- достаточной для решения поставленных задач квалификацией;
- способностью помочь другим развить свои таланты и приобрести опыт.

Настоящего лидера отличают именно эти *способности, а* не *намерения.* Это относится и к качествам руководителя.

Повышение эффективности коллективной работы

- заинтересованность;
- надежда, оптимизм, готовность к работе;
- определение задач и решений;
- проявление взаимопомощи;
- доверительные уважительные отношения;
- единение.

Последнее очень важно: эффективность работы группы при этом выше всего.

Для успеха проекта недостаточно только распределить роли и обязанности. Помимо структуры, следует придерживаться определенных принципов и методов

Общее представление о проекте

Важнейший фактор для успеха проекта - единое понимание целей и задач проекта всеми участниками. Каждый из них изначально имеет **свое** мнение, касающееся приложения. В процессе формирования общего представления о проекте все эти мнения обсуждаются, что позволяет добиться единства целей всех членов проектной группы и заказчика.

На основе выработанного представления о проекте создается документ «Концепция проекта», который:

- описывает не только то, что делает продукт, но и то, чего он не делает;
- конкретизирует продукт (например, позволяет включать и исключать определенные функциональные возможности из данной версии); •побуждает группу достичь сформулированной цели;
- содержит описание путей реализации проекта, благодаря чему проектная группа и заказчик могут начать работу.

Задания для практического занятия:

1.Распределение ролей в бригаде (см. приложение).

2.Выполнить работу в соответствии с заданием преподавателя.

3.Оформить отчет

Вопросы для закрепления теоретического материала к практическому занятию:

1. Основные модели организации коллектива при разработке ПО:
2. Недостатки коллективного подхода.
3. Обязанности членов группы.
4. Модель проектной группы. Цели и роли
5. Задачи проектной группы.

Порядок выполнения отчета по практической работе

1. Защита отчета по практической работе заключается в предъявлении преподавателю полученных результатов (на экране монитора и печатном виде), демонстрации полученных навыков и ответах на вопросы преподавателя.

2. Ответить на контрольные вопросы.

Рекомендуемая литература

Основные источники:

1. Зубкова Т.М. Технология разработки программного обеспечения: учебное пособие /Т.М. Зубкова; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего образования «Оренбургский государственный университет», Кафедра программного обеспечения вычислительной техники и автоматизированных систем. - Оренбург: ОГУ, 2017. - 469 с.: ил. - Библиогр.: с. 454-459. - ISBN 978-5-7410-1785-2; [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=485553.

Дополнительные источники:

1. Антонов В.Ф. Методы и средства проектирования информационных систем: учебное пособие / В.Ф. Антонов, А.А. Москвитин; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Северо-Кавказский федеральный университет». - Ставрополь: СКФУ, 2016. - 342 с.: ил. - Библиогр. в кн.; [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=458663.
2. Введение в программные системы и их разработку / С.В. Назаров, С.Н. Белоусова, И.А. Бессонова и др. - 2-е изд., испр. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 650 с. : схем., табл., ил. - Библиогр. в кн.; [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=429819.
3. Долженко А.И. Технологии командной разработки программного обеспечения информационных систем / А.И. Долженко. - 2-е изд., исправ. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 301 с.: схем., ил. - Библиогр. в кн.; [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428801.
4. Митина О.А. Методы и средства проектирования информационных систем и технологий: курс лекций / О.А. Митина; Министерство транспорта Российской Федерации. -

Москва: Альтаир: МГАВТ, 2016. - 76 с.: ил. - Библиогр. в кн.; [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=482395.
5. Сафонов В.О. Развитие платформы облачных вычислений Microsoft Windows Azure / В.О. Сафонов. - 2-е изд., испр. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 393 с.: ил.; [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428823.

Интернет-ресурсы:

1. Компьютерные книги. Режим доступа: [http://computers.plib.ru/programming/Books.VBasic6/index.html 09.04.2020];
2. Технология программирования. Электронное пособие по дисциплине "Технология Программирования". Чернев Дмитрий Алексеевич. Режим доступа: [http://www.tehprog.ru 09.04.2020];
3. Все для программиста! Режим доступа: [http://www.codenet.ru/ 09.04.2020];
4. On-line библиотека свободно доступных материалов по информационным технологиям. Режим доступа: [http://digitland.ru 09.04.2020].