**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

**Курсовая работа**

по дисциплине «Алгоритмы и алгоритмические языки» на тему:

Технология разработки программного обеспечения

Выполнил:

студент группы БФИ2202

Пляго П.А.

Научный руководитель:

Доцент кафедры «Информатика»

к.п.н. Гуриков С. Р.

Москва, 2023 г.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc136286321)

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 5](#_Toc136286322)

[1. Введение 5](#_Toc136286323)

[2. Основание для разработки 5](#_Toc136286324)

[3. Назначение разработки 5](#_Toc136286325)

[4. Требования к программе или программному изделию 5](#_Toc136286326)

[4.1 Требования к функциональным характеристикам 5](#_Toc136286327)

[4.2 Требования к надежности 6](#_Toc136286328)

[4.3 Условия эксплуатации 6](#_Toc136286329)

[4.4 Требования к составу и параметрам технических средств 7](#_Toc136286330)

[4.5 Требования к информационной и программной совместимости 7](#_Toc136286331)

[4.6 Требования к маркировке и упаковке 8](#_Toc136286332)

[4.7 Требования к транспортированию и хранению 8](#_Toc136286333)

[5. Требования к программной документации 9](#_Toc136286334)

[6. Технико-экономические показатели 9](#_Toc136286335)

[7. Стадии и этапы разработки 9](#_Toc136286336)

[8. Порядок контроля и приемки 9](#_Toc136286337)

[ГЛАВА 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 11](#_Toc136286338)

[1.1 Основные понятия и определения 11](#_Toc136286339)

[1.2 Типы программного обеспечения 15](#_Toc136286340)

[1.3 Жизненный цикл и этапы разработки программного обеспечения 26](#_Toc136286341)

[1.4 Модели жизненного цикла ПО 36](#_Toc136286342)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ 48](#_Toc136286343)

[2.1. Разработка электронного конспекта 48](#_Toc136286344)

[2.2 Разработка тестовой программы 49](#_Toc136286345)

[2.3. Дополнительные возможности программы 62](#_Toc136286346)

[ЗАКЛЮЧЕНИЕ 74](#_Toc136286347)

[СПИСОК ИМПЛЬЗОВАННЫХ ИСТОЧНИКОВ 75](#_Toc136286348)

[ПРИЛОЖЕНИЕ A 76](#_Toc136286349)

[ПРИЛОЖЕНИЕ B 80](#_Toc136286350)

# ВВЕДЕНИЕ

Современный мир зависит от программного обеспечения, которое используется во многих областях жизни, включая бизнес, образование, медицину и технологии. Также стоит упомянуть, что разработка программного обеспечения имеет прямое отношение к экономическим и социальным аспектам. Она может повышать производительность и эффективность бизнеса, улучшать качество жизни людей и даже способствовать развитию науки и медицины, поэтому развитие данной сферы приобрело **особую актуальность**.

Цель изучения разработки программного обеспечения может быть различной в зависимости от индивидуальных потребностей и целей каждого человека. Некоторые люди могут изучать программирование для того, чтобы создавать собственные приложения и веб-сайты, другие - для поиска работы в IT-секторе, а еще другие - для продвижения в своей текущей работе. Независимо от цели, изучение разработки программного обеспечения может предоставить ряд преимуществ, включая следующие:

* Создание собственных приложений или веб-сайтов для личного использования или коммерческих целей
* Развитие навыков аналитического мышления, логики и решения проблем
* Увеличение шансов на поиск работы в IT-секторе, который является одной из наиболее динамично развивающихся отраслей современной экономики
* Развитие карьеры и продвижение внутри компании, особенно если ваша компания имеет IT-отдел или работает с технологическими продуктами и услугами.

В целом, технология разработки программного обеспечения может столкнуться с различными проблемами, но их можно решить с помощью опыта, квалификации и профессионализма.

Кроме того, существует постоянный рост и развитие технологий, которые используются в разработке программного обеспечения, поэтому исследуемая мной тема **представляет собой актуальную научную задачу,** Новые языки программирования, инструменты, платформы и методологии появляются каждый год, и разработчики должны постоянно обновлять свои знания и навыки, чтобы оставаться конкурентоспособными.

**Объект исследования**,проведенного в рамках данной работы, является средства и возможности технологий разработки программного обеспечения.

**Предметом исследования**, является современные технологии разработки программного обеспечения.

**Цели работы и задачи исследования.** Цель работы – исследование основ проектной деятельности и тестирование студентов по теме «Технологий разработки программного обеспечения».

Поставленная цель определила следующие основные задач исследования:

1. Изучение и анализ знаний, умений и навыков, которые необходимы для работы в сфере разработки программного обеспечения.
2. Разработка программы, содержащей теоретический материал и тест.

**Методы исследования.** Определение сути изучаемой проблемы, прослеживание истории появления и развития технологий разработки программного обеспечения. Анализ учебной литературы по теме: «Технология разработки программного обеспечения». Зубкова Т.М.

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

# Введение

Областью применения разрабатываемого программного продукта является углубленное изучение, а также контроль качества усвоения студентами основных определений, принципов и особенностей, используемых в сфере технология разработки программного обеспечения.

# Основание для разработки

* + Лист утверждения тем курсовых работ (протокол №5 заседания кафедры от 01.02.2023 г.), подписанный научным руководителем: доцентом кафедры «Информатика» Гуриковым С.Р.;
  + Наименование разработки  «Технология разработки программного обеспечения».

# Назначение разработки

Разрабатываемый программный продукт предназначен для изучения и, в последующем, контроля качества усвоения студентами основных определений, принципов и методов, используемых в сфере применения технологий разработки программного обеспечения.

# Требования к программе или программному изделию

# 4.1 Требования к функциональным характеристикам

Программная разработка должна быть написана в четком соответствии с материалами занятий по дисциплине «Алгоритмы и алгоритмические языки». Структуру программного кода тестовых вопросов изменять нельзя, она должна соответствовать материалам занятий.

Студенту, за счет часов самостоятельной работы, разрешается заниматься дополнительным поиском информации с целью расширения возможностей своей программной разработки, с последующим описанием их в пояснительной записке.

Дополнительные возможности программы:

1. Запись результатов тестирования (в формате Верно/Неверно) в одномерный массив с выводом в DataGridView;
2. Создание системы авторизации пользователей, с использованием базы данных;
3. Построение диаграммы с отображением результатов тестирования;
4. Запись результатов тестирования (в формате 1/0) в одномерный массив с выводом в DataGridView и возможность удаления введенного элемента через InputBox;
5. Запись результатов тестирования (в формате 1/0) в Блокнот;
6. Запись результатов тестирования (в формате 1/0) в Excel, построение диаграммы и сохранение ее скриншота в формате .jpg;
7. Создание панели ToolStrip, при нажатии на кнопку «открыть» будет возможность открыть скриншот диаграммы из Excel;
8. Запись результатов тестирования (в формате 1/0) в pdf при нажатии на кнопку сохранить на панели ToolStrip;
9. Создание анимационной заставки;
10. Запись сортировки простым выбором результатов тестирования (в формате 1/0) с выводом в DataGridView;

# 4.2 Требования к надежности

Разрабатываемое программное обеспечение должно быть спроектировано таким образом, чтобы обеспечить защиту и надежную работу при наличии ошибок во входных данных и/или от некорректных действий пользователя - предполагается, что программный продукт должен быть спроектирован таким образом, чтобы внутренняя или внешняя (некритическая для системы) ошибка не приводила к аварийной остановке.

# 4.3 Условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

Программа будет работать в температурном режиме от + 5 до + 35 °C при относительной влажности 90 % и атмосферном давлении 462 мм.рт.ст., поскольку такие условия приблизительно соответствуют условиям эксплуатации современных компьютеров непромышленного исполнения.

# 4.4 Требования к составу и параметрам технических средств

Для корректной работы программного продукта вычислительная система должна обладать следующими характеристиками:

* процессор с тактовой частотой не ниже 1ГЦ;
* оперативная память объемом не менее 4ГБ ОЗУ;
* периферийные устройства: клавиатура, мышь;
* монитор с разрешающей способностью не ниже 1280х1024; .
* жесткий диск объемом не менее 500МБ;
* желательно наличие принтера для печати отчета итогов работы программы.

Для корректной работы вычислительной среды необходимо наличие системного программного обеспечения, основным элементом которого является операционная система. В связи с тем, что Microsoft является лидером в сегменте продаж операционных систем, то целесообразнее использовать операционную систему семейства Windows не ниже Windows 8.

# 4.5 Требования к информационной и программной совместимости

Требования к информационным структурам (файлов) на входе и выходе не предъявляются.

Исходные коды программы должны быть реализованы на языке Visual С#. В качестве интегрированной среды разработки программы должна быть использована среда MS Visual Studio.

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы.

# 4.6 Требования к маркировке и упаковке

Программа поставляется в виде программного изделия - внешнем флеш- носителе.

Программное изделие должно иметь маркировку с обозначением товарного знака компании-разработчика, типа (наименования), номера версии, порядкового номера, даты изготовления и номера сертификата соответствия Госстандарта России (если таковой имеется). Маркировка должна быть нанесена на программное изделие в виде наклейки, выполненной полиграфическим способом с учетом требований ГОСТ 9181-74.

Упаковка флеш-носителя - пакет для хранения.

Упаковка программного изделия должна проводиться в закрытых вентилируемых помещениях при температуре от плюс 15 до плюс 40 °С и относительной влажности не более 80 % при отсутствии агрессивных примесей в окружающей среде.

Подготовленные к упаковке программные изделия укладывают в тару, представляющую собой коробки из картона. Для заполнения свободного пространства в упаковочную тару укладываются прокладки из гофрированного картона или пенопласта. На верхний слой прокладочного материала укладывается товаросопроводительная документация - упаковочный лист и ведомость упаковки.

# 4.7 Требования к транспортированию и хранению

* + транспортировка разрабатываемого программного продукта должна осуществляться студентом в назначенный срок по указанию преподавателя, принимающего результаты курсовой работы;
  + ответственным за хранение программного продукта является студент;
  + результаты выполненной работы хранятся на кафедре «Информатика» в течение года.

# Требования к программной документации

Предварительный перечень программной документации:

* + пояснительная записка к курсовой работе, оформленная в соответствии с ГОСТ 7.32-2001;

# Технико-экономические показатели

В данной работе не рассчитываются.

# Стадии и этапы разработки

Стадии и этапы разработки определены в план-графике выполнения курсовой работы.

# Порядок контроля и приемки

* контроль и приемка программного продукта осуществляется в течение семестра поэтапно в соответствии с план-графиком;
* КР в окончательном виде предоставляется на проверку преподавателям за неделю до ее защиты;
* курсовая работа подлежит защите, в ходе которой студент представляет свою разработку и выполняет практические задания с целью демонстрации практических навыков на компьютере. На защите КР преподавателями не осуществляется проверка программной разработки и пояснительной записки;
* в случае регулярного и качественного выполнения план-графика сдачи этапов КР, студент может быть освобожден от защиты КР;
* в случае отсутствия в пояснительной записке глав или подпунктов, которые были оговорены в техническом задании, студенту на защите КР выставляется оценка «неудовлетворительно»;
* в случае невыполнения план-графика в течение семестра, студенту будут выданы дополнительные задания;
* в случае обнаружения заимствования чужих курсовых работ, студенту выставляется оценка «неудовлетворительно»;
* оценку программного продукта производят преподаватели, осуществляющие проверку курсовых работ.

# ГЛАВА 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

# 1.1 Основные понятия и определения

Программа (program, routine) — упорядоченная последовательность команд (инструкций) компьютера или операторов для решения задачи. При эксплуатации программы выполняется некоторая работа.

Программное обеспечение (software)— совокупность программ обработки данных и необходимых для их эксплуатации документов.

Задача (problem, task) — проблема, подлежащая решению.

Приложение (application)— программная реализация задачи, решенная на компьютере.

Термин «задача» в программировании означает единицу работы вычислительной системы, требующую выделения вычислительных ресурсов (процессорного времени, памяти).

Процесс создания программ можно представить как последовательность следующих действий:

* постановка задачи;
* алгоритмизация решения задачи;
* программирование.

Постановка задачи (problem definition)— это точная формулировка решения задачи на компьютере с описанием входной и выходной информации.

Алгоритм— система точно сформулированных правил, определяющая процесс преобразования допустимых исходных данных (входной информации) в желаемый результат (выходную информацию) за конечное число шагов.

Программирование (programming)— теоретическая и практическая деятельность, связанная с созданием программ.

По отношению к программному обеспечению (ПО) компьютерные пользователи делятся на следующие группы:

* системные программисты. Занимаются разработкой, эксплуатацией и сопровождением системного программного обеспечения;
* прикладные программисты. Осуществляют разработку и отладку программ для решения различных прикладных задач;
* конечные пользователи. Имеют элементарные навыки работы с компьютером и используемыми ими прикладными программами;
* администраторы сети. Отвечают за работу вычислительных сетей;
* администраторы баз данных. Обеспечивают организационную поддержку базы данных.

Сопровождение программы— поддержка работоспособности программы, переход ее на новые версии, внесения изменений, исправление ошибок и т. д.

Основные характеристики программ:

* алгоритмическая сложность;
* состав функций обработки информации;
* объем файлов, используемых программой;
* требования к операционной системе (ОС) и техническим средствам обработки, в том числе объем дисковой памяти, размер оперативной памяти для запуска программы, тип процессора, версия ОС, наличие вычислительной сети и т. д.

Программный продукт— комплекс взаимосвязанных программ для решения определенной проблемы (задачи) массового спроса, подготовленный к реализации как любой вид промышленной продукции.

Программные продукты (изделия) используются для удовлетворения потребностей пользователей, широкого распространения и продажи. Программная продукция — это программа (программный комплекс, программная система), которую стремятся продать на широком рынке тысячам, миллионам пользователей. Товарное программное обеспечение разрабатывается со специальной целью удовлетворить требованиям максимально большого числа пользователей. Это оказывает влияние на весь жизненный цикл программ, но в наибольшей степени на начальный этап разработки, на определение требований к системе.

Программный продукт должен быть соответствующим образом подготовлен к эксплуатации (отлажен), иметь необходимую техническую документацию, предоставлять сервис и гарантию надежной работы, иметь товарный знак изготовителя, а также код государственной регистрации. Только при таких условиях программный комплекс может быть назван программным продуктом.

Утилитарные программы(«программы для себя») предназначены для удовлетворения нужд их разработчиков. Чаще всего утилитарные программы имеют сервисное назначение в технологии обработки данных либо являются программами решения функциональных задач, не предназначенных для широкого распространения.

В настоящее время существуют варианты легального распространения программных продуктов, которые появились с использованием глобальных телекоммуникаций:

* freeware — бесплатные программы, свободно распространяемые, под- держиваются самим пользователем, который правомочен вносить в них необходимые изменения;
* shareware — некоммерческие (условно бесплатные) программы, которые могут использоваться, как правило, бесплатно.

Ряд производителей использует OEM-программы (Original Equipment Manufacturer), то есть встроенные программы, устанавливаемые на компьютеры или поставляемые вместе с компьютерами.

Технология программирования — совокупность методов и средств, применяемых в процессе разработки программного обеспечения. Или это система методов, способов и приемов разработки и отладки программ.

На основании анализа различных формулировок (политехнический словарь, энциклопедический словарь и др.) под технологией программирования в широком смысле следует понимать технологию разработки программного средства как совокупность абсолютно всех технологических процессов его создания — от момента зарождения идеи о данном программном средстве (ПС) до составления необходимой программной документации.

Как любая другая технология, технология программирования представляет собой набор технологических инструкций, включающих:

* указание последовательности выполнения технологических операций;
* перечисление условий, при которых выполняется та или иная операция;
* описания самих операций, где для каждой операции определены исходные данные, результаты, а также инструкции, нормативы, стандарты, критерии, методы оценки и т.п. (рис 1).

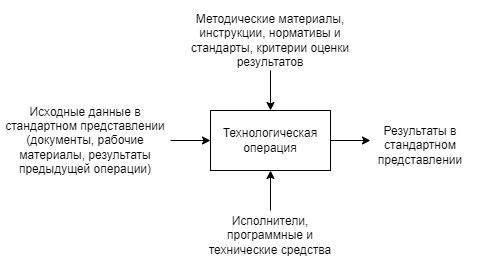


Рисунок 1 – Структура описания технологической операции

Кроме набора операций и их последовательности, технология также определяет способ описания проектируемой системы, точнее модели, используемой на конкретном этапе разработки.

Различают технологии, используемые на конкретных этапах разработки или для решения отдельных задач этих этапов, и технологии, охватывающие несколько этапов или весь процесс разработки. В основе первых, как правило, лежит ограниченно применимый метод, позволяющий решить конкретную задачу. В основе вторых обычно лежит базовый метод или подход, определяющий совокупность методов, используемых на разных этапах разработки, или методологию.

Чтобы разобраться в существующих технологиях программирования и определить основные тенденции их развития, целесообразно рассматривать эти технологии в историческом контексте, выделяя основные этапы развития программирования как науки.

# 1.2 Типы программного обеспечения

Меняя программы для компьютера, его можно превратить в рабочее место бухгалтера, конструктора, статистика или агронома, редактировать на нем документы или играть в игру. При своем выполнении программы могут использовать различные устройства компьютера для ввода-вывода данных. Таким образом, для эффективного использования персональной электронно-вычислительной машины (ПЭВМ) необходимо знать назначение и свойства необходимых при работе с ним программ.

Все программное обеспечение можно условно разделить на три типа:

* + прикладное программное обеспечение;
  + системное ПО;
  + инструментальное ПО.

Прикладное ПОслужит программным инструментарием решения функциональных задач и является самым многочисленным классом ПО. В данный класс входят программные продукты, выполняющие обработку информации различных предметных областей. Таким образом, прикладное ПО — комплекс взаимосвязанных программ для решения задач определенного класса предметной области.

**Пакеты прикладных программ**

Классификация пакетов прикладных программ (ППП) приведена на рисунке 2.



Рисунок 2 – Классификация пакетов прикладных программ

Проблемно-ориентированные ППП. Для некоторых предметных областей возможна типизация функций управления, структуры данных и алгоритмов обработки. Разработано значительное количество ППП одинакового функционального назначения:

* + ППП автоматизированного бухгалтерского учета;
  + ППП финансовой деятельности;
  + ППП управления персоналом;
  + ППП управления производством;
  + банковские информационные системы и т. п. Основные тенденции развития:
  + создание программных комплексов в виде автоматизированных рабочих мест (АРМ) управленческого персонала;
  + создание интегрированных систем управления предметной областью на базе вычислительных сетей, объединяющих АРМы;
  + организация данных больших информационных систем в виде распределенной БД на сети ЭВМ;
  + наличие простых языковых средств конечного пользователя и др.

ППП автоматизированного проектирования предназначены для поддержки работы конструкторов и технологов, связанных с разработкой чертежей, схем, графическим моделированием и конструированием. Отличительными особенностями этого класса ППП являются высокие требования к аппаратному обеспечению, наличие библиотек встроенных функций, объектов, интерфейсов с графическими системами и БД (AutoCAD).

К ППП общего назначения относят:

1. Системы управления базами данных (СУБД), обеспечивающие организацию и хранение локальных БД на автономно работающих компьютерах либо централизованное хранение БД на файл-сервере и сетевой доступ к ним. В современных СУБД содержатся элементы CASE-технологии процесса проектирования, в частности:
   * визуализирована схема БД;
   * осуществлена автоматическая поддержка целостности БД при различных видах обработки (включение, удаление, модификация);
   * предоставляются «мастера», обеспечивающие поддержку процесса проектирования;
   * созданы шаблоны (прототипы) структур БД, отчетов, форм и т. д.
2. Серверы БД — это ПО, предназначенное для создания и использования при работе в сети интегрированных БД в архитектуре «клиент — сервер». Многопользовательские СУБД в сетевом варианте обработки информации хранят данные на файл-сервере, специально выделенном компьютере, но сама обработка ведется на рабочих станциях. Серверы БД в отличие от этого большую часть обработки данных (хранение, поиск, извлечение и передачу данных клиенту) выполняют самостоятельно, одновременно обеспечивая данными большое число пользователей сети. Общим для различных видов серверов БД является использование реляционного языка SQL (Structured Query Language) для реализации запросов к данным. Большинство серверов БД поддерживают несколько платформ, широкий спектр протоколов передачи данных. Проблемы — обеспечение целостности данных, тиражирование данных по узлам сети и синхронное обновление.
3. Генераторы отчетов (серверы отчетов), обеспечивающие реализацию запросов и формирование отчетов в печатном или экранном виде в условиях сети с архитектурой «клиент — сервер». Сервер отчетов подключается к серверу БД, использующему драйверы сервиса БД (Crystal Reports, Profit for Windows).
4. Текстовые процессоры, предназначенные для работы с текстовыми документами. Развитием данного направления являются издательские системы (например, Microsoft Word).
5. Табличные процессоры, являющиеся удобной средой для вычислений конечным пользователем, содержат средства деловой графики, средства специализированной обработки (Microsoft Excel).
6. Средства презентационной графики — специализированные про- граммы, предназначенные для создания изображений и их показа на экране, подготовки слайд-фильмов, мультфильмов и их проектирования (Microsoft PowerPoint).
7. Интегрированные пакеты — набор нескольких программных продуктов, функционально дополняющих друг друга, поддерживающих единые информационные технологии, реализованных на единой операционной и вычислительной платформе (Microsoft Office). Компоненты интегрированных пакетов могут работать изолированно друг от друга, имеют общий интерфейс, благодаря этому их проще изучать.

Методо-ориентированные ППП. Данный класс охватывает программные продукты, обеспечивающие независимо от предметной области и функции информационных систем математические, статистические и другие методы решения задач. Наиболее распространены методы математического программирования, решения дифференциальных уравнений, имитационного моделирования, исследования операций (Storm, SYSTAT, SAS и др.).

Офисные ППП. Данный класс охватывает программы, обеспечивающие ориентационное управление деятельностью офиса:

* + органайзеры (планировщики) — ПО для планирования рабочего времени, составления протоколов встреч, расписаний, ведение записей и телефонной книжки. В состав входят: калькулятор, записная книжка, часы, календарь и т. п.
  + программы-переводчики, средства проверки орфографии, распознавания текста (Tiger — система распознавания русского языка, Stylus Lingvo Office, содержащий Fine Reader, Stylus for Windows — переводчик на указанный язык, корректор орфографии Lingvo Corrector и резидентный словарь Lingvo);
  + коммуникационные пакеты, предназначенные для организации взаимодействия пользователей с удаленными абонентами или информационными ресурсами сети;
  + браузеры, средства создания WWW-страниц;
  + средства электронной почты (Pegasys Mail).

Настольные издательские системы. Данный класс ПО включает программы (PageMaker, CorelDraw, Photoshop for Windows и т. д.), обеспечивающие информационную технологию компьютерной издательской деятельности:

* + форматирование и редактирование текстов;
  + автоматическую разбивку текста на страницы;
  + компьютерную верстку печатной страницы;
  + монтирование графики;
  + подготовку иллюстраций и т. п.

Программные средства мультимедиа. Основное значение данных программных средств — создание и использование аудио- и видеоинформации для расширения информационного пространства пользователя (различные БД компьютерных произведений искусства, библиотеки звуковых записей и т. д.).

Интеллектуальные системы. К системам искусственного интеллекта относятся:

* + программы-оболочки для создания экспертных систем путем наполнения баз знаний и правил логического вывода;
  + готовые экспертные системы для принятия решений в рамках определенных предметных областей;
  + системы анализа и распознавания речи, текста и т. п.

Примеры систем искусственного интеллекта: FIDE, MYSIN, Guru и др.

**Системное ПО**

Системное ПО (System Software) — совокупность программ и программных комплексов для обеспечения работы компьютера и вычислительных сетей.

В составе системного ПО можно выделить две составляющие:

1. базовое ПО;
2. сервисное ПО.

Базовое ПО — минимальный набор программных средств, обеспечивающих работу компьютера.

Сервисное ПО — программы и программные комплексы, которые расширяют возможности базового ПО и организуют более удобную среду работы пользователя.

1. Базовое ПО.

В базовое ПО входят:

* + операционная система;
  + операционные оболочки (текстовые, графические);
  + сетевая операционная система.

Компьютер выполняет действия в соответствии с предписаниями программы, созданной на одном из языков программирования. При работе пользователя на компьютере часто возникает необходимость выполнять операции с прикладной программой в целом, организовать работу внешних устройств, проверить работу различных блоков, скопировать информацию и т. д. Эти операции используются для работы с любой программой. Поэтому целесообразно из всего многообразия операций, выполняемых компьютером, выделить типовые и реализовать их с помощью специализированных программ, которые следует принять в качестве стандартных средств, поставляемых вместе с аппаратной частью. Программы, организующие работу устройств и не связанные со спецификой решаемой задачи, вошли в состав комплекса программ, названного операционной системой.

Операционная система (ОС) — совокупность программных средств, обеспечивающих управление аппаратной частью компьютера и прикладными программами, а также их взаимодействием между собой и пользователем.

Операционная система образует автономную среду, не связанную ни с одним из языков программирования. Любая прикладная программа связана с ОС и может эксплуатироваться только на компьютерах, где имеется аналогичная системная среда. Программа, созданная в среде одной ОС, не функционирует в среде другой ОС, если в ней не обеспечена возможность конвертации (преобразования) программ.

Для работы с ОС необходимо овладеть языком этой среды — совокупностью команд, структура которых определяется синтаксисом этого языка.

ОС выполняет следующие функции:

* + управление работой каждого блока ПО и их взаимодействием;
  + управление выполнением программ;
  + организацию хранения информации во внешней памяти;
  + взаимодействие пользователя с компьютером (поддержку интерфейса пользователя).

Обычно ОС хранится на жестком диске, а при его отсутствии выделяется специальный гибкий диск, который называется системным диском. При включении компьютера ОС автоматически загружается с диска в оперативную память и занимает в ней определенное место. ОС создается не для отдельной модели компьютера, а для серии компьютеров, в структуру которых заложена и развивается во всех последующих моделях определенная концепция.

Операционные системы для персонального компьютера (ПК) делятся на:

* + одно- и многозадачные (в зависимости от числа параллельного выполнения прикладных процессов);
  + одно- и многопользовательские (в зависимости от числа пользователей, одновременно работающих с ОС);
  + непереносимые и переносимые на другие типы компьютеров;
  + несетевые и сетевые, обеспечивающие работу ЭВМ.
  + функциональным возможностям вспомогательных программ (утилит) в составе ОС;
  + наличию компрессии магнитных дисков;
  + возможности архивирования файлов;

поддержке многозадачного режима работы;

* + наличию качественной документации;
  + условию и сложности процесса инсталляции.

Сетевая ОС — это комплекс программ, обеспечивающий обработку, передачу и хранение данных в сети. Сетевая ОС предоставляет пользователям различные виды сетевых служб (управление файлами, электронная почта, процессы управления сетью и т. д.), поддерживает работу в абонентских системах. Сетевые ОС используют архитектуру «клиент — сервер» или одноранговую архитектуру. Вначале эти ОС поддерживали только локальные вычислительные сети (ЛВС), сейчас они распространяются и на объединения ЛВС.

Операционные оболочки — специальные программы, предназначенные для облегчения общения пользователя с командами ОС. Операционные оболочки имеют текстовый и графический варианты интерфейса конечного пользователя. Наиболее популярны такие текстовые оболочки MS DOS, как Norton Commander, DOS Navigator, Volkov Commander и др. Эти программы существенно упрощают задание управляющей информации для выполнения команд ОС, уменьшают сложность работы конечного пользователя.

1. Сервисное ПО.

Расширением базового ПО компьютера является набор сервисных, дополнительно устанавливаемых программ, которые можно классифицировать по функциональному признаку следующим образом:

* + программы диагностики работоспособности компьютера;
  + антивирусные программы, обеспечивающие защиту компьютера, обнаружение и восстановление зараженных файлов;
  + программы обслуживания дисков, обеспечивающие проверку качества поверхности магнитного диска, контроль сохранности файловой системы на логическом и физическом уровнях, сжатие дисков, резервное копирование данных на внешние носители и др.;
  + программы архивирования данных, которые обеспечивают процесс сжатия информации с целью уменьшения объема памяти для ее хранения;
  + программы обслуживания сети.

Эти программы часто называют утилитами.

Утилиты — программы, служащие для выполнения вспомогательных операций обработки данных или обслуживания компьютеров.

**Инструментальные ПО**

Инструментарий технологии программирования обеспечивает процесс разработки программ и включает специализированное ПО, которое является инструментальным средством разработки. ПО данного класса поддерживает все технологические этапы процесса проектирования, программирования, отладки и тестирования создаваемых программ. Пользователями данного ПО являются системные и прикладные программисты.

Инструментарий технологии программирования — это программные продукты, предназначенные для поддержки технологии программирования (рис. 3).



Рисунок 3 – Инструментарий технологии программирования

Средства для создания приложений — совокупность языков и систем программирования, инструментальные среды пользователя, а также различные программные компоненты для отладки и поддержки создаваемых программ.

Язык программирования — это формализованный язык для описания алгоритма решения задач на компьютере. Языки программирования можно условно разделить на следующие классы:

* + машинные языки — это языки, воспринимаемые аппаратной частью компьютера (машинные коды);
  + машинно-ориентированные языки, отражающие структуру конкретного типа компьютера (ассемблер);
  + процедурно-ориентированные языки — это языки, в которых имеется возможность описания программы как совокупности процедур, или подпрограмм (Си#, Паскаль и др.);
  + проблемно-ориентированные языки, предназначенные для решения задач определенного класса (ЛИСП, ПРОЛОГ и др.).

Другой классификацией языков является их деление на языки, ориентированные на реализацию основ структурного программирования, основанного на модульной структуре программного продукта и типовых управляющих структурах алгоритмов обработки данных различных программных модулей, и объектно-ориентированные языки, поддерживающие понятия объектов, их свойств и методов обработки.

Системы программирования включают:

* + компилятор (транслятор);
  + интегрированную среду разработки программ (не всегда);
  + отладчик;
  + средства оптимизации кода программ;
  + набор библиотек;
  + редактор связей;
  + сервисные средства (утилиты) (для работы с библиотеками, текстовыми и двоичными файлами);
  + справочные системы;

систему поддержки и управления продуктами программного комплекса.

Компилятор транслирует всю программу без ее выполнения. Интерпретаторы выполняют пооперационную обработку и выполнение программы.

Отладчики (debugger) — специальные программы, предназначенные для трассировки и анализа выполнения других программ. Трассировка — это обеспечение выполнения в пооператорном варианте.

Инструментальная среда пользователя — это специальные средства, встроенные в пакеты прикладных программ, такие как:

* + библиотека функций, процедур, объектов и методов обработки;
  + макрокоманды;
  + клавишные макросы;
  + языковые макросы;
  + конструкторы экранных форм и объектов;
  + генераторы приложений;
  + языки запросов высокого уровня;
  + конструкторы меню и др.

Интегрированные среды разработки программ объединяют набор средств для их комплексного применения на технологических этапах создания программы.

Средства для создания информационных систем (ИС) и технологий поддерживают полный цикл проектирования сложной информационной системы или технологии от исследования объекта автоматизации до оформления проектной и прочей документации на информационную систему или технологию. Они позволяют вести коллективную работу над проектом за счет возможности работы в локальной сети, экспорта-импорта любых фрагментов проекта, организации управления проектом.

Одним из современных средств разработки ИС является CASE- технология (CASE — Computer-Aided System Engineering) — программный комплекс, автоматизирующий весь технологический процесс анализа, проектирования, разработки и сопровождения сложных программных систем.

Средства CASE-технологий делятся на:

* + встроенные в систему реализации — все решения по проектированию и реализации привязки к выбранной СУБД;
  + независимые от системы реализации — все решения по проектированию ориентированы на унификацию (определение) начальных этапов жизненного цикла программы и средств их документирования, обеспечивают большую гибкость в выборе средств реализации.

Основное достоинство CASE-технологии — это поддержка коллективной работы над проектом за счет возможности работы в локальной сети разработчиков, экспорта (импорта) любых фрагментов проекта, организованного управления проектами.

В некоторых CASE-системах поддерживается генерация кода программ — создание каркаса программ и создание полного продукта.

# 1.3 Жизненный цикл и этапы разработки программного обеспечения

Программные системы (ПС) и комплексы программ создаются, эксплуатируются и развиваются во времени.

Жизненный цикл ПС включает в себя все этапы развития от возникновения потребности в программе определенного целевого назначения до полного прекращения использования этого ПС вследствие его морального старения или потери необходимости решения задачи.

По длительности жизненного цикла ПС можно разделить на два класса: с малым и большим временем жизни.

Программы с малым временем жизни создаются в основном для получения конкретных результатов вычислений. Эти программы относительно невелики, разрабатываются одним специалистом или небольшой группой и не предназначены для тиражирования. Требования к таким программам по функциональным, конструктивным и качественным показателям контролируются в основном разработчиками и только иногда представителями заказчика. Основные затраты в жизненном цикле (ЖЦ) этих программ приходятся на этап системного анализа и проектирования, который продолжается от месяца до 1–2 лет.

Программы с большим временем жизни создаются для регулярной обработки информации. Размеры ПС могут изменяться в широких пределах. Они обладают свойством познаваемости и возможности модификации в процессе длительного сопровождения и использования различными специалистами. Их тиражируют, они сопровождаются документацией как промышленные изделия. Проектированием и эксплуатацией таких ПС занимаются большие коллективы специалистов, для чего необходимы формализация требуемых технических характеристик, а также формализованные испытания и определения достигнутых показателей качества ПС. Большая часть жизненного цикла приходится на эксплуатацию и сопровождение. Затраты в процессе эксплуатации и сопровождения могут значительно превышать затраты на системный анализ и проектирование.

Для получения представления о возможной структуре жизненного цикла ПО необходимо обратиться к соответствующим стандартам, описывающим технологические процессы и международным организациям:

IEEE — Institute of Electrical and Electronic Engineers (Институт инженеров по электротехнике и электронике);

ISO — International Standards Organization (Международная организация по стандартизации);

EIA — Electronic Industry Association (Ассоциация электронной промыш- ленности);

IEC — International Electrotechnical Commission (Международная комиссия по электротехнике).

А также к некоторым национальным и региональным институтам и организациям (американским и европейским, поскольку именно они оказывают наибольшее влияние на развитие технологий разработки ПО во всем мире):

ANSI — American National Standards Institute (Американский национальный институт стандартов);

SEI — Software Engineering Institute (Институт программной инженерии); ECMA — European Computer Manufacturers Association (Европейская ассоциация производителей компьютерного оборудования).

Разработан набор стандартов, регламентирующих различные аспекты жизненного цикла и вовлеченных в него процессов.

Состав процессов жизненного цикла регламентируется стандартом ГОСТ Р ИСО/МЭК 12207-2010 (принят на основании международного стандарта ISO/IEC 12207:2008 «Information Technologe — Software Life Cycle Processes», «Информационные технологии — процессы жизненного цикла программного обеспечения»).

Условность выделения этапов ЖЦ связана с тем, что на любом этапе возможно принятие решений, которые потребуют пересмотра решений, принятых ранее. Основные этапы ЖЦ могут быть следующими.

Постановка задачи. В процессе постановки задачи четко формулируют назначение программного обеспечения и определяют основные требования к нему. Каждое требование представляет собой описание необходимого или желаемого свойства программного обеспечения.

Различают функциональные требования, определяющие функции, которые должно выполнять разрабатываемое программное обеспечение, и эксплуатационные требования, определяющие особенности его функционирования.

Требования к программному обеспечению, имеющему прототипы, обычно определяют по аналогии, учитывая структуру и характеристики уже существующего программного обеспечения.

Для формулирования требований к программному обеспечению, не имеющему аналогов, иногда необходимо провести специальные исследования, называемые предпроектными. В процессе таких исследований определяют разрешимость задачи, возможно, разрабатывают методы ее решения (если они новые) и устанавливают наиболее существенные характеристики разрабатываемого программного обеспечения. Для выполнения предпроектных исследований, как правило, заключают договор на выполнение научно-исследовательских работ. В любом случае этап постановки задачи заканчивается разработкой технического задания, фиксирующего принципиальные требования, и принятием основных проектных решений.

Анализ требований и определение спецификаций. Спецификациями называют точное формализованное описание функций и ограничений разрабатываемого программного обеспечения. Соответственно различают функциональные и эксплуатационные спецификации, а также спецификацию качества будущего ПО.

Совокупность спецификаций представляет собой общую логическую модель проектируемого программного обеспечения. Для получения спецификаций выполняют анализ требований технического задания, формулируют содержательную постановку задачи, выбирают математический аппарат формализации, строят модель предметной области, определяют подзадачи и выбирают или разрабатывают методы их решения. Часть спецификаций может быть определена в процессе предпроектных исследований и, соответственно, зафиксирована в техническом задании.

На этом этапе также целесообразно сформировать тесты для поиска ошибок в проектируемом программном обеспечении, обязательно указав ожидаемые результаты.

Проектирование. Основной задачей этого этапа является определение подробных спецификаций разрабатываемого программного обеспечения. Процесс проектирования сложного программного обеспечения обычно включает:

* + проектирование общей структуры — определение основных компонентов и их взаимосвязей;
  + декомпозицию компонентов и построение структурных иерархий в соответствии с рекомендациями блочно-иерархического подхода;
  + проектирование компонентов.

Результатом проектирования является детальная модель разрабатываемого программного обеспечения вместе со спецификациями его компонентов всех уровней. Тип модели зависит от выбранного подхода (структурный, объектный или компонентный) и конкретной технологии проектирования. Однако в любом случае процесс проектирования охватывает как проектирование программ (подпрограмм) и определение взаимосвязей между ними, так и проектирование данных, с которыми взаимодействуют эти программы или подпрограммы.

Принято различать также два аспекта проектирования:

* + логическое проектирование, которое включает те проектные операции, которые непосредственно не зависят от имеющихся технических и программных средств, составляющих среду функционирования будущего программного продукта;
  + физическое проектирование — привязка к конкретным техническим и программным средствам среды функционирования, то есть учет ограничений, определенных в спецификациях.

Реализация. Реализация представляет собой процесс поэтапного написания кодов программы на выбранном языке программирования (кодирование), их тестирование и отладку.

Сопровождение. Сопровождение — это процесс создания и внедрения новых версий программного продукта. Причинами выпуска новых версий мо- гут служить:

* + необходимость исправления ошибок, выявленных в процессе эксплуатации предыдущих версий;
  + необходимость совершенствования предыдущих версий, например улучшения интерфейса или повышения его производительности, расширения состава выполняемых функций;
  + изменение среды функционирования, например появление новых технических средств и/или программных продуктов, с которыми взаимодействует сопровождаемое программное обеспечение.

На этом этапе в программный продукт вносят необходимые изменения, которые, так же как в остальных случаях, могут потребовать пересмотра проектных решений, принятых на любом предыдущем этапе. Роль этого этапа существенно возросла, так как продукты создаются итерационно: сначала выпускается сравнительно простая версия, затем следующая с большими возможностями, затем следующая и т. д. Именно это и послужило причиной выделения этапа сопровождения в отдельный процесс жизненного цикла в соответствии со стандартом ГОСТ Р ИСО/МЭК 12207-2010.

Структура ЖЦ ПО по стандарту ГОСТ Р ИСО/МЭК 12207-2010 базируется на трех группах процессов:

* + основные процессы ЖЦ ПО (приобретение, поставка, разработка, эксплуатация, сопровождение);
  + вспомогательные процессы, обеспечивающие выполнение основных процессов (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит, решение проблем);
  + организационные процессы (управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение самого ЖЦ, обучение).

Разработка включает в себя все работы по созданию ПО и его компонент в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации, подготовку материалов, необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала и т. д.

Эксплуатация включает в себя работы по внедрению компонентов ПО в эксплуатацию, в том числе конфигурирование базы данных и рабочих мест пользователей, обеспечение эксплуатационной документацией, проведение обучения персонала и т. д. А также непосредственно эксплуатацию, в том числе локализацию проблем и устранение причин их возникновения, модификацию ПО в рамках установленного регламента, подготовку предложений по совершенствованию, развитию и модернизации системы.

Управление проектом связано с вопросами планирования и организации работ, создания коллективов разработчиков и контроля со сроками и качеством выполняемых работ. Техническое и организационное обеспечение проекта включает выбор методов и инструментальных средств для реализации проекта, определение методов описания промежуточных состояний разработки, разработку методов и средств испытаний ПО, обучение персонала и т. п. Обеспечение качества проекта связано с проблемами верификации, проверки и тестирования ПО. Верификация — это процесс определения того, отвечает ли текущее состояние разработки, достигнутое на данном этапе, требованиям этого этапа.

Проверка позволяет оценить соответствие параметров разработки исходным требованиям. Проверка частично совпадает с тестированием, которое связано с идентификацией различий между действительными и ожидаемыми результатами и оценкой соответствия характеристик ПО исходным требованиям. В процессе реализации проекта важное место занимают вопросы идентификации, описания и контроля конфигурации отдельных компонентов и всей системы в целом.

Управление конфигурацией является одним из вспомогательных процессов, поддерживающих основные процессы жизненного цикла ПО, прежде всего процессы разработки и сопровождения ПО. При создании проектов сложных ИС, состоящих из многих компонентов, каждый из которых может иметь разновидности или версии, возникает проблема учета их связей и функций, создания унифицированной структуры и обеспечения развития всей системы.

Управление конфигурацией позволяет организовывать, систематически учитывать и контролировать внесение изменений в ПО на всех стадиях ЖЦ. Общие принципы и рекомендации конфигурационного учета, планирования и управления конфигурациями ПО отражены в стандарте ГОСТ Р ИСО/МЭК 12207-2010.

Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными на предыдущем этапе, и результатами. Результатами анализа, в частности, являются функциональные модели, информационные модели и соответствующие им диаграммы. ЖЦ ПО носит итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних этапах.

Процесс жизненного цикла определяется как совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные. На рисунке 4 представлены процессы жизненного цикла по указанному стандарту.

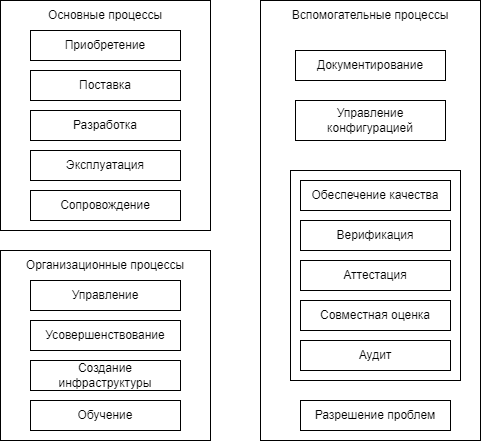


Рисунок 4 – Структура процессов жизненного цикла программного обеспечения

По стандарту процесс разработки включает следующие действия:

* + подготовительная работа — выбор модели жизненного цикла (см. далее), стандартов, методов и средств разработки, а также составление плана работ;
  + анализ требований к системе — определение ее функциональных возможностей, пользовательских требований, требований к надежности и безопасности, требований к внешним интерфейсам и т. д.;
  + проектирование архитектуры системы — определение состава необходимого оборудования, программного обеспечения и операций, выполняемых обслуживающим персоналом;
  + анализ требований к программному обеспечению — определение функциональных возможностей, включая характеристики производительности, среды функционирования компонентов, внешних интерфейсов, спецификаций надежности и безопасности, эргономических требований, требований к используемым данным, установке, приемке, пользовательской документации, эксплуатации и сопровождению;
  + проектирование архитектуры программного обеспечения — определение структуры программного обеспечения, документирование интерфейсов его компонентов, разработка предварительной версии пользовательской документации, а также требований к тестам и плану интеграции;
  + детальное проектирование программного обеспечения — подробное описание компонентов программного обеспечения и интерфейсов между ними, обновление пользовательской документации, разработка и документирование требований к тестам и плану тестирования компонентов программного обеспечения, обновление плана интеграции компонентов;
  + кодирование и тестирование программного обеспечения — разработка и документирование каждого компонента, а также совокупности тестовых процедур и данных для их тестирования, тестирование компонентов, обновление пользовательской документации, обновление плана интеграции программного обеспечения;
  + интеграция программного обеспечения — сборка программных компонентов в соответствии с планом интеграции и тестирование программного обеспечения на соответствие квалификационным требованиям, представляющим собой набор критериев или условий, которые необходимо выполнить, чтобы квалифицировать программный продукт как соответствующий своим спецификациям и готовый к использованию в заданных условиях эксплуатации;
  + квалификационное тестирование программного обеспечения — тестирование программного обеспечения в присутствии заказчика для демонстрации его соответствия требованиям и готовности к эксплуатации, при этом проверяются также готовность и полнота технической и пользовательской документации;
  + интеграция системы — сборка всех компонентов системы, включая программное обеспечение и оборудование;
  + квалификационное тестирование системы — тестирование системы на соответствие требованиям к ней и проверка оформления и полноты документации;
  + установка программного обеспечения — установка программного обеспечения на оборудовании заказчика и проверка его работоспособности;
  + приемка программного обеспечения — оценка результатов квалификационного тестирования программного обеспечения и системы в целом и документирование результатов оценки совместно с заказчиком, окончательная передача программного обеспечения заказчику.

# 1.4 Модели жизненного цикла ПО

**Каскадная модель**

Каскадная модель имела широкое распространение в 1970–1985 гг. Основной характеристикой этой модели является разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем (рис. 5). Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

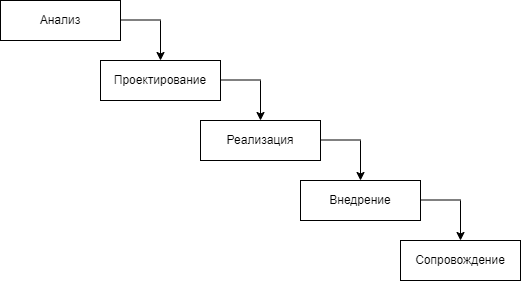


Рисунок 5 – Каскадная схема разработки ПО

Положительные стороны применения каскадного подхода заключаются в следующем:

* + на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
  + выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Каскадный подход хорошо зарекомендовал себя при построении ПС, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования и предоставить разработчикам свободу реализовать их как можно лучше с технической точки зрения. В эту категорию попадают сложные расчетные системы, системы реального времени и другие подобные задачи. Эту модель широко использовало министерство обороны США в 1970– 1980-х гг. для разработки ПО. Однако в процессе использования этого подхода обнаружился ряд его недостатков, вызванных прежде всего тем, что реальный процесс создания ПО никогда полностью не укладывался в такую жесткую схему. В процессе создания ПО постоянно возникала потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ПО принимал вид, показанный на рисунке 6.

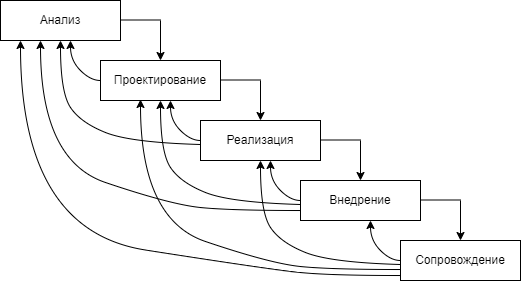


Рисунок 6 – Реальный процесс разработки ПО по каскадной схеме

Основным недостатком каскадного подхода является существенное запаздывание с получением результатов. Согласование результатов с пользователями производится только в точках, планируемых после завершения каждого этапа работ, требования к ПС «заморожены» в виде технического задания на все время ее создания. Таким образом, пользователи могут внести свои замечания только после того, как работа над системой будет полностью завершена. В случае неточного изложения требований или их изменения в течение длительного периода создания ПО пользователи получают систему, не удовлетворяющую их потребностям.

Итеративные или инкрементальные модели(известно несколько таких моделей) предполагают разбиение создаваемой системы на набор кусков, которые разрабатываются с помощью нескольких последовательных проходов всех работ или их части.

На первой итерации разрабатывается фрагмент системы, не зависящий от других (рис. 7). При этом большая часть или даже полный цикл работ осуществляются на нем, затем оцениваются результаты, и на следующей итерации либо первый фрагмент переделывается, либо разрабатывается следующий, который может зависеть от первого, либо как-то совмещается доработка первого фрагмента с добавлением новых функций. В результате на каждой итерации можно анализировать промежуточные результаты работ и реакцию на них всех заинтересованных лиц, включая пользователей, и вносить корректирующие изменения на следующих итерациях. Каждая итерация может содержать полный набор видов деятельности от анализа требований до ввода в эксплуатацию очередной части ПО.

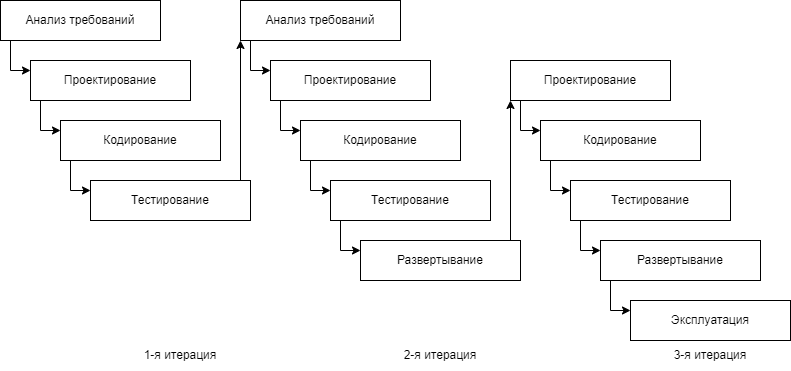


Рисунок 7 – Возможный ход работ по итеративной модели

Каскадная модель с возможностью возвращения на предшествующий шаг при необходимости пересмотреть его результаты становится итеративной.

Итеративный процесс предполагает, что разные виды деятельности не привязаны намертво к определенным этапам разработки, а выполняются по мере необходимости, иногда повторяются до тех пор, пока не будет получен нужный результат.

Вместе с гибкостью и возможностью быстро реагировать на изменения, итеративные модели вносят дополнительные сложности в управление проектом и отслеживание его хода. При использовании итеративного подхода значительно сложнее становится адекватно оценить текущее состояние проекта и спланировать долгосрочное развитие событий, а также предсказать сроки и ресурсы, необходимые для обеспечения определенного качества результата.

**Спиральная модель**

Спиральная модель получила широкое распространение в конце 1980-х — начале 1990-х гг. Она была предложена Б. Боэмом (Boehm) для преодоления перечисленных проблем, которые были у каскадной модели ЖЦ. Спиральная модель (рис. 8) делает упор на начальные этапы ЖЦ: анализ и проектирование. На этих этапах реализуемость технических решений проверяется путем создания прототипов. Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта, и в результате выбирается обоснованный вариант, который доводится до реализации.

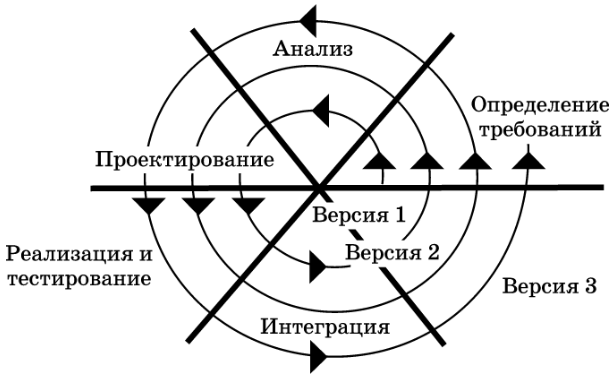


Рисунок 8 – Спиральная модель ЖЦ

Разработка итерациями отражает объективно существующий спиральный цикл создания системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем. При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная же задача — как можно бы- стрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

Основная проблема спирального цикла — определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

**Изменение жизненного цикла программного обеспечения при использовании CASE-технологий**

CASE-технологии представляют собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных программных систем, основанных как на структурном, так и на объектном подходах, которые поддерживаются комплексом взаимосвязанных средств автоматизации.

В основе любой CASE-технологии лежит парадигма методология/метод/нотация/средство.

Методология строится на базе некоторого подхода и определяет шаги работы, их последовательность, а также правила распределения и назначения методов. Метод определяет способ достижения той или иной цели — выполнение шага работы.

Нотацией называют систему обозначений, используемых для описания некоторого класса моделей. Нотации бывают графические (представление моделей в виде графов, диаграмм, таблиц, схем и т. п.) и текстовые (описания моделей на формальных и естественных языках). В CASE-технологиях нотации используют для описания структуры проектируемой системы, элементов данных, этапов обработки и т. п.

Средства — инструментарий для поддержки методов, средства создания и редактирования графического проекта, организации проекта в виде иерархии уровней абстракции, а также проверки соответствия компонентов разных уровней. Различают:

* + CASE-средства анализа требований, проектирования спецификаций и структуры, редактирования интерфейсов (первое поколение CASE-I);
  + CASE-средства генерации исходных текстов и реализации интегрированного окружения поддержки полного жизненного цикла разработки программного обеспечения (второе поколение CASE-II).

CASE-I в основном включают средства для поддержки графических моделей, проектирования спецификаций, экранных редакторов и словарей данных. CASE-II отличаются большими возможностями и обеспечивают: контроль, анализ и связывание системной информации и информации по управлению процессом проектирования, построение прототипов и моделей системы, тестирование, верификацию и анализ сгенерированных программ.

Автоматизируя трудоемкие операции, современные CASE-средства существенно повышают производительность труда программистов и улучшают качество создаваемого программного обеспечения.

CASE-средства:

* + обеспечивают автоматизированный контроль совместимости спецификаций проекта;
  + уменьшают время создания прототипа системы;
  + ускоряют процесс проектирования и разработки;
  + автоматизируют формирование проектной документации для всех этапов жизненного цикла в соответствии с современными стандартами;
  + частично генерируют коды программ для различных платформ разработки;
  + поддерживают технологии повторного использования компонентов системы;
  + обеспечивают возможность восстановления проектной документации по имеющимся исходным кодам.

Появление CASE-технологий изменило все этапы жизненного цикла программного обеспечения, при этом наибольшие изменения касаются анализа и проектирования, которые предполагают строгое и наглядное описание разрабатываемого программного обеспечения.

В таблице 1 показано, какие качественные изменения процесса разработки программного обеспечения происходят при переходе к использованию CASE-средств.

Таблица 1 - Качественные изменения процесса разработки ПО

|  |  |
| --- | --- |
| **Традиционная разработка** | **Разработка с использование**  **CASE-средств** |
| Основные усилия на кодирование и тестирование | Основные усилия на анализ и проектирование |
| **«**Бумажные» спецификации | Быстрое интерационное прототипирование |
| Ручное кодирование | Автоматическая генерация кодов |
| Ручное документирование | Автоматическая генерация документации |
| Тестирование кодов | Автоматический контроль проекта |
| Сопровождение кодов | Сопровождение спецификаций проектирования |

Использование CASE-средств позволяет существенно снизить трудозатраты на разработку сложного программного обеспечения (табл. 2) в основном за счет автоматизации процессов документирования и контроля.

Таблица 2 – Трудозатраты на разработку ПО

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Способ разработки** | **Трудозатраты этапа разработки, %** | | | |
| **Анализ** | **Проектирование** | **Кодирование** | **Тестирование** |
| Традиционная разработка | 20 | 15 | 20 | 45 |
| Структурный подход | 30 | 30 | 15 | 25 |
| CASE-технологии | 40 | 40 | 5 | 15 |

Однако следует иметь в виду, что современные CASE-средства дороги, а их использование требует более высокой квалификации разработчиков. Следовательно, их имеет смысл использовать в сложных проектах, причем чем сложнее разрабатываемое программное обеспечение, тем больше выигрыш от использования CASE-технологий. На сегодняшний день практически все промышленно производимое сложное программное обеспечение разрабатывается с использованием CASE-средств.

**Rational Objectory Process – модель жизненного цикла (методология объективно-ориентированного программирования)**

Известно, что объектно-ориентированное проектирование программного обеспечения стало результатом появления объектно-ориентированного программирования (ООП). Ранние стадии описания предметной области и разработки архитектуры системы не поддерживались, первые варианты использования объектно-ориентированной методологии в большей степени являлись чистым повторением принципов ООП. Такие вопросы, как декомпозиция предметной области, спецификация требований, интерфейс пользователя не рассматривались, однако успехи объектно-ориентированного программирования заставили распространить новую технологию на весь жизненный цикл ПО. В результате все преимущества подхода применяются не только в процессе кодирования, но и на более ранних этапах. Таким образом, были определены основные компоненты методологии:

* + модель жизненного цикла;
  + действия;
  + нотация языка.

**Жизненный цикл UML2(Rational Objectory Process)**

Фирма Rational Software, разработавшая язык UML, предложила также и свою модель ЖЦ, которая называется Rational Objectory Process (ROP). Основные свойства ROP-технологии:

* + итеративный процесс, в течение которого происходит последовательное уточнение результатов;
  + направление на создание моделей, а не на разработку других элементов проекта (например, текстовых документов).

Действия Rational Objectory Process определяются в первую очередь блоками использования (use case) (рис. 9).

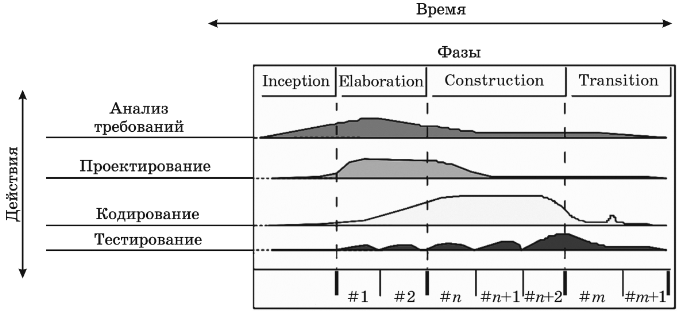


Рисунок 9 – Модель жизненного цикла UML

Rational Objectory Process разбит на циклы, каждый из которых, в свою очередь, состоит из четырех фаз:

* + начальная стадия (Inception);
  + разработка (Elaboration);
  + конструирование (Construction);
  + ввод в эксплуатацию (Transition).

Результатом работы каждого такого цикла является своя версия программной системы.

Каждая стадия завершается в четко определенной контрольной точке (milestone). В этот момент времени должны достигаться важные результаты и приниматься критически важные решения о дальнейшей разработке.

Начальная стадия может принимать множество разных форм. Для крупных проектов — это всестороннее изучение всех возможностей реализации на протяжении нескольких месяцев. Здесь же вырабатывается бизнес-план проекта, определяются его стоимость, примерный доход, а также ограничения ресурсов — иными словами, выполняется некоторый начальный анализ оценки проекта.

Окончанием начального этапа могут служить следующие результаты:

* + начальный проектный словарь терминов;
  + общее описание системы — основные требования к проекту, его характеристики и ограничения;
  + начальная модель вариантов использования;
  + начальный бизнес-план;
  + план проекта, отражающий стадии и итерации;
  + один или несколько прототипов.

На стадии разработки выявляются более детальные требования к системе, выполняются высокоуровневый анализ предметной области и проектирование базовой архитектуры системы, создается план конструирования, устраняются наиболее рискованные элементы проекта.

Самым важным результатом стадии разработки является описание базовой архитектуры будущей системы. Эта архитектура включает:

* + модель предметной области, которая служит отправным пунктом для формирования основных абстракций предметной области;
  + технологическую платформу, определяющую основные элементы технологии реализации системы и их взаимодействие.

Стадия разработки занимает примерно пятую часть времени создания проекта, результатом которой являются:

* + оценка времени реализации каждого варианта использования;
  + идентификация всех наиболее серьезных рисков и возможности их ликвидации.

Сущность стадии конструирования заключается в определении последовательности итераций конструирования и вариантов использования, реализуемых на каждой итерации, которые являются одновременно инкрементными и повторяющимися.

Результатом стадии конструирования является продукт, готовый к передаче пользователям, содержащий, как правило, руководство пользователей и готовый к интеграции на требуемых платформах.

Назначением стадии ввода в эксплуатацию является передача готового продукта в полное распоряжение конечных пользователей. Данная стадия включает:

* + бета-тестирование, позволяющее убедиться, что новая система соответствует ожиданиям пользователей;
  + параллельное функционирование с существующей системой, которая подлежит постепенной замене;
  + оптимизацию производительности;
  + обучение пользователей и специалистов службы сопровождения.

**Вывод к 1 главе**

Мы изучили основные понятия и определения, типы программного обеспечения, жизненный цикл и этапы разработки программного обеспечения и модели жизненного цикла ПО в сфере технологий разработки программного обеспечения. Таким образом, мы изучили, какие знания необходимы для работы в сфере разработки программного обеспечения.

# ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

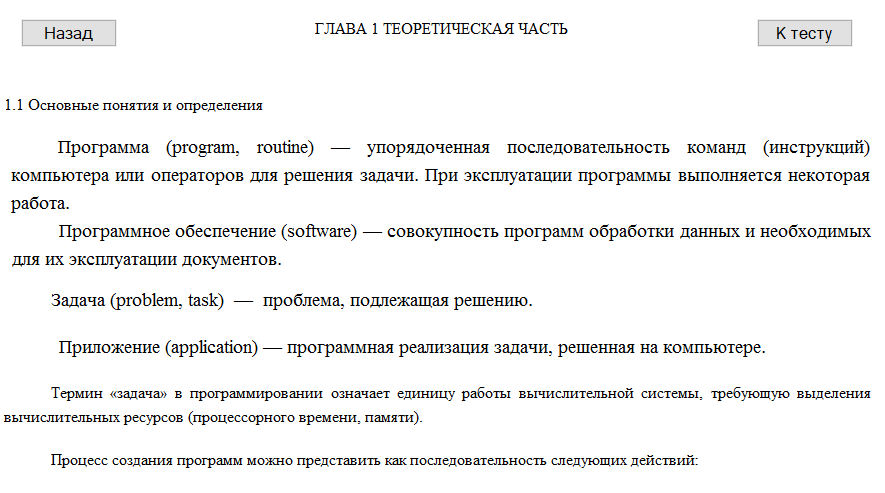
Программа для курсовой работы была разработана в среде программирования Microsoft Visual Studio 2022 и реализована на языке программирования Visual C#.

Для того чтобы запустить программу на выполнение, необходимо перейти в папку kursovayaRabota открыть ее в Visual Studio 2022 выбрать Form20 и запустить код.

# 2.1. Разработка электронного конспекта

Для того, чтобы разработать электронный конспект, понадобилось создать Form11, в котором был отражен теоретический материал по вышеупомянутой теме.

На форме представлены два элемента button и webBrowser. Это продемонстрировано на рисунке 10.

Рисунок 10 – Разработка электронного пособия

При помощи элемента управления button при переходе на данную форму происходит.

public partial class Form11 : Form

{

public Form11()

{

InitializeComponent();

string Dir = Directory.GetCurrentDirectory();

webBrowser1.Url = new Uri(String.Format("file:///{0}/../../Resources/teorya.html", Dir));

}

private void button1\_Click(object sender, EventArgs e)

{

Form2 f2 = new Form2();

this.Hide();

f2.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

Form1 f1 = new Form1();

this.Hide();

f1.Show();

}

}

# 2.2 Разработка тестовой программы

Для доступа к каждому следующему вопросу пользователю необходимо ответить на текущий вопрос, следуя указаниям в задании, и нажать на кнопку «Подтвердить и продолжить». В противном случае программа выдает сообщение: «Вы не ответили на вопрос». При нажатии кнопки «Пропуск вопроса» происходит переход к следующему вопросу без сообщения, при этом ответ приравнивается к неправильному.

Если пользователь ответил правильно или неправильно на вопрос, то выдаются соответствующие сообщения: «Вы не ответили на вопрос», «Вы ответили на вопрос верно», «Вы ответили на вопрос неверно», в случае с несколькими вопросами на одной форме выдаются сообщения: «Вы не ответили на вопрос», «Вы ответили на вопросы верно», «Вы ответили верно не на все вопросы», «Вы ответили на вопросы неверно».

**Перейдем к описанию** работы элемента CheckBox: при помощи флажков, пользователь может выбрать один или несколько элементов, а затем в коде проверяем ответ с помощью свойства компоненты «.Checked» на вопрос показанный на рисунке 11.

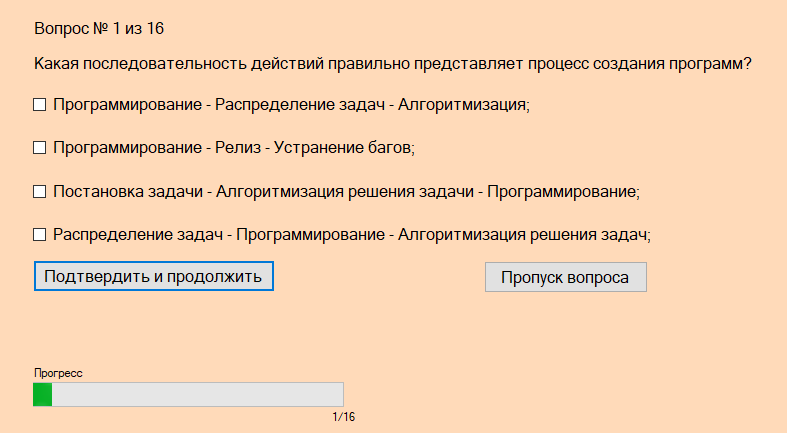


Рисунок 11 – Элемент управления CheckBox

//DLL

public static void Skip\_question(int numb\_q)

{

mass[numb\_q] = "Неверно";

}

public static int Vopros1(CheckBox CB1, CheckBox CB2, CheckBox CB3, CheckBox CB4)

{

if ((!CB1.Checked) && (!CB2.Checked) && (!CB3.Checked) && (!CB4.Checked))

{

MessageBox.Show("Вы не выбрали ни один из вариантов", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if ((!CB1.Checked) && (!CB2.Checked) && (CB3.Checked) && (!CB4.Checked))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[0] = "Верно";

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[0] = "Неверно";

return 0;

}

}

//Form2

private void button1\_Click(object sender, EventArgs e)

{

int rez = Class1.Vopros1(checkBox1, checkBox2, checkBox3, checkBox4);

if (!(rez == -1))

{

Form3 f3 = new Form3();

this.Hide();

f3.Show();

}

}

private void button2\_Click(object sender, EventArgs e)

{

Class1.Skip\_question(0);

Form3 f3 = new Form3();

this.Hide();

f3.Show();

}

Следующим использованным элементом управления является TextBox, используется для ввода пользователем текста, затем в коде считывается введенный текст, с помощью свойства компоненты «.Text», и сравнивается с правильным ответом на вопрос, представленный на рисунке 12.

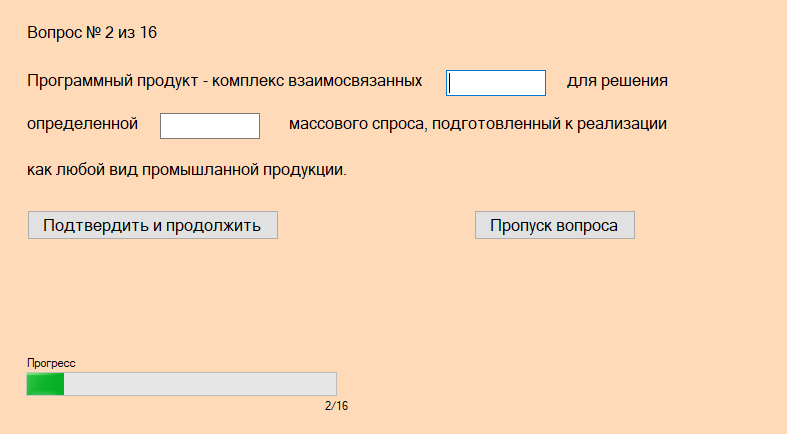


Рисунок 12 – Элемент управления TextBox

[//DLL](file:///\\DLL)

public static void Skip\_question(int numb\_q)

{

mass[numb\_q] = "Неверно";

}

public static int Vopros2(string right\_ans\_1, string right\_ans\_2\_1, string right\_ans\_2\_2, string ans\_1, string ans\_2)

{

if ((ans\_1 == "") || (ans\_2 == ""))

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if ((ans\_1 == right\_ans\_1) && ((ans\_2 == right\_ans\_2\_1)|| (ans\_2 == right\_ans\_2\_2)))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[1] = "Верно";

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[1] = "Неверно";

return 0;

}

}

//Form3

private void button1\_Click(object sender, EventArgs e)

{

int rez = Class1.Vopros2("программ", "задачи", "проблемы", textBox1.Text, textBox2.Text);

if (!(rez == -1))

{

Form4 f4 = new Form4();

this.Hide();

f4.Show();

}

}

private void button2\_Click(object sender, EventArgs e)

{

Class1.Skip\_question(1);

Form4 f4 = new Form4();

this.Hide();

f4.Show();

}

Далее был использован элемент управления RadioButton. Для того, чтобы выбрать один из предложенных вариантов, затем в коде с помощью свойства компоненты «.Checked» проверяем правильность выбранного ответа на вопрос представленный на рисунке 13.

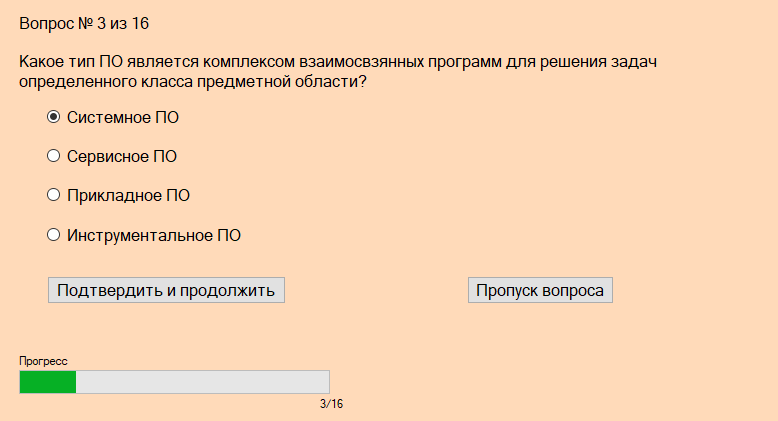


Рисунок 13 – Элемент управления RadioButton

//DLL

public static void Skip\_question(int numb\_q)

{

mass[numb\_q] = "Неверно";

}

public static void Vopros3(RadioButton RB1, RadioButton RB2, RadioButton RB3, RadioButton RB4)

{

if ((!RB1.Checked) && (!RB2.Checked) && (RB3.Checked) && (!RB4.Checked))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[2] = "Верно";

}

else

{

MessageBox.Show("Вы ответили неправильно", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[2] = "Неверно";

}

}

//Form4

private void button1\_Click(object sender, EventArgs e)

{

Class1.Vopros3(radioButton1, radioButton2, radioButton3, radioButton4);

Form5 f5 = new Form5();

this.Hide();

f5.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

Class1.Skip\_question(2);

Form5 f5 = new Form5();

this.Hide();

f5.Show();

}

Следующий элемент управления – ListBox. Для того, чтобы сопоставить предложенные варианты в нужные строчки с помощью свойства «.Text» проверяем выбран ли ответ, при помощи свойства компоненты «.Items[i]» проверяем правильно ли сопоставлены ответы на вопрос представленный на рисунке 14.

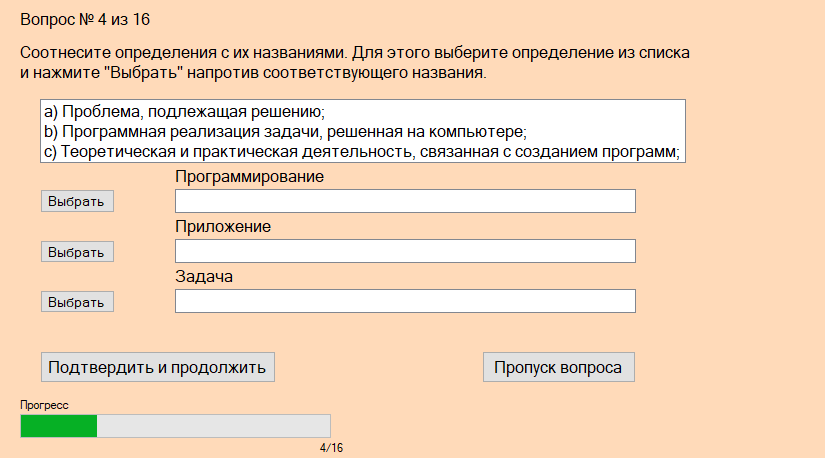


Рисунок 14 – Элемент управления ListBox

//DLL

public static void Skip\_question(int numb\_q)

{

mass[numb\_q] = "Неверно";

}

public static void chooseListBox(ListBox LB1, ListBox LB2)

{

if (LB1.SelectedIndex == 0)

{

LB2.Items.Add(LB1.Text);

LB2.Text = Convert.ToString(LB1.SelectedIndex);

LB1.Items.RemoveAt(0);

}

if (LB1.SelectedIndex == 1)

{

LB2.Items.Add(LB1.Text);

LB2.Text = Convert.ToString(LB1.SelectedIndex);

LB1.Items.RemoveAt(1);

}

if (LB1.SelectedIndex == 2)

{

LB2.Items.Add(LB1.Text);

LB2.Text = Convert.ToString(LB1.SelectedIndex);

LB1.Items.RemoveAt(2);

}

}

public static int Vopros4(ListBox LB1, ListBox LB2, ListBox LB3)

{

if ((LB1.Text == "") || (LB2.Text == "") || (LB3.Text == ""))

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if ((LB1.Items[0] == "c) Теоретическая и практическая деятельность, связанная с созданием программ;") && (LB2.Items[0] == "b) Программная реализация задачи, решенная на компьютере;") && (LB3.Items[0] == "a) Проблема, подлежащая решению;"))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[3] = "Верно";

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[3] = "Неверно";

return 0;

}

}

//Form

private void button1\_Click(object sender, EventArgs e)

{

Class1.chooseListBox(listBox1, listBox2);

}

private void button2\_Click(object sender, EventArgs e)

{

Class1.chooseListBox(listBox1, listBox3);

}

private void button3\_Click(object sender, EventArgs e)

{

Class1.chooseListBox(listBox1, listBox4);

}

private void button4\_Click(object sender, EventArgs e)

{

int rez = Class1.Vopros4(listBox2, listBox3, listBox4);

if (!(rez == -1))

{

Form6 f6 = new Form6();

this.Hide();

f6.Show();

}

}

private void button5\_Click(object sender, EventArgs e)

{

Class1.Skip\_question(3);

Form6 f6 = new Form6();

this.Hide();

f6.Show();

}

Далее **перейдем к описанию** работы элемента управления ComboBox. Пользователь может выбрать элемент списка, в коде используется свойство компоненты «.SelectedIndex», для того чтобы определить правильно ли ответил пользователь на вопрос представленный на рисунке 15.

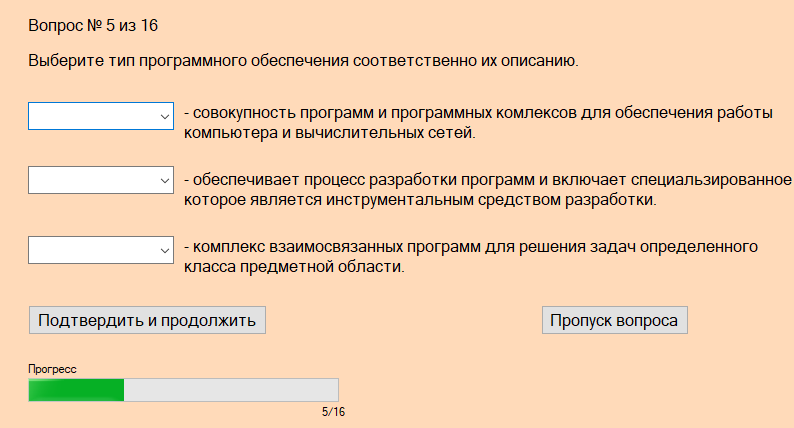


Рисунок 15 – Элемент управления ComboBox

//DLL

public static void Skip\_question(int numb\_q)

{

mass[numb\_q] = "Неверно";

}

public static int Vopros5(ComboBox CB1, ComboBox CB2, ComboBox CB3)

{

if ((CB1.SelectedIndex == -1) || (CB2.SelectedIndex == -1) || (CB3.SelectedIndex == -1))

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if ((CB1.SelectedIndex == 1) && (CB2.SelectedIndex == 2) && (CB3.SelectedIndex == 0))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[4] = "Верно";

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[4] = "Неверно";

return 0;

}

}

//Form6

private void button1\_Click(object sender, EventArgs e)

{

int rez = Class1.Vopros5(comboBox1, comboBox2, comboBox3);

if (!(rez == -1))

{

Form7 f7 = new Form7();

this.Hide();

f7.Show();

}

}

private void button2\_Click(object sender, EventArgs e)

{

Class1.Skip\_question(4);

Form7 f7 = new Form7();

this.Hide();

f7.Show();

}

Элемент управления HScrollBar. Для того, чтобы пользователь мог прокрутить и выбрать нужное значение. Компоненте задаются максимальные и минимальные значения, при изменении значение записывается в label, в коде с помощи свойства «.Value» сравнивается ответ пользователя с правильным ответом на вопрос, представленный на рисунке 16.

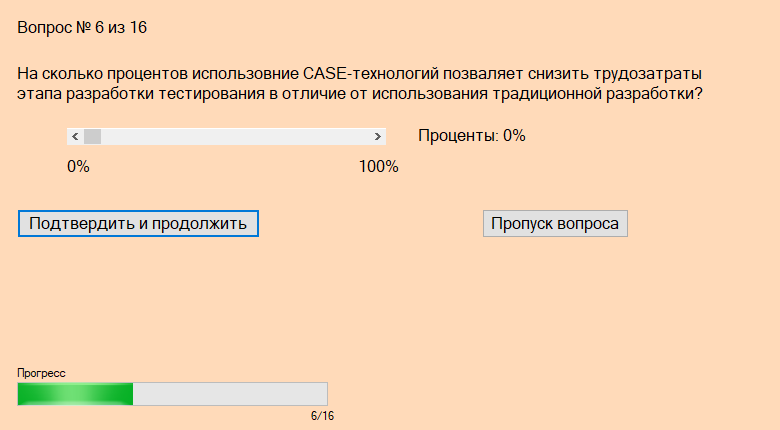


Рисунок 16 – Элемент управления HScrollBar

//DLL

public static void Skip\_question(int numb\_q)

{

mass[numb\_q] = "Неверно";

}

public static void Vopros6(HScrollBar SB)

{

if (SB.Value == 30)

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[5] = "Верно";

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[5] = "Неверно";

}

}

//From7

private void button1\_Click(object sender, EventArgs e)

{

Class1.Vopros6(hScrollBar1);

Form8 f8 = new Form8();

this.Hide();

f8.Show();

}

private void hScrollBar1\_Scroll(object sender, ScrollEventArgs e)

{

label1.Text = string.Format("Проценты: {0}%", hScrollBar1.Value);

}

private void button2\_Click(object sender, EventArgs e)

{

Class1.Skip\_question(5);

Form8 f8 = new Form8();

this.Hide();

f8.Show();

}

**Перейдем к описанию** работы элементов управления TrackBar. Для того, чтобы пользователь мог выбирать значение, при помощи ползунка. В компоненте устанавливается шаг, снизу подписываются года в label, в коде используется свойство «.Value», чтобы определить индекс ответа пользователя, и определить его верность на вопрос, представленный на рисунке 17.

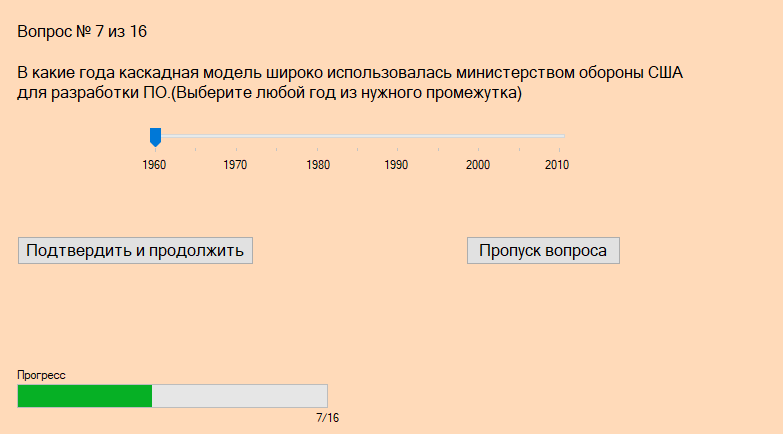


Рисунок 17 – Элемент управления TrackBar

//DLL

public static void Skip\_question(int numb\_q)

{

mass[numb\_q] = "Неверно";

}

public static void Vopros7(TrackBar TB)

{

if ((TB.Value == 2) || (TB.Value == 3) || (TB.Value == 4))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[6] = "Верно";

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[6] = "Неверно";

}

}

//Form8

private void button1\_Click(object sender, EventArgs e)

{

Class1.Vopros7(trackBar1);

Form9 f9 = new Form9();

this.Hide();

f9.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

Class1.Skip\_question(6);

Form9 f9 = new Form9();

this.Hide();

f9.Show();

}

Заключительным элементом управления является CheckedListBox. Для выбора пользователем нескольких вариантов ответа, в коде были использованы свойство «.CheckItems.Count» для определения количества выбранных ответов и свойство «.CheckItems» для определения выбранных вариантов и сравнения с правильными ответами на вопрос, представленный на рисунке 18.

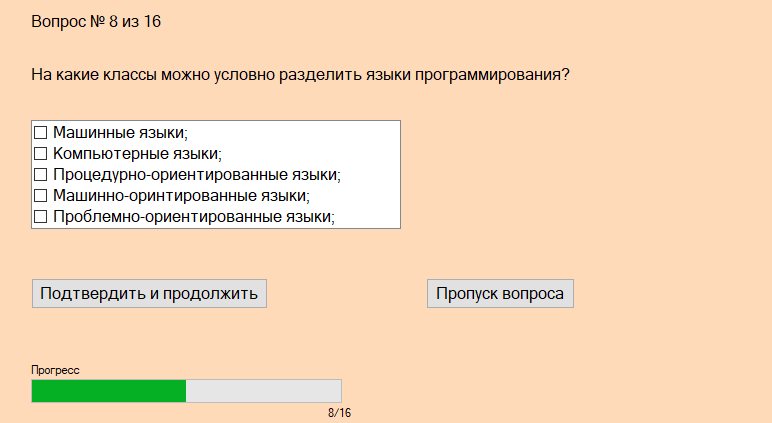


Рисунок 18 – Элемент управления CheckedListBox

//DLL

public static void Skip\_question(int numb\_q)

{

mass[numb\_q] = "Неверно";

}

public static int Vopros8(CheckedListBox CLB)

{

if (CLB.CheckedItems.Count == 0)

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else

{

int count = 0;

foreach(string index in CLB.CheckedItems)

{

if ((index== "Машинные языки;") || (index == "Процедурно-ориентированные языки;") || (index == "Машинно-оринтированные языки;") || (index == "Проблемно-ориентированные языки;"))

{

count += 1;

}

}

if (count == 4)

{

MessageBox.Show("Вы ответили правильно ", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[7] = "Верно";

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[7] = "Неверно";

return 0;

}

}

//Form9

private void button1\_Click(object sender, EventArgs e)

{

int rez = Class1.Vopros8(checkedListBox1);

if (!(rez == -1))

{

Form10 f10 = new Form10();

this.Hide();

f10.Show();

}

}

private void button2\_Click(object sender, EventArgs e)

{

Class1.Skip\_question(7);

Form10 f10 = new Form10();

this.Hide();

f10.Show();

}

**Перейдем к описанию** итоговой формы, которая представлена на рисунке 19.

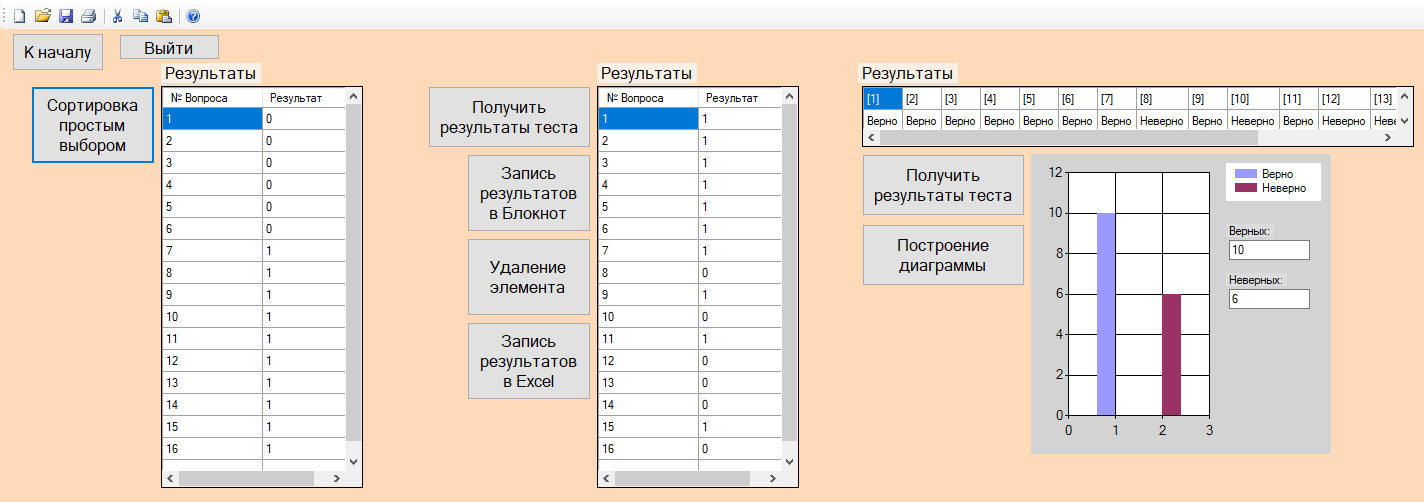


Рисунок 19 – Итоговая форма

На данной форме выводится массив с результатами прохождения теста в элемент dataGridView, с помощью которого можно узнать в каких заданиях была допущена ошибка. Также на форме присутствует диаграмма и два textBox с общими результатами. Вдобавок на форме есть две кнопки, одна из которых позволяет, при необходимости, записать результаты прохождения теста в редактор Блокнот, а вторая возвращает пользователя на главную форму. Также на форме есть еще 2 элемента dataGridView, при нажатии на кнопки в одну заполняется результаты теста в формате 1/0, а в другую отсортированные простым выбором результаты теста. Еще есть панель ToolStrip, в ней работает кнопка сохранить, которая создает pdf файл и записывает туда результаты теста в формате 1/0, а также кнопка открыть, при нажатии на которую открывается результат теста из Excel, который был добавлен при нажатии на кнопку «Запись результатов в Excel».

# 2.3. Дополнительные возможности программы

Разработка программного проекта курсовой работы предполагала программирование следующих дополнительных возможностей, с целью повысить оригинальность и индивидуальность работы.

Было реализовано:

1. Создание авторизации пользователей;
2. Построение диаграммы, отображающей результаты тестирования;
3. Запись результатов ответов тестовой программы (в формате Верно/Неверно) в одномерный массив с выводом в DGV.
4. Запись результатов ответов тестовой программы (в формате 1/0) в текстовый файл;
5. Создание анимационной заставки;
6. Запись результатов ответов тестовой программы (в формате 1/0) в Excel и сохранение скриншота диаграммы в формате .jpg;
7. Запись результатов ответов тестовой программы (в формате 1/0) в pdf;
8. Создание панели ToolStrip, при нажатии на кнопку «открыть», открывается фотография диаграммы из Excel;
9. Удаление выбранного в InputBox элемента массива;
10. Запись отсортированных простым выбором ответов тестовой программы (в формате 1/0) в dataGridView;

Перейдем к **рассмотрению вышеперечисленных** возможностей.

При запуске проекта появляется окно авторизации пользователей. При этом использовалось 2 textBox для того, чтобы пользователь ввел логин и пароль, button по нажатию которой проверяется есть ли в базе данных, переданный логин и пароль. Дизайн формы представлен на рисунке 20.

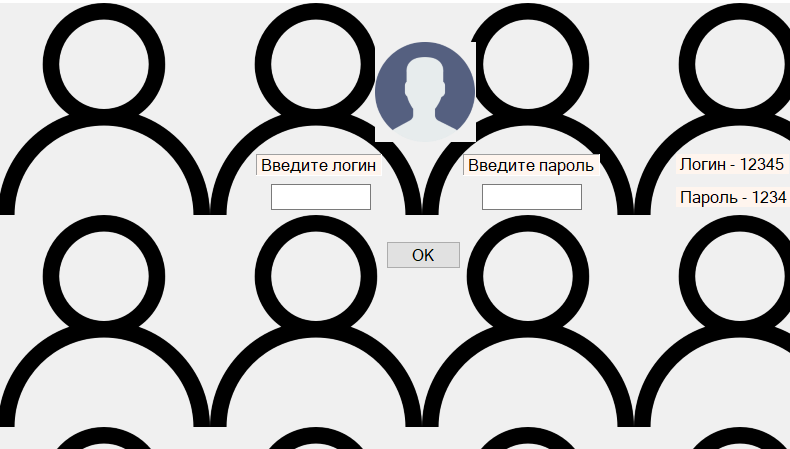


Рисунок 20 – Форма с авторизацией

Программный код для реализации этой задачи представлен ниже:

//DLL

public static void Avtoriz(Form A, Form B, TextBox t1, TextBox t2)

{

string Log = t1.Text;

string Pass = t2.Text;

var p = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Users1.mdb");

p.Open();

var c = new OleDbCommand("SELECT [Логин], [Пароль] FROM [users]", p);

OleDbDataReader reader = c.ExecuteReader();

bool f = true;

while (reader.Read() & (f))

{

if ((Log == reader[0].ToString()) & (Pass == reader[1].ToString()))

{

MessageBox.Show("Вы авторизованы!", "Авторизация", MessageBoxButtons.OK, MessageBoxIcon.Information);

f = false;

A.Hide();

B.Show();

}

}

reader.Close();

if (f)

{

MessageBox.Show("Неправильный логин или пароль!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//Form1

private void button1\_Click(object sender, EventArgs e)

{

Form2 f2 = new Form2();

Class1.Avtoriz(this, f2, textBox1, textBox2);

}

**Перейдем к описанию** следующей дополнительной возможности – построение диаграммы. На итоговой форме тестирования пользователь может увидеть наглядный результат прохождения теста в виде диаграммы, созданной с помощью элемента Chart. В свойстве **Series** были переименованы на «Верно» и «Неверно» члены диаграммы. Построенная диаграмма была продемонстрирована на рис. 19.

Программный код для реализации данной задачи представлен ниже:

//DLL

public static int n = 0;

public static void Diagram(Chart C, TextBox TB1, TextBox TB2)

{

C.Series[0].Points.AddXY(1, n);

C.Series[1].Points.AddXY(2, Math.Abs(n-8));

Vivod(TB1, n);

Vivod(TB2, 8-n);

}

public static void Vivod(TextBox t, double c)

{

t.Text = Convert.ToString(c);

}

//Form10

private void button2\_Click(object sender, EventArgs e)

{

Class1.Diagram(chart1, textBox1, textBox2);

}

**Перейдем к описанию** следующей дополнительной возможности – запись результатов ответов тестовой программы (в формате 1/0) в текстовый файл. Пользователь может осуществить это с помощью нажатия соответствующую кнопку на итоговой форме тестирования.

Для этого используется результаты тестирования записанные в формате 1/0 в элемент dataGridView и кнопка, которая создает текстовый файл и заносит туда результаты.

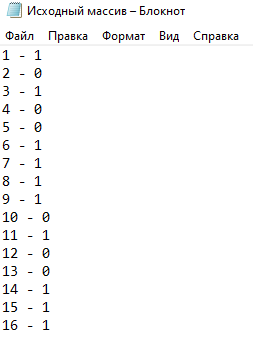


Рисунок 21 – Результат записи в программу Блокнот

Программный код для реализации задачи представлен ниже:

[//DLL](file:///\\DLL)

public static int[] massNum = new int[16];

public static void ZapisBloknot()

{

StreamWriter rez = File.CreateText("Исходный массив.txt");

for (int i = 0; i < massNum.Length; i++)

{

rez.WriteLine((i+1).ToString() + " - " + massNum[i]);

}

rez.Close();

System.Diagnostics.Process.Start("Исходный массив.txt");

}

//Form10

private void button4\_Click(object sender, EventArgs e)

{

Class1.ZapisBloknot();

}

При запуске проекта проигрывается анимационная заставка.

Она была реализована с помощью элемента timer, а также ProgressBar и label, благодаря которым показывается шкала загрузки и ее значение. Еще на форме есть PictureBox.

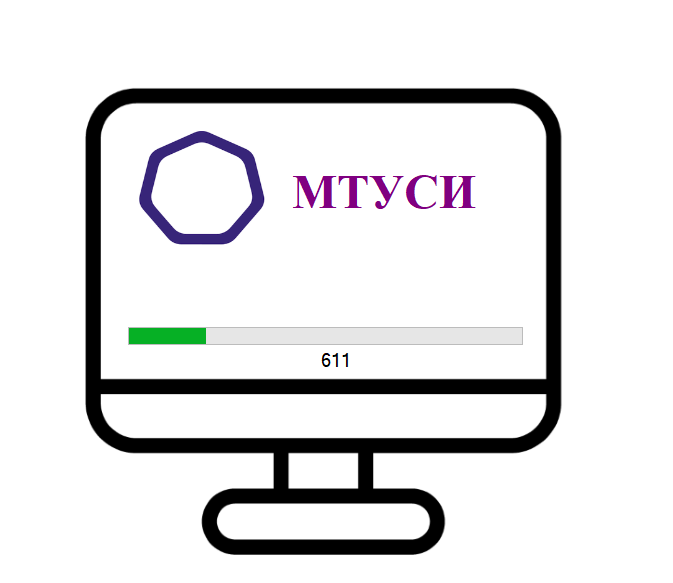


Рисунок 22 – Анимационная заставка

//From20

private void timer1\_Tick(object sender, EventArgs e)

{

progressBar1.Maximum= 1000;

progressBar1.Value= 0;

for (int i = 0; i <= progressBar1.Maximum - 1; i++)

{

progressBar1.Increment(1);

label1.Text=progressBar1.Value.ToString();

label1.Refresh();

}

if (progressBar1.Value == progressBar1.Maximum)

{

timer1.Enabled = false;

Form21 f21 = new Form21();

this.Hide();

f21.ShowDialog();

}

}

Cледующей дополнительной возможностью является запись результатов ответов тестовой программы (в формате 1/0) в Excel. Пользователь может осуществить это с помощью нажатия соответствующую кнопку на итоговой форме тестирования.

Для этого используется элемент button, при нажатии на который создастся новый файл Excel, занесется результаты тестирования в таблицу, а затем по этим данным построится диаграмма, фотографию которой сохранит программа.

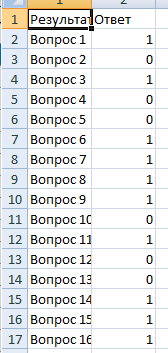


Рисунок 23 – Результат записи в таблицу Excel

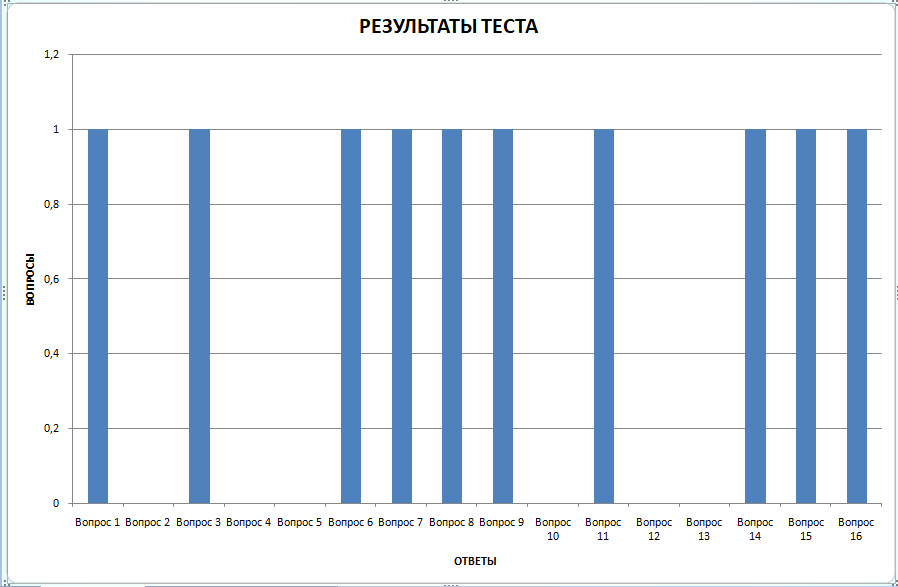


Рисунок 24 – Диаграмма в Excel, построенная по результатам теста

//Form 10

private void button7\_Click(object sender, EventArgs e)

{

Microsoft.Office.Interop.Excel.Application excelApp = new Microsoft.Office.Interop.Excel.Application();

excelApp.Visible = true;

var t = Type.Missing;

var Book = excelApp.Workbooks.Add(t);

var Lists = Book.Worksheets;

Worksheet List = Lists.Item[1];

List.Cells[1, 1] = "Результаты тестирования";

List.Range["A2", t].Value2 = "Вопрос 1";

List.Range["A3", t].Value2 = "Вопрос 2";

List.Range["A4", t].Value2 = "Вопрос 3";

List.Range["A5", t].Value2 = "Вопрос 4";

List.Range["A6", t].Value2 = "Вопрос 5";

List.Range["A7", t].Value2 = "Вопрос 6";

List.Range["A8", t].Value2 = "Вопрос 7";

List.Range["A9", t].Value2 = "Вопрос 8";

List.Range["A10", t].Value2 = "Вопрос 9";

List.Range["A11", t].Value2 = "Вопрос 10";

List.Range["A12", t].Value2 = "Вопрос 11";

List.Range["A13", t].Value2 = "Вопрос 12";

List.Range["A14", t].Value2 = "Вопрос 13";

List.Range["A15", t].Value2 = "Вопрос 14";

List.Range["A16", t].Value2 = "Вопрос 15";

List.Range["A17", t].Value2 = "Вопрос 16";

List.Range["B1", t].Value2 = "Ответ";

List.Range["B2", t].Value2 = mas[0];

List.Range["B3", t].Value2 = mas[1];

List.Range["B4", t].Value2 = mas[2];

List.Range["B5", t].Value2 = mas[3];

List.Range["B6", t].Value2 = mas[4];

List.Range["B7", t].Value2 = mas[5];

List.Range["B8", t].Value2 = mas[6];

List.Range["B9", t].Value2 = mas[7];

List.Range["B10", t].Value2 = mas[8];

List.Range["B11", t].Value2 = mas[9];

List.Range["B12", t].Value2 = mas[10];

List.Range["B13", t].Value2 = mas[11];

List.Range["B14", t].Value2 = mas[12];

List.Range["B15", t].Value2 = mas[13];

List.Range["B16", t].Value2 = mas[14];

List.Range["B17", t].Value2 = mas[15];

Microsoft.Office.Interop.Excel.Chart Diagr = excelApp.Charts.Add(t, t, t, t);

Diagr.SetSourceData(List.Range["A2", "B17"],

Microsoft.Office.Interop.Excel.XlRowCol.xlColumns);

Diagr.ChartType = XlChartType.xlColumnClustered;

Diagr.HasLegend = false;

Axis Goriz\_Os = Diagr.Axes(Microsoft.Office.Interop.Excel.XlAxisType.xlCategory,

Microsoft.Office.Interop.Excel.XlAxisGroup.xlPrimary);

Goriz\_Os.HasTitle = true;

Goriz\_Os.AxisTitle.Text = "ОТВЕТЫ";

Axis Vertic\_Os = Diagr.Axes(Microsoft.Office.Interop.Excel.XlAxisType.xlValue,

Microsoft.Office.Interop.Excel.XlAxisGroup.xlPrimary);

Vertic\_Os.HasTitle = true;

Vertic\_Os.AxisTitle.Text = "ВОПРОСЫ";

string location = System.Reflection.Assembly.GetExecutingAssembly().Location;

string path = System.IO.Path.GetDirectoryName(location);

string file = path + "\\Excel.jpg";

excelApp.ActiveChart.Export(file, t, t);

}

Еще одной дополнительной возможность является панель инструментов.

Для ее реализации используется компонента ToolStrip, на которой находятся кнопки сохранить и открыть. Кнопка «Сохранить» отвечает за создание pdf файла и записи в него результатов тестирования в формате 1/0, кнопка «Открыть» позволяет в свою очередь открыть фотографию диаграммы из Excel по результатом тестирования.



Рисунок 25 – Панель инструментов ToolStrip

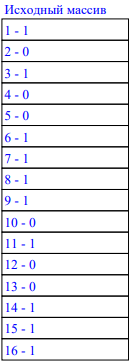


Рисунок 26 – Результат записи в pdf файл

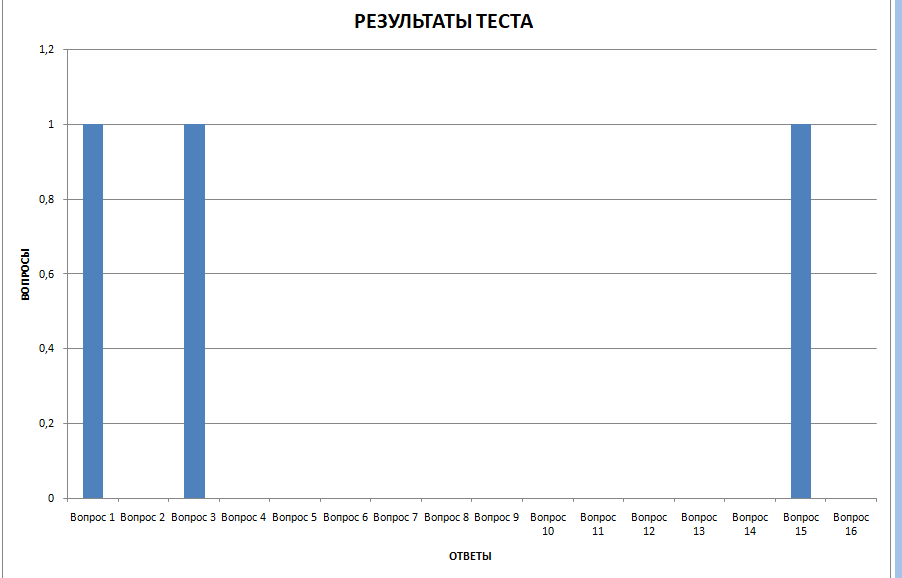


Рисунок 27 – Фотография диаграммы из Excel открытая при помощи копки «Открыть» на панели инструментов

//Form10

private void открытьToolStripButton\_Click(object sender, EventArgs e)

{

OpenFileDialog otkr = new OpenFileDialog();

otkr.DefaultExt = "\*.jpg";

otkr.Filter = "Файлы изображений (\*.jpg)|\*.jpg";

otkr.Title = "Выберите полученный скриншот диаграммы";

if (otkr.ShowDialog() != DialogResult.OK)

{

MessageBox.Show("Вы не выбрали файл", "Открыть", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

System.Diagnostics.Process.Start(otkr.FileName);

}

private void сохранитьToolStripButton\_Click(object sender, EventArgs e)

{

Class1.Add\_pdf();

}

Другой дополнительной возможностью является удаление элемента массива.

Для ее реализации потребовалась компонента dataGridView, в которую записывается результирующий массив, после удаления элемента. Также компонента Button, из-за которой удаляется предыдущая запись из DataGridView и записывается новый результат.

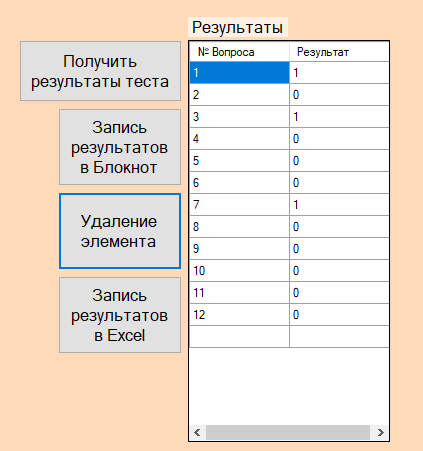


Рисунок 28 – Результат удаления элемента из массива

//Form10

int[] mas = Class1.massNum;

int[] delMas = new int[16];

public Form10()

{

InitializeComponent();

Class1.Get\_Delmas(ref delMas);

}

private void button6\_Click(object sender, EventArgs e)

{

string number = Interaction.InputBox("Введите номер строки, которую хотите удалить", "Удаление элемента", "", 500, 300);

int n = Convert.ToInt32(number);

dataGridView2.Rows.Clear();

Class1.Del\_El(ref delMas, n-1);

Class1.New\_Mas(delMas, dataGridView2);

}

Последней дополнительной возможностью является запись массива отсортированного простым выбором.

Для ее создания потребовалась компонента dataGridView, в которую записывается результат сортировки массива, а также button, благодаря которой происходит сортировка и заполнение dataGridView.

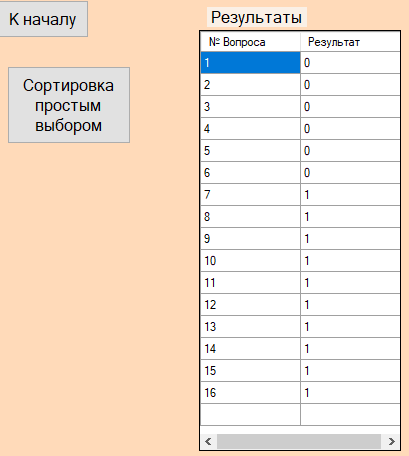


Рисунок 29 – Результат сортировки массива простым выбором

//Form10

private void button8\_Click(object sender, EventArgs e)

{

dataGridView3.Rows.Clear();

int[] mas = new int[16];

Class1.Sort\_Prost(ref mas);

Class1.New\_Mas(mas, dataGridView3);

}

По итогам работы над второй главой можно отметить, что было разработан электронный конспект и создана программа для тестирования пользователя по теме «Технология разработки программного обеспечения».

При этом были реализованы:

1. Разработан электронный конспект по теме «Технология разработки программного обеспечения»
2. Разработана тестовая программа, которая содержит 16 вопросов по теме курсовой работы.
3. Записаны результаты ответа на каждый вопрос в элемент управления dataGridView;
4. Создана диаграмма, отображающая результаты теста;
5. Создана система авторизации;
6. Создана анимационная заставка;
7. Реализована запись результатов тестирования в текстовый файл, Excel и pdf файл;
8. Создана панель инструментов на основе компонента ToolStrip;
9. Реализована запись результатов тестирования в формате 1/0 в компонент dataGridView.
10. Реализовано удаление элемента из массива результатов теста.
11. Реализована сортировка простым выбором и запись получившегося результата в dataGridView.

# ЗАКЛЮЧЕНИЕ

В рамках работы над курсовым проектом был разработан программный продукт, который предназначен для предоставления возможности дистанционного изучения материала по теме курсовой работы, а также для проверки знаний учащихся с помощью тестовой прграммы.

В результате выполненной работы были решены следующие задачи:

1. Проанализирован и структурирован теоретический материал по теме исследования;
2. Разработан электронный конспект и программы, проверяющая усвоение знаний по данной теме;
3. Реализована запись результатов ответов программы в формате «Верно/Неверно» в одномерный массив с выводом на элемент управления dataGridView;
4. Реализовано построение диаграммы, отображающей результаты тестирования;
5. Создана система авторизации;
6. Создана анимационная заставка;
7. Реализована запись результатов тестирования в текстовый файл, Excel и pdf файл;
8. Создана панель инструментов на основе компонента ToolStrip;
9. Реализована запись результатов тестирования в формате 1/0 в компонент dataGridView.
10. Реализовано удаление элементов из массива результатов теста.
11. Реализована сортировка простым выбором и запись получившегося результата в dataGridView.

**Таким образом,** цели и задачи, поставленные на курсовую работу, были успешно выполнены.

# СПИСОК ИМПЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гуриков С. Р. Введение в программирование на языке Visual C#: учебное пособие / С.Р. Гуриков. — Москва : ФОРУМ : ИНФРА-М, 2020. — 447 с
2. ГОСТ 7.32 -2017 Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления.
3. Зубкова Т. М. Технология разработки программного обеспечения / Т. М. Зубкова. — 3-е изд., стер. — Санкт-Петербург : Лань, 2023. — 252 с.

# ПРИЛОЖЕНИЕ A

1. Какая последовательность действий представляет процесс создания программ?
2. Программирование – Распределение задач – Алгоритмизация;
3. Программирование – Релиз – Устранение багов;
4. Постановка задачи – Алгоритмизация решения задачи – Программирование;
5. Распределение задач – Программирование – Алгоритмизация решения задач;

Правильный ответ: 3)

1. Вставьте вместо пропусков нужные слова.

Программный продукт – комплекс взаимосвязанных …. для решения определенной …. массового спроса, подготовленный к реализации как любой вид промышленной продукции.

Правильный ответ: программ; задач/проблем.

1. Какой тип ПО является комплексом взаимосвязанных программ для решения задач определенного класса предметной области?
2. Системное ПО
3. Сервисное ПО
4. Прикладное ПО
5. Инструментальное ПО

Правильный ответ: 3)

1. Соотнесите определения с их названиями.

Определения:

1. Проблема, подлежащая решению;
2. Программная реализация задачи, решенная на компьютере;
3. Теоретическая и практическая деятельность, связанная с созданием программ;

Названия:

1. Программирование
2. Приложение
3. Задача

Правильный ответ: a) 3; b) 2; c) 1.

1. Выберите тип программного обеспечения соответственно их описанию.
2. совокупность программ и программных комплексов для обеспечения работы компьютера и вычислительных сетей
3. обеспечивает процесс разработки программ и включает специализированное, которое является инструментальным средством разработки.
4. комплекс взаимосвязанных программ для решения задачи определенного класса предметной области.

Варианты ответов: Прикладное ПО, Системное ПО, Инструментальное ПО

Правильный ответ: 1) Системное ПО; 2) Инструментальное ПО; 3) Прикладное ПО.

1. На сколько процентов использование CASE-технологий позволяет снизить трудозатраты этапа разработки тестирования в отличие от использования традиционной разработки?

Правильный ответ: 30%

1. В какие года каскадная модель широко использовалась министерством обороны США для разработки ПО?

Правильный ответ; 1970-1980г.

1. На какие классы можно условно разделить языки программирования?
2. Машинные языки;
3. Компьютерные языки;
4. Процедурно-ориентированные языки;
5. Машинно-ориентированные языки;
6. Проблемно-ориентированные языки;

Правильный ответ: 1), 3), 4), 5).

1. Какая последовательность хода работы соответствует первой итерации по итеративной модели?
2. Анализ требований – Проектирование – Кодирование – Тестирование;
3. Анализ требований – Кодирование – Развертывание;
4. Кодирование – Тестирование – Развертывание – Эксплуатация;
5. Анализ требований – Проектирование – Кодирование – Эксплуатация;

Правильный ответ: 1).

1. Программный продукт должен быть соответствующим образом подготовлен к … , иметь необходимую техническую … , предоставить сервис и гарантию надежной работы, иметь товарный знак изготовителя, а также код гос регистрации.

Правильный ответ: эксплуатации/использованию, документацию.

1. Какой этап является первоначальным в каскадной схеме разработки ПО?
2. Проектирование
3. Кодирование
4. Реализация
5. Анализ

Правильный ответ: 4)

1. Соотнесите определения с их названиями.

Определения:

1. Специальные средства, встроенные в пакеты прикладных программ;
2. Транслирует всю программу без ее выполнения;
3. Выполняет пооперационную обработку и выполнение программ;

Названия:

1. Инструментальная среда пользователя
2. Интерпретатор
3. Компилятор

Правильный ответ: a) 1; b) 3; c) 2.

1. Выберите процессы жизненного цикла соответственно их описанию.
2. Основной задачей этого этапа является определение подобных спецификаций разрабатываемого программного обеспечения.
3. Предоставляет собой процесс поэтапного написания кодов программы на выбранном языке программирования (кодирование), их тестирование и отладку.
4. Это процесс создания и внедрения новых версий программного продукта.

Варианты ответов: Проектирование, сопровождение, реализация.

Правильный ответ: 1) Проектирование; 2) Реализация;

3)Сопровождение.

1. На сколько процентов использование CASE-технологий повышает трудозатраты этапа разработки анализ в отличие от использования структурного подхода?

Правильный ответ: 10%.

1. Сколько этапов существует в каскадной схеме разработки ПО?

Правильный ответ: 5.

1. Разработка программного обеспечения с использованием CASE-средства предполагает
2. Автоматическая генерация кода;
3. Ручное документирование;
4. Автоматический контроль проекта;
5. Основные силы на анализ и проектирование;
6. Тестирование кодов;

Правильный ответ: 1), 3), 4).

# ПРИЛОЖЕНИЕ B

DLL

using System;

using System.Collections.Generic;

using System.Data.OleDb;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

using iTextSharp;

using iTextSharp.text;

using iTextSharp.text.pdf;

namespace DLL

{

public class Class1

{

//создание массива и счетчика, в который будет добавлятся правильные и неправильные решения

public static int n = 0;

public static string[] mass = new string[16];

public static int[] massNum = new int[16];

//система авторизации

public static void Avtoriz(Form A, Form B, TextBox t1, TextBox t2)

{

string Log = t1.Text;

string Pass = t2.Text;

var p = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Users1.mdb");

p.Open();

var c = new OleDbCommand("SELECT [Логин], [Пароль] FROM [users]", p);

OleDbDataReader reader = c.ExecuteReader();

bool f = true;

while (reader.Read() & (f))

{

if ((Log == reader[0].ToString()) & (Pass == reader[1].ToString()))

{

MessageBox.Show("Вы авторизованы!", "Авторизация", MessageBoxButtons.OK, MessageBoxIcon.Information);

f = false;

A.Hide();

B.Show();

}

}

reader.Close();

if (f)

{

MessageBox.Show("Неправильный логин или пароль!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//пропуск вопроса

public static void Skip\_question(int numb\_q)

{

mass[numb\_q] = "Неверно";

massNum[numb\_q] = 0;

}

//проверка 1 вопроса с элементом CheckBox

public static int Vopros1(CheckBox CB1, CheckBox CB2, CheckBox CB3, CheckBox CB4)

{

if ((!CB1.Checked) && (!CB2.Checked) && (!CB3.Checked) && (!CB4.Checked))

{

MessageBox.Show("Вы не выбрали ни один из вариантов", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if ((!CB1.Checked) && (!CB2.Checked) && (CB3.Checked) && (!CB4.Checked))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[0] = "Верно";

massNum[0] = 1;

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[0] = "Неверно";

massNum[0] = 0;

return 0;

}

}

//проверка 9 вопроса с элементом CheckBox

public static int Vopros9(CheckBox CB1, CheckBox CB2, CheckBox CB3, CheckBox CB4)

{

if ((!CB1.Checked) && (!CB2.Checked) && (!CB3.Checked) && (!CB4.Checked))

{

MessageBox.Show("Вы не выбрали ни один из вариантов", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if ((CB1.Checked) && (!CB2.Checked) && (!CB3.Checked) && (!CB4.Checked))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[8] = "Верно";

massNum[8] = 1;

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[8] = "Неверно";

massNum[8] = 0;

return 0;

}

}

//проверка 2 вопроса с элементом TextBox

public static int Vopros2(string right\_ans\_1, string right\_ans\_2\_1, string right\_ans\_2\_2, string ans\_1, string ans\_2)

{

if ((ans\_1 == "") || (ans\_2 == ""))

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if ((ans\_1 == right\_ans\_1) && ((ans\_2 == right\_ans\_2\_1)|| (ans\_2 == right\_ans\_2\_2)))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[1] = "Верно";

massNum[1] = 1;

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[1] = "Неверно";

massNum[1] = 0;

return 0;

}

}

//проверка 10 вопроса с элементом TextBox

public static int Vopros10(string right\_ans\_1\_1, string right\_ans\_1\_2, string right\_ans\_2, string ans\_1, string ans\_2)

{

if ((ans\_1 == "") || (ans\_2 == ""))

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if (((ans\_1 == right\_ans\_1\_1) || (ans\_1 == right\_ans\_1\_2)) && (ans\_2 == right\_ans\_2))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[9] = "Верно";

massNum[9] = 1;

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[9] = "Неверно";

massNum[9] = 0;

return 0;

}

}

//проверка 3 вопроса с элементом RadioButton

public static void Vopros3(RadioButton RB1, RadioButton RB2, RadioButton RB3, RadioButton RB4)

{

if ((!RB1.Checked) && (!RB2.Checked) && (RB3.Checked) && (!RB4.Checked))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[2] = "Верно";

massNum[2] = 1;

}

else

{

MessageBox.Show("Вы ответили неправильно", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[2] = "Неверно";

massNum[2] = 0;

}

}

//проверка 11 вопроса с элементом RadioButton

public static void Vopros11(RadioButton RB1, RadioButton RB2, RadioButton RB3, RadioButton RB4)

{

if ((!RB1.Checked) && (!RB2.Checked) && (!RB3.Checked) && (RB4.Checked))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[10] = "Верно";

massNum[10] = 1;

}

else

{

MessageBox.Show("Вы ответили неправильно", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[10] = "Неверно";

massNum[10] = 0;

}

}

//перемещение вариантов ответа из элемента listBox1 в другие

public static void chooseListBox(ListBox LB1, ListBox LB2)

{

if (LB1.SelectedIndex == 0)

{

LB2.Items.Add(LB1.Text);

LB2.Text = Convert.ToString(LB1.SelectedIndex);

LB1.Items.RemoveAt(0);

}

if (LB1.SelectedIndex == 1)

{

LB2.Items.Add(LB1.Text);

LB2.Text = Convert.ToString(LB1.SelectedIndex);

LB1.Items.RemoveAt(1);

}

if (LB1.SelectedIndex == 2)

{

LB2.Items.Add(LB1.Text);

LB2.Text = Convert.ToString(LB1.SelectedIndex);

LB1.Items.RemoveAt(2);

}

}

//проверка 4 вопроса с элементом ListBox

public static int Vopros4(ListBox LB1, ListBox LB2, ListBox LB3)

{

if ((LB1.Text == "") || (LB2.Text == "") || (LB3.Text == ""))

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if ((LB1.Items[0] == "c) Теоретическая и практическая деятельность, связанная с созданием программ;") &&

(LB2.Items[0] == "b) Программная реализация задачи, решенная на компьютере;") && (LB3.Items[0] == "a) Проблема, подлежащая решению;"))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[3] = "Верно";

massNum[3] = 1;

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[3] = "Неверно";

massNum[3] = 0;

return 0;

}

}

//проверка 12 вопроса с элементом ListBox

public static int Vopros12(ListBox LB1, ListBox LB2, ListBox LB3)

{

if ((LB1.Text == "") || (LB2.Text == "") || (LB3.Text == ""))

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if ((LB1.Items[0] == "a) Специальные средства, встроенные в пакеты прикладных программ;") && (LB2.Items[0] == "c) Выполняет пооперационную обработку и выполнение программы;") && (LB3.Items[0] == "b) Транслирует всю программу без ее выполнения;"))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[11] = "Верно";

massNum[11] = 1;

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[11] = "Неверно";

massNum[11] = 0;

return 0;

}

}

//проверка 5 вопроса с элементом ComboBox

public static int Vopros5(ComboBox CB1, ComboBox CB2, ComboBox CB3)

{

if ((CB1.SelectedIndex == -1) || (CB2.SelectedIndex == -1) || (CB3.SelectedIndex == -1))

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if ((CB1.SelectedIndex == 1) && (CB2.SelectedIndex == 2) && (CB3.SelectedIndex == 0))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[4] = "Верно";

massNum[4] = 1;

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[4] = "Неверно";

massNum[4] = 0;

return 0;

}

}

//проверка 13 вопроса с элементом ComboBox

public static int Vopros13(ComboBox CB1, ComboBox CB2, ComboBox CB3)

{

if ((CB1.SelectedIndex == -1) || (CB2.SelectedIndex == -1) || (CB3.SelectedIndex == -1))

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else if ((CB1.SelectedIndex == 0) && (CB2.SelectedIndex == 2) && (CB3.SelectedIndex == 1))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[12] = "Верно";

massNum[12] = 1;

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[12] = "Неверно";

massNum[12] = 0;

return 0;

}

}

//проверка 6 вопроса с элементом HScrollBar

public static void Vopros6(HScrollBar SB)

{

if (SB.Value == 30)

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[5] = "Верно";

massNum[5] = 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[5] = "Неверно";

massNum[5] = 0;

}

}

//проверка 14 вопроса с элементом HScrollBar

public static void Vopros14(HScrollBar SB)

{

if (SB.Value == 10)

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[13] = "Верно";

massNum[13] = 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[13] = "Неверно";

massNum[13] = 0;

}

}

//проверка 7 вопроса с элементом TrackBar

public static void Vopros7(TrackBar TB)

{

if ((TB.Value == 2) || (TB.Value == 3) || (TB.Value == 4))

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[6] = "Верно";

massNum[6] = 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[6] = "Неверно";

massNum[6] = 0;

}

}

//проверка 15 вопроса с элементом TrackBar

public static void Vopros15(TrackBar TB)

{

if (TB.Value == 1)

{

MessageBox.Show("Вы ответили правильно", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[14] = "Верно";

massNum[14] = 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[14] = "Неверно";

massNum[14] = 0;

}

}

//проверка 8 вопроса с элементом CheckedListBox

public static int Vopros8(CheckedListBox CLB)

{

if (CLB.CheckedItems.Count == 0)

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else

{

int count = 0;

foreach(string index in CLB.CheckedItems)

//подсчитываем сколько выбрано правильных ответов

{

if ((index == "Машинные языки;") || (index == "Процедурно-ориентированные языки;") || (index == "Машинно-оринтированные языки;") || (index == "Проблемно-ориентированные языки;"))

count += 1;

//если ответ неверный то счетчик уменьшается на 1

else if (index == "Компьютерные языки;")

count -= 1;

}

//сравниваем количество выбранных ответов с правильным

if (count == 4)

{

MessageBox.Show("Вы ответили правильно ", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[7] = "Верно";

massNum[7] = 1;

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[7] = "Неверно";

massNum[7] = 0;

return 0;

}

}

}

//проверка 16 вопроса с элементом CheckedListBox

public static int Vopros16(CheckedListBox CLB)

{

if (CLB.CheckedItems.Count == 0)

{

MessageBox.Show("Вы не ответили на вопрос", "Нет ответа", MessageBoxButtons.OK, MessageBoxIcon.Information);

return -1;

}

else

{

int count = 0;

foreach (string index in CLB.CheckedItems)

{

//подсчитываем сколько выбрано правильных ответов

if ((index == "Автоматическая генерация кода;") || (index == "Автоматический контроль проекта;") || (index == "Основные силы на анализ и проектирование;"))

count += 1;

//если ответ неверный то счетчик уменьшается на 1

else if ((index == "Ручное документирование;") || (index == "Тестирование кодов;"))

count -= 1;

}

//сравниваем количество выбранных ответов с правильным

if (count == 3)

{

MessageBox.Show("Вы ответили правильно ", "Правильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

n = n + 1;

mass[15] = "Верно";

massNum[15] = 1;

return 1;

}

else

{

MessageBox.Show("Вы ответили неправильно ", "Неправильный ответ", MessageBoxButtons.OK, MessageBoxIcon.Information);

mass[15] = "Неверно";

massNum[15] = 0;

return 0;

}

}

}

//вывод результатов теста в DataGridView в строчку

public static void Vivod\_DGV(DataGridView grid)

{

grid.ColumnCount = mass.Length;

grid.RowCount = 2;

for (int i = 0; i<mass.Length; i++)

{

grid.Rows[0].Cells[i].Value = "[" + (i+1) + "]";

grid.Rows[1].Cells[i].Value = mass[i];

}

}

//передает первый элемент массива в Vivid\_DGV

public static void Result\_DGV(DataGridView DGV)

{

for (int i = 0; i<mass.Length; i++)

{

Vivod\_DGV(DGV);

}

}

//вывод результатов теста в DataGridView в ряд

public static void Result\_DGV\_ROW(DataGridView DGV)

{

for (int i = 0; i<massNum.Length;i++)

{

DGV.Rows.Add(i + 1, massNum[i]);

}

}

//создание диаграммы

public static void Diagram(Chart C, TextBox TB1, TextBox TB2)

{

C.Series[0].Points.AddXY(1, n);

C.Series[1].Points.AddXY(2, Math.Abs(n-16));

Vivod(TB1, n);

Vivod(TB2, 16-n);

}

//конвертирует в строковый тип для заполнения TextBox в диаграмме

public static void Vivod(TextBox t, double c)

{

t.Text = Convert.ToString(c);

}

//запись в блокнот

public static void ZapisBloknot()

{

StreamWriter rez = File.CreateText("Исходный массив.txt");

for (int i = 0; i < massNum.Length; i++)

{

rez.WriteLine((i+1).ToString() + " - " + massNum[i]);

}

rez.Close();

System.Diagnostics.Process.Start("Исходный массив.txt");

}

//после удаления элемента записывает получившийся массив в DataGridView в ряд

public static void New\_Mas(int[] rezmas, DataGridView DGV)

{

for (int i = 0; i<rezmas.Length; i++)

{

DGV.Rows.Add(i+1, rezmas[i]);

}

}

//удаление выбранного элемента и запись получившегося массива

public static void Del\_El(ref int[] mas, int a)

{

int max = mas.GetLength(0);

for (int i = 0; i< mas.GetLength(0); i++)

{

if (i == a)

max = i;

}

for (int i = max; i < mas.GetLength(0) - 1; i++)

mas[i] = mas[i + 1];

Array.Resize(ref mas, mas.GetLength(0)-1);

}

//создание массива для удаления элементов

public static void Get\_Delmas(ref int[] mas)

{

for (int i = 0; i < massNum.Length; i++)

{

mas[i] = massNum[i];

}

}

//сортировка простым выбором

public static void Sort\_Prost(ref int[] mas)

{

//запись элементов массива с результатами в новый массив, для того чтобы при удалении элементов, не сбивался порядок

for (int i = 0; i<massNum.Length;i++)

{

mas[i] = massNum[i];

}

for (int i = 0; i<mas.Length-1; i++)

{

int k = i;

int x = mas[i];

for (int j=i+1; j<mas.Length; j++)

{

if (mas[j] < x)

{

k = j;

x = mas[j];

}

}

mas[k] = mas[i];

mas[i] = x;

}

}

//запись результатов в pdf файл

public static void Add\_pdf()

{

var document = new Document();

var zap = PdfWriter.GetInstance(document, new System.IO.FileStream("ZapPDF.pdf", System.IO.FileMode.Create));

document.Open();

var Shrift = BaseFont.CreateFont(@"C:\WINDOWS\Fonts\times.ttf", "CP1251", BaseFont.EMBEDDED);

var font = new Font(Shrift, 10, Font.NORMAL, BaseColor.BLUE);

var tabl = new PdfPTable(1);

var cell = new PdfPCell();

cell.HorizontalAlignment = Element.ALIGN\_LEFT;

cell.Colspan = 2;

cell.Border = 0;

cell.FixedHeight = 16.0F;

cell.Phrase = new Phrase("Исходный массив", font);

tabl.AddCell(cell);

cell.BackgroundColor = BaseColor.WHITE;

cell.Colspan = 1;

cell.Border = 15;

for (int i = 0; i< massNum.Length; i++)

{

cell.Phrase = new Phrase((i+1).ToString() + " - " + massNum[i].ToString(), font);

tabl.AddCell(cell);

}

tabl.TotalWidth = document.PageSize.Width - 500;

tabl.WriteSelectedRows(0, -1, 150, 750, zap.DirectContent);

document.Close();

zap.Close();

System.Diagnostics.Process.Start("msedge.exe", System.IO.Directory.GetCurrentDirectory() + @"\ZapPDF.pdf");

}

//сброс всех результатов теста

public static void BackToStart()

{

n = 0;

mass = new string[16];

massNum = new int[16];

}

}

}