МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ
КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им.И.АРАБАЕВА
ОСПО ИНСТИТУТА НОВЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

«УТВЕРЖДАЮ»
Директор ИНИТ
КГУ им.И.Арабаева
к.т.н., и.о.доц. У.Керимов
_____
«__» _____ 2023 г.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

по дисциплине Системное и прикладное программирование

для студентов специальности 230701 - «Прикладная информатика (по отраслям)»
гр.ПИ-1[illegible]23

форма обучения Очное

Курс 1 **Семестр** 1,2

Часов: всего: 90, лекций:54, практ. занятий:36

СРС 60

Учебно-методический комплекс разработал(а): преподователь Масиянова А.К.

Рассмотрена и утверждена на заседании ОСПО ИНИТ КГУ им.И. Арабаева

Протокол № 1 **от «** 08 **»** 09 **20** 23 **г.**

Зав. ОСПО ИНИТ: И.С.Сейтказиева _____

Одобрено учебно-методическим советом ИНИТ КГУ им.И. Арабаева
Протокол № 1 **от «** 08 **»** 09 **20** 23 **г.**

Председатель УМС: _____

Бишкек 2023г.

Содержание

1. Цели освоения дисциплины

Курс преследует следующие цели: Ознакомление со структурой ЭВМ, ознакомление со структурой операционных систем и их сервисов на примерах MS-DOS, Windows и Linux/UNIX. Знания, полученные в рамках этого курса, могут быть применены студентами в широком спектре их дальнейшей профессиональной деятельности, например при создании автоматизированных систем обработки данных.

2. Компетенции обучающегося, формируемые в результате освоения дисциплины /модуля

В результате освоения дисциплины формируются следующие компетенции:

В результате освоения дисциплины студент:

1**. должен знать:**

- понимать принципы работы ЭВМ, принципы работы операционных систем, принципы взаимодействия программ и операционной системы.

2. должен уметь:

- обладать теоретическими знаниями о:

устройстве процессоров i80x86, методах адресации, системе команд процессоров i80x86, способах программирования на языке ассемблера в среде операционной системы MS-DOS и совместимых с ней ОС.

3. должен владеть:

- ориентироваться в вопросах структуры системного программного обеспечения, структуре операционных систем, в разновидностях и основных особенностях операционных систем MS-DOS, Windows и Linux/UNIX;

приобрести навыки составления программ на языке ассемблера, использующих сервисы операционной системы MS-DOS и совместимых с ней ОС

4. Структура и содержание дисциплины/ модуля

Общая трудоемкость дисциплины составляет - 90 часа(ов).
Форма промежуточного контроля дисциплины экзамен в 2 семестре.
Суммарно по дисциплине можно получить 100 баллов,
из них текущая работа оценивается в 50 баллов,
итоговая форма контроля - в 50 баллов.
Минимальное количество для допуска к зачету 28 баллов.
86 баллов и более - **"отлично"** (отл.);
71-85 баллов - **"хорошо"** (хор.);
55-70 баллов - **"удовлетворительно"**
(удов.); 54 балла и менее -
"неудовлетворительно" (неуд.).

4.1 Структура и содержание аудиторной работы по дисциплине

I полугодие

№	Раздел дисциплины	Семестр	Неделя семестра	Количество часов	
				Лек	Прак
1	Основные понятия о программировании. Виды программного обеспечения	1	1	2	
2	Системное программное обеспечение. Виды системных программ	1	2	2	
3	Операционные системы. Виды ОС. Развитие и различие ОС. Краткая характеристика ОС WINDOWS	1	3	4	2
4	Основные возможности ОС. Утилиты. Вспомогательные программы	1	4	4	2
5	Базовое программное обеспечение. Операционные оболочки	1	5	4	2
6	Дополнительные устройства ПК. Драйвера. Установка периферийных устройств	1	6	4	4
7	Память. Организация памяти IBM/PC/XT/AT Адресация памяти. Сегмент и смещение.	1	7	4	4
8	Команды пересылки данных. Прямые косвенные адресации. Адресация команды	1	8	2	2
9	Инструментальное программное обеспечение. Языки программирования	1	9	2	2
10	Языки программирования высокого уровня. Этапы подготовки программы.	1	10	2	2
11	Разработка программы. Интерпретация.	1	11	2	2
Итого за I семестр:				**32**	**22**

II полугодие

№	Раздел дисциплины	Семестр	Неделя семестра	Количество часов	
				Лек	Прак
1	Архитектура системных программ Архитектура программного обеспечения	2	1	2	
2	Прикладное программное обеспечение. Виды прикладных программ	2	2	2	
3	Прикладное программное обеспечение профессионального обеспечения	2	3	2	
4	Классификация ПО. Инструментальные программные средства общего назначения	2	4	2	
5	Текстовые редакторы и издательские системы	2	5	2	2
6	Электронные таблицы. Системы Управления Базами Данных	2	6	2	2
7	Обзор системного программного обеспечения	2	7	2	2
8	Тенденции развития системного программирования. Задачи системного программирования	2	8	2	2
9	Функции файловой системы. Функции системного назначения	2	9	2	2
10	Современные возможности системного программирования.	2	10	2	2
11	Искусственный интеллект. Использование ИИ в системном программировании.	2	11	2	2
Итого за II семестр:				**22**	**14**
Всего за учебный год:				**54**	**36**

4.2 Содержание дисциплины

Тема 1. Основные понятия о программировании. Виды программного обеспечения
Тема 2. Системное программное обеспечение. Виды системных программ
Тема 3. Операционные системы. Виды ОС. Развитие и различие ОС. Краткая характеристика ОС WINDOWS
Тема 4. Основные возможности ОС. Утилиты. Вспомогательные программы
Тема 5. Базовое программное обеспечение. Операционные оболочки
Тема 6. Дополнительные устройства ПК. Драйвера. Установка периферийных устройств
Тема 7. Память. Организация памяти IBM/PC/XT/AT Адресация памяти. Сегмент и смещение.
Тема 8. Команды пересылки данных. Прямые косвенные адресации. Адресация команды
Тема 9. Инструментальное программное обеспечение. Языки программирования
Тема 10. Языки программирования высокого уровня. Этапы подготовки программы.
Тема 11. Разработка программы. Интерпретация.
Тема 12 Архитектура системных программ Архитектура программного обеспечения
Тема 13 Прикладное программное обеспечение. Виды прикладных программ
Тема 14 Прикладное программное обеспечение профессионального обеспечения
Тема 15 Классификация ПО. Инструментальные программные средства общего назначения
Тема 16 Текстовые редакторы и издательские системы
Тема 17 Электронные таблицы. Системы Управления Базами Данных
Тема 18 Обзор системного программного обеспечения
Тема 19 Тенденции развития системного программирования. Задачи системного программирования
Тема 20 Функции файловой системы. Функции системного назначения
Тема 21 Современные возможности системного программирования.
Тема 22 Искусственный интеллект. Использование ИИ в системном программировании.

Темы для самостоятельной работы студента:

1. Основной объект интерфейса: окно и его основные части.
2. Диалоговое окно и стандартные элементы управления, предназначенные для ввода информации и управления работой программы.
3. Визуализация научных и инженерных данных
4. Шаблоны функций.
5. Шаблоны классов.
6. Наследование.
7. Виртуальные функции и абстрактные базовые классы.
8. Множественное наследование.
9. Интерфейс пользователя. Основные понятия.
10. Стандартизация пользовательского интерфейса.
11. Интерфейс командной строки.
12. Текстовый интерфейс.
13. Оконный интерфейс.
14. Графический оконный интерфейс.
15. Web-интерфейс.
16. Современный графический пользовательский интерфей

Краткое содержание курса

Введение

Прикладные программы – это приложения, которые необходимы пользователям для решения конкретных задач. Они делятся на два вида – программы общего назначения и программы специального назначения. Например, текстовый редактор относится к программам общего назначения, а программа для бухгалтерского учета на предприятии – к специальному виду. Но так или иначе, их функционирование невозможно без системных программ. К системным в первую очередь относится операционная система. Прикладные программы обращаются к ней через системные вызовы. Даже в самых простых задачах необходимо реализовывать открытие и сохранение файла, обращаться к функциям вставки даты и времени.

Графический пользовательский интерфейс – это взаимодействие между приложением и пользователем посредством произвольного доступа к объектам с помощью устройств ввода. Иными словами, кнопки, диалоговые окна, поля ввода информации, расположенные в окне любой программы.

Форма – это основной элемент приложения с графическим интерфейсом. Свойства формы (или окна приложения) зависят от использованной системы программирования, но большинство стандартных свойств должны поддерживать все среды. К форме можно применять различные методы, т.е. функции, которые можно выполнить после написания соответствующего кода. Условно формой называют окно во время его редактирования, т.е. когда приложение еще не сделано, то оно имеет форму, а когда оно уже готово и выполняется, то оно содержит окно.

C#

Курс лекций содержит теоретические сведения раздела «Реализация программных модулей с использованием принципов технологии объектно-ориентированного программирования» МДК 01.02 «Прикладное программирование» ПМ 01. «Разработка программных модулей программного обеспечения для компьютерных систем», для специальности 230115 «Программирование в компьютерных системах».

Цели и задачи курса лекций:

- изучение объектно-ориентированного языка программирования Visual C#;
- приобретение практических навыков программирования задач и отладки программ на языке Visual C#.

Рассматриваемый курс содержит: 19 тем; список литературы.

Раздел «Реализация программных модулей с использованием принципов технологии объектно-ориентированного программирования» имеет практическую направленность, так как знакомит студентов с разработкой современных приложений в различных отраслях промышленности.

Целью преподавания раздела МДК 01.02 является освоение студентами новых принципов программирования, основанных на объектно-ориентированной модели. Приобретение знаний и умений по установке, настройке, поддержке и сопровождению программного обеспечения. Овладение студентами теоретическими знаниями, необходимыми для проектирования и программирования современных приложений.

Задачи изучения раздела - научить студентов создавать программы на основе новой концепции объектно-ориентированного программирования, познакомить студентов с различными новыми технологиями разработки современного

программного обеспечения. В результате изучения курса студенты должны уметь разрабатывать различные приложения с помощью языка программирования C#

Тема «Принципы технологии объектно-ориентированного программирования»

При изучении объектно-ориентированного программирования (ООП) наибольшей проблемой является использование новой терминологии и понимание нового подхода к решению старых задач - новой *технологии* программирования. Определения новых терминов и характеристики *методов* программирования составляют содержание данной темы.

Как в любом виде деятельности в программировании имеется своя *технология*: - это знания, правила, навыки и инструменты, позволяющие получать гарантированный качественный результат. Но само по себе соблюдение ряда правил не дает гарантию качества результата. Это объясняется спецификой программирования. Во-первых, это не наука, где знание какой-либо формулы позволяет однозначно решить задачу, подставив

- нее исходные данные и получив результат. Во-вторых, эти правила необходимо соблюдать не столько на бумаге, сколько в голове. То есть технология программирования - это скорее способ организации процесса обдумывания программы, нежели ее записи. Из сказанного следует, что если пишущий программу - мыслит, то он уже придерживается какой-то технологии программирования, даже не подозревая об этом. Простейший метод заключается в написании программы сразу от начала до конца, без использования каких-либо общих принципов, то есть "как бог на душу положит".

Рассмотрим наиболее известные из технологий:

- метод "северо-западного" угла (имеется в виду лист бумаги или экран дисплея). Программа пишется сразу от начала до конца, без использования каких-либо общих принципов;
- технология структурного программирования, в ней предполагается придерживаться принципов модульности, нисходящего и пошагового проектирования программ, одновременного проектирования программ и структур данных.
- технология объектного программирования: связана с использованием при проектировании программы понятий объектов и их классов.

Что первично: алгоритм (процедура, функция) или обрабатываемые им данные? В традиционной технологии программирования взаимоотношения процедуры - данные имеют более-менее свободный характер, причем процедуры (функции) являются ведущими в этой связке: как правило, функция вызывает функцию, передавая данные друг - другу по цепочке. Соответственно, технология структурного проектирования программ прежде всего уделяет внимание разработке алгоритма.

- технологии ООП взаимоотношения данных и алгоритма имеют более регулярный характер: во-первых, *класс* (базовое понятие этой технологии) объединяет в себе данные (структурированная переменная) и *методы* (функции). Во-вторых, схема взаимодействия функций и данных принципиально иная. Метод (функция), вызываемый для одного объекта, как правило, не вызывает другую функцию непосредственно. Для начала он должен иметь доступ к другому объекту (создать, получить указатель, использовать внутренний объект в текущем и т.д.), после чего он уже может вызвать для него один из известных методов. Таким образом, структура программы определяется взаимодействием объектов различных классов между собой. Как правило, имеет место иерархия классов, а технология ООП иначе может быть названа как программирование "от класса к классу"

.

Модульное программирование.

Принцип модульности формулируется как требование разработки программы в виде совокупности модулей (функций). При этом разделение на модули должно носить не механический характер, а исходить из логики программы:

- размер модуля должен быть ограничен;
- модуль должен выполнять логически целостное и завершенное действие;
- модуль должен быть универсальным, то есть по возможности параметризованным: все изменяемые характеристики выполняемого действия должны передаваться через параметры;
- входные параметры и результат модуля желательно передавать не через глобальные переменные, а через формальные параметры и результат функции.

Еще одной, но уже физической единицей программы является текстовый файл, содержащий некоторое количество функций и определений типов данных и переменных. Модульное программирование на уровне файлов - это возможность разделить полный текст программы на несколько файлов, транслировать их независимо друг от друга.

Принцип модульности распространяется не только на программы, но и на данные: любой набор параметров, характеризующих логический или физический объект, должен быть представлен в программе в виде единой структуры данных (структурированной переменной).

Олицетворением принципа модульности является библиотека стандартных функций. Она, как правило, обеспечивает полный набор параметризованных действий, используя общие структуры данных. Библиотеки представляют собой аналогичные Си-программы, независимо оттранслированные и помещенные в каталог библиотек.

Нисходящее программирование.

Нисходящее проектирование программы заключается в том, что разработка идет от общей неформальной формулировки некоторого действия

программы на естественном языке, "от общего к частному": к замене ее одной из трех формальных конструкций языка программирования:

- простой последовательности действий;
- конструкции выбора или оператора if;
- конструкции повторения или цикла.
- записи алгоритма это соответствует движению от внешней (объемлющей) конструкции к внутренней (вложенной). Эти конструкции также могут содержать в своих частях неформальное описание действий, то есть нисходящее проектирование по своей природе является пошаговым. Отметим основные свойства такого подхода:
- первоначально программа формулируется в виде некоторого неформального действия на естественном языке;
- первоначально определяются входные параметры и результат действия;
- очередной шаг детализации не меняет структуру программы, полученную на предыдущих шагах;
- если в процессе проектирования получаются идентичные действия в различных ветвях, то это означает необходимость оформления этого действия отдельной функцией;
- необходимые структуры данных проектируются одновременно с детализацией программы.
- результате проектирования получается программа, в которой принципиально отсутствует оператор перехода goto, поэтому такая технология называется "программирование без goto".

Пошаговое программирование.

Нисходящее проектирование по своей природе является пошаговым, ибо предполагает каждый раз замену одной словесной формулировки на единственную конструкцию языка. Но в процессе разработки программы могут быть и другие шаги, связанные с детализацией самой словесной формулировки в более подробную.

То, что этот принцип выделен отдельно, говорит о необходимости предотвратить соблазн детализации программы сразу от начала до конца и развивать умение выделять и сосредоточивать внимание на главных, а не второстепенных деталях алгоритма.

Вообще нисходящее пошаговое проектирование программы не дает гарантии получения "правильной" программы, но позволяет возвратиться при обнаружении тупиковой ситуации к одному из верхних шагов детализации.

Структурное программирование.

При нисходящей пошаговой детализации программы необходимые для работы структуры данных и переменные появляются по мере перехода от неформальных определений к конструкциям языка, то есть процессы

детализации алгоритма и данных идут параллельно. Однако это касается прежде всего отдельных локальных переменных и внутренних параметров. С самой же общей точки зрения предмет (в нашем случае - данные) всегда первичен по отношению к выполняемым с ним действиям (в нашем случае - алгоритм). Поэтому на самом деле способ организации данных в программе более существенно влияет на ее структуру алгоритма, чем что-либо другое, и процесс проектирования структур данных должен опережать процесс проектирования алгоритма их обработки.

Структурное программирование - модульное нисходящее пошаговое проектирование алгоритма и структур данных.

Тема «Абстрактные типы данных в C++ (ADT)»

Один из наиболее важных этапов разработки программ заключается в выборе эффективного способа представления данных. Во многих случаях недостаточно объявить простую переменную или массив. C# имеет встроенный тип **struct,** представляющий запись. Структура это особый случай класса, в котором все поля являются открытыми.

Пример

```
struct student {
int id;
char name [20]; };
struct student s= { 567, “Лопухин Олег”}
```

Что определяет тип? **Тип** определяет два типа информации: набор свойств и набор операций. Предположим, что нужно определить новый тип данных, для этого:

1. необходимо обеспечить способ хранения данных, возможно разработав структуру.
2. необходимо обеспечить способы манипулирования данными.

- информатике был разработан успешный способ определения новых типов данных. Это трехэтапный процесс перехода от абстрактного понятия к конкретной реализации:
- Создайте абстрактное описание свойств типа и операций, которые можно выполнять с данным типом. Это описание не должно быть связано с конкретной реализацией. Оно даже не должно быть связано с конкретным языком программирования. **Такое формальное абстрактное описание называется абстрактным типом данных**(ADT).
- Разработайте программный интерфейс, реализующий ADT, т.е. укажите, как следует хранить данные, и опишите набор функций, выполняющий требуемые операции.

- Создайте код для реализации интерфейса. Конечно, этот этап имеет большое значение, но программисту, который использует новый тип, не обязательно знать подробности реализации.

Рассмотримпрактическийслучайпредставленияданных.

Предположим, что нужно создать программу, которая позволяет вводить список всех фильмов, просмотренных в течение года. Для каждого фильма желательно записать наименование, рейтинг и т.п. Это предполагает использование структуры для каждого фильма. Для реализации этой задачи можно создать массив структур**.**

Существует лучший способ. Переопределим структуру так, чтобы каждая содержала указатель на следующую структуру. Тогда при создании каждой новой структуры ее адрес можно сохранять в предшествующей структуре. Структуру **film** необходимо задать следующим образом:

#define TSIZE 45 /* размер массива, предназначенного для хранения заголовка фильма*/ struct film**{**
char title [

Задача 1.1.

Постановка задачи. Рассмотреть алгоритм сортировки TimSort одномерного массива.

Внешнее описание: сначала указывается количество и тип элементов массива, затем исходный массив вводится в первый компонент DataGridView1, отсортированный массив выводится во второй компонент DataGridView2.

Функциональная спецификация: элементы массива можно вводить как вручную, так и с помощью датчика случайных чисел. Система программирования: Microsoft Visual C#.

Особенности реализации и возникшие трудности: Timsort – гибридный алгоритм стабильной сортировки, сочетающий сортировку вставками и сортировку слиянием. Предназначен для работы с большим количество различных данных. Основная идея алгоритма в том, что в реальном мире сортируемые массивы данных часто содержат в себе упорядоченные подмассивы. Основная идея алгоритма:

1) По специальному алгоритму входной массив разделяется на подмассивы.
2) Каждый подмассив упорядочивается сортировкой вставками.
3) Отсортированные подмассивы собираются в единый массив с помощью модифицированной сортировки слиянием.

Перед непосредственной реализацией алгоритма создаются интерфейс класса и дочерние классы TimSort, используя конструктор, для каждого типа данных – целочисленный и вещественный. Затем все методы алгоритма для каждого из двух типов данных (разделение на подмассивы, сортировка и слияние в массив) объединяются в базовый класс TimSort:

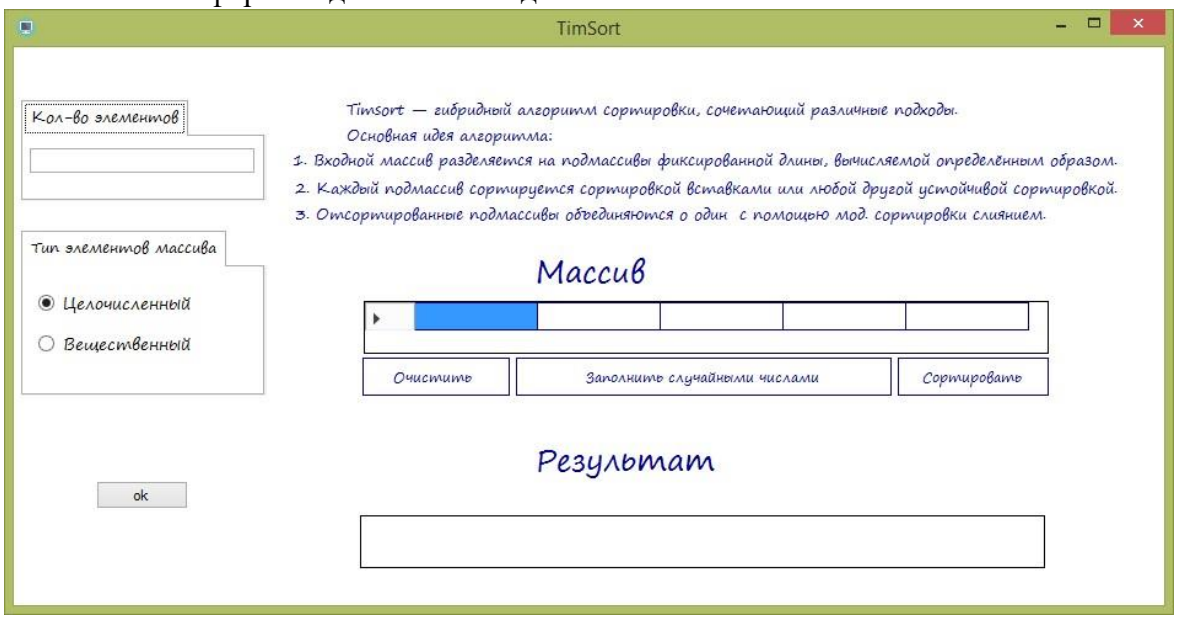
```
public class TimSort
    {       public TimsortInteger ts1;       public TimsortFlt ts2;       public int kol;
        public string type;       public TimSort()
      {
        ts1 = new TimsortInteger();          ts2 = new TimsortFlt();
        type = "int";
      }
```

```
    ~TimSort() { }
}
```

Достоинства приложения: выбор количества и типа элементов (целый, вещественный) массива.

Недостатки приложения: реализован только один алгоритм сортировки. Разработка интерфейса. В приложении использовались следующие компоненты: текстовое окно для вывода информации о сортировке (TextBox), таблицы для ввода элементов массива и вывода результата (DataGridView), кнопки для изменения настроек, запуска сортировки, очистки табличных полей, заполнения таблицы случайными числами (Button), статические поля для объяснений (Label), кнопки переключателей (RadioButton), панели для объединения компонентов.

Окончательно форма задачи имеет вид:



Дополнительные процедуры и функции, используемые в приложении: 1) void print_arr<T>(T[] a, int n) – процедура вывода элементов отсортированного массива в таблицу.

2) Обработчик кнопки «ОК»:

Запоминание количества элементов в массиве, запоминание выбора типа элементов (с помощью значения логической переменной), изменение количества столбцов таблицы в зависимости от заданного количества элементов. В этой процедуре происходит проверка на максимальный размер компонента DataGridView (не более 654) и на ошибочный ввод числа элементов.

3) Обработчик кнопки «Сортировать»:

```
if (timsort.type == "int")
        {
          int[] a = new int[timsort.kol];              for (int i = 0; i != timsort.kol; i++)
          {                  try                {
              a[i] = Convert.ToInt16(dataGridView1.Rows[0].Cells[i].Value);
            }
            catch (OverflowException)
            {
              MessageBox.Show("Переполнение типа", "Переполнение типа",
```

```
MessageBoxButtons.OK, MessageBoxIcon.Information);                    break;
                }
                        }
                timsort.ts1.timSort(ref a, timsort.kol);              print_arr(a, timsort.kol);
            }
```

В этом фрагменте реализуется сортировка целочисленного массива, аналогичный код для вещественного типа записывается в конструкции else. Массив для обработки берется из коллекции, используемой в соответствующем классе. Заполнение массива данными из сетки таблицы производится в блоке try/catch/break с проверкой значений на переполнение памяти. После этого вызывается метод для непосредственной сортировки и процедура вывода результата.

4) Обработчик кнопки «Заполнить случайными числами». Функция датчика случайных чисел меняется в зависимости от типа элементов. Для целых чисел:
dataGridView1.Rows[0].Cells[i].Value= r.Next(-50, 50);

Для вещественных чисел: dataGridView1.Rows[0].Cells[i].Value =Math.Round((-50.6 + r.NextDouble() * (50.6 -20.3)),3);

Метод Next() возвращает случайное целое число из заданного диапазона. Метод NextDouble() возвращает случайное вещественное число с заданным количеством знаков после запятой.

5) Обработчик кнопки «Очистить»:

```
for (int i = 0; i != timsort.kol; i++)
        {
          dataGridView1.Rows[0].Cells[i].Value ="";
        }
```

Очищает все ячейки таблицы в цикле.

Задача 1.2.

Постановка задачи. Рассмотреть два метода сортировки одномерного массива.

Внешнее описание. Записать элементы массива в текстовые поля, нажать соответствующую алгоритму сортировки кнопку, выдать результат в статическое поле в виде текста.

Функциональная спецификация. Для упрощения работы зададим заранее количество элементов массива, равное пяти. Соответственно, создадим пять текстовых полей, размещённых с помощью метода grid() (сетка).

Система программирования. Python.

Особенности реализации и возникшие трудности: в стандартной библиотеке Tkinter() нет виджета, отвечающего за создание таблиц, поэтому при решении задачи можно использовать два метода: 1) Изменить библиотеку на wxPython, Qt и др.

2) Использовать виджеты Entry, выровняв их по сетке с помощью цикла.

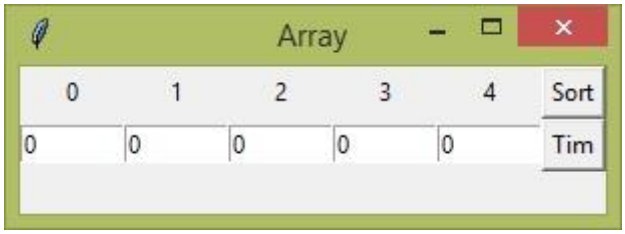
Для решения задачи воспользуемся вторым способом.

По кнопке **«Sort»** реализуется алгоритм сортировки простым выбором. По кнопке «Tim» - алгоритм TimSort, рассмотренный в предыдущей задаче. В отличие от C# и многих других систем программирования, в Python TimSort является стандартным методом сортировки, поэтому он вызывается одной командой **a.Sort()** или **Sorted(a).**

В отдельной функции реализуется формирование значения текстовой переменной, состоящей из элементов отсортированного массива (списка).

Достоинства приложения: простота реализации.

Недостатки приложения: пользователем нельзя изменить размер массива. Разработка интерфейса. В приложении использовались следующие виджеты: статический текст для вывода индексов элементов массива и для вывода результата (Label), пять текстовых полей для ввода элементов, выровненных по сетке (Entry), кнопки для сортировки по выбранному алгоритму (Button). Окончательно форма задачи имеет вид:



Классы, процедуры и функции, используемые в приложении:

1) Основной класс App(Frame), в котором перечислены функции, влияющие на формирование графического окна приложения.
2) get_values(root) – получение числового значения из текстового поля.
3) sort (root, array1) – алгоритм сортировки методом простого выбора.
4) ts(root,array1) – встроенный алгоритм сортировки TimSort.
5) calc_CR(root) и calc_ts(root) – ввод данных в функцию сортировки и вывод значения результата в текстовую переменную для каждого метода соответственно.
6) create_widgets(root) – создание элементов управления на экране.
7) run(root) – функция для запуска приложения.

Задача 1.3

Постановка задачи. Кодирование сигнала в цифровых системах связи с помощью натурального и симметричного кода.

Внешнее описание: Ввести напряжение ограничения, кодируемые значения, число уровней квантования. Вывести двоичные коды значений в зависимости от выбора алгоритма кодирования.

Функциональная спецификация: расчет производится в двух соответствующих процедурах — обработчиков кнопок. Т. к. коды представляют собой двоичные числа, то в программе производится перевод числа из десятичной системы счисления в двоичную. Система программирования: Microsoft Visual Basic

Особенности реализации и возникшие трудности: перевод числа из десятичной системы в двоичную осуществляется по правилу последовательного деления числа на 2 и сбор остатков с конца. Остатки записываются в числовой массив. Этот алгоритм реализован с помощью цикла:

```
For i = 1 To m          a1(i) = u1 Mod 2          u1 = u1 \ 2       Next
```

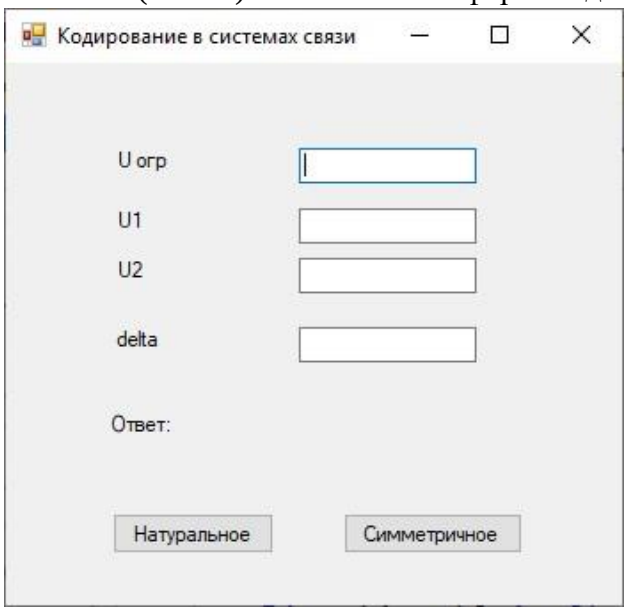
Сбор остатков происходит конвертацией элементов массива в строку в обратном цикле с отрицательным шагом:

```
 For i = m To 1 Step -1
        ot1 = ot1 + Str(a1(i))
    Next
```

В формуле подсчета числа импульсов в кодовой комбинации используется функция округления в сторону большего целого. В VB встроенная в библиотеку Math функция Round производит округление по правилам математики, поэтому к аргументу этой функции надо прибавить 0,5, чтобы округление всегда велось в большую сторону.

Достоинства приложения: выполнение всех необходимых функций.

Недостатки приложения: не учитываются неверные значения. Разработка интерфейса. В приложении использовались следующие компоненты: 4 текстовых поля для ввода исходных данных (TextBox), статические поля для пояснений и вывода результата (Label), 2 кнопки для выполнения соответствующего расчета (Button). Окончательно форма задачи имеет вид:



Задача 1.4.

Постановка задачи. Подсчитать количество теплоты, выделяемой в проводнике при протекании по нему тока. Возьмем соответствующую формулу: $Q=U^2t/R$, где Q — количество теплоты, Дж; U — напряжение, В; t — время, с; R — сопротивление, Ом. При этом R=pl/s, где p — удельное сопротивление материала проводника, Ом•м; l — длина проводника, м; s — площадь поперечного сечения проводника, $см^2$.

Внешнее описание: Ввести необходимые исходные данные. Результат вывести на форму и сохранить в текстовом файле.

Функциональная спецификация: после ввода значений в текстовые поля результат автоматически появляется в последнем текстовом поле, благодаря обработчику события TextBox_TextChanged. Полученный результат можно вставить в текстовый документ с помощью соответствующей кнопки. Система программирования: Microsoft Visual Basic

Особенности реализации и возникшие трудности: непосредственный расчет производится в отдельной процедуре solve(), которая вызывается из каждого обработчика TextBox_TextChanged. Перед вычислением производится проверка на соответствие данному из поля числовому значению. Обработчик кнопки «Выход» очищает поля ввода и вывода и выводит диалоговое окно закрытия программы.

Кнопка «Вставить в документ» активна только в том случае, когда в поле результата появится значение.

После основного расчета в обработчике этой кнопки формируется текстовая переменная, включающая значения из текстовых полей. Для записи информации в текстовый файл в VB существует метод StreamWriter пространства имен System.IO. Блок вывода в файл имеет вид:

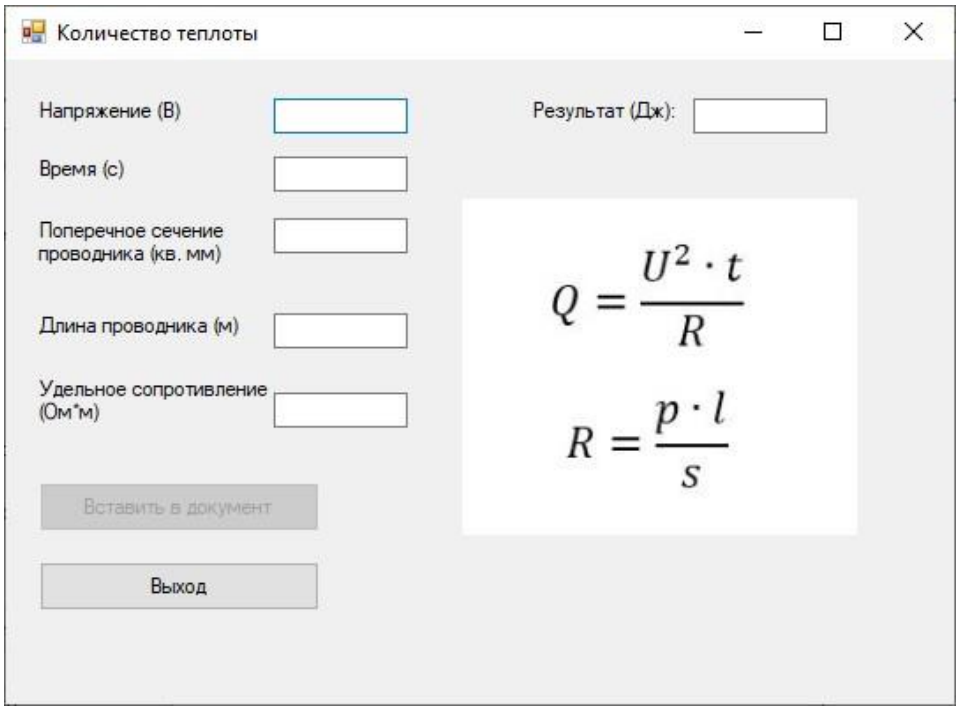
```
Dim fw As System.IO.StreamWriter
    Try
      fw = New System.IO.StreamWriter("Test1.txt", False)          fw.WriteLine(s)
    Finally
      Process.Start("Test1.txt", "Notepad.exe")          fw.Close()       End Try
```

Логический параметр false означает, что файл можно перезаписывать. После успешного сохранения файл должен открыться в стандартном приложении Блокнот.

Достоинства приложения: отсутствие лишних кнопок для выполнения расчета.

Недостатки приложения: перегруженность полями ввода. Разработка интерфейса. В приложении использовались следующие компоненты: текстовые поля для ввода исходных данных и вывода результата (TextBox), 2 кнопки для сохранения результата в файл и выхода из приложения (Button), графический элемент для визуализации формул (PictureBox), поясняющий текст (Label).

Окончательно форма задачи имеет вид:



Глава 2. Работа с текстовой информацией.

Одной из главных повседневных задач пользователя компьютера является работа с текстовой информацией. Доклады и отчеты, заметки и книги, статьи и документы… Порой приходится в потоке символов буквы отделять от чисел, на ходу придумывать имена и пароли, считать количество набранных

символов, добавлять дату работы с документом и многое другое. Для этих целей используются разнообразные программные средства.

Задача 2.1

Постановка задачи. Генератор новых имен путем перестановки букв из текстового файла.

Внешнее описание: к программе подгружаются 2 файла с мужскими и женскими именами. По щелчку на соответствующей кнопке формируется новое имя с помощью определенного алгоритма.

Функциональная спецификация: алгоритм для каждого имени (мужское и женское) реализуется с помощью обработчика кнопки.

Женские имена: если последняя буква имени в текстовом файле «-а» или «-я», то ее отбрасываем. Оставшееся слово переворачиваем. Если теперь последняя буква слова – гласная, то добавляем окончание «-я», если согласная, то окончание «-а».

Например, имя «Галина» превратится в «Нилага», а «Елена» в «Нелея». Мужские имена: если последняя буква имени в текстовом файле «-й», то отбрасываем эту букву и предпоследнюю. Остаток переворачиваем. Если теперь последняя буква является гласной, то к концу слова добавляем «-й».

Например, «Сергей» превратится в «Грес», а «Иван» станет «Навий».

Система программирования: Microsoft Visual C#

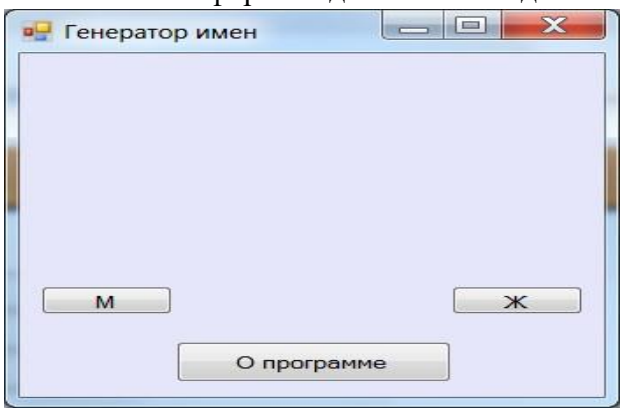
Особенности реализации и возникшие трудности: заранее создаются 2 текстовых файла в кодировке Unicode с мужскими и женскими именами. Из данного файла выбирается одна строка случайным образом. После обрезки последней буквы (если необходимо) данное слово помещается в массив символов, в котором используется функция reverse(). После дальнейших манипуляций с массивом он снова образует текстовую переменную, которая выводится в текстовое поле.

Достоинства приложения: простой понятный интерфейс, большое количество комбинаций, можно изменять исходные файлы.

Недостатки приложения: невозможность добавления полученного имени в новый файл, нет контроля на благозвучность имени.

Разработка интерфейса. В приложении использовались следующие компоненты: текстовое поле для вывода результата (RichTextBox), 2 кнопки для выполнения соответствующего расчета (Button).

Окончательно форма задачи имеет вид:



Работа над приложением:

1) Random rnd = new Random(); - инициализация датчика псевдослучайных чисел.

2) string[] imyaM = File.ReadAllLines("m.txt"); - чтение данных из текстового файла в строковую переменную.

3) string resm=imyaM[rnd.Next(0, linesCount)]; - выбор случайной строки из файла. 4) конвертация в массив символов и переворачивание:

char[] resm2 = resm.ToCharArray(); Array.Reverse(resm2);

5) if (resm[resm.Length - 1] == 'a') resm = resm.TrimEnd('a'); - пример удаления последней буквы в женском имени. 6) Пример добавления окончания к женскому имени:

if (resm3[resm3.Length - 1] == 'a' || resm3[resm3.Length - 1] == 'e' || resm3[resm3.Length - 1] == 'и' || resm3[resm3.Length - 1] == 'o') resm3 = String.Concat(resm3, "я"); else resm3 = String.Concat(resm3, "a");

Задача 2.1 а

Постановка задачи. Расширенная версия генератора мужских и женских имен и фамилий.

Внешнее описание: из массивов с буквами формируются по 2 мужских и женских имени с разными окончаниями, а также по одной мужской и женской фамилии с тем же корнем.

Функциональная спецификация: алгоритм для каждого имени и фамилии (мужское и женское) реализуется с помощью обработчика одной кнопки. С помощью датчика случайных чисел из массивов с гласными и согласными буквами формируется основа (корень слова), а затем к нему дописывается окончание по правилу русского языка.

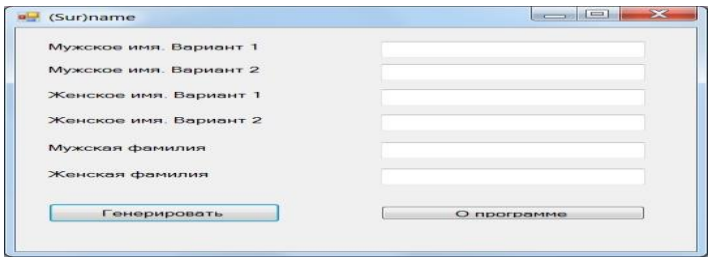
Система программирования: Microsoft Visual C#

Особенности реализации и возникшие трудности: чтобы слово начиналось как с гласной, так и с согласной буквы, создается дополнительный массив из гласных букв и пробелов для формирования первой буквы слова. Для удобочитаемости имени гласные и согласные буквы в слове чередуются. К сожалению, это откладывает негативный отпечаток на генератор, например, нельзя создать слово с двумя согласными подряд – «Петр, Петров», или с двумя гласными – «Виола, Виолова».

Достоинства приложения: простой понятный интерфейс, большое количество комбинаций для имен и фамилий.

Недостатки приложения: невозможность добавления полученного имени и фамилии в новый файл, нет контроля на благозвучность имени, не учитывается частота появления буквы в именах и фамилиях. Разработка интерфейса. В приложении использовались следующие компоненты: текстовые поля для вывода результатов (TextBox), кнопки для выполнения соответствующего расчета и вывода информации о программе (Button).

Окончательно форма задачи имеет вид:



Задача 2.1 б

Постановка задачи. Генератор учетных данных пользователя.

Внешнее описание: программа состоит из двух частей – генерирование имен, фамилий, отчеств и генерирования логина и пароля. Полученный результат можно скопировать в буфер обмена.

Функциональная спецификация: для реализации алгоритма первой части задачи заранее создаются 9 текстовых файлов: основы имени, фамилии и отчества (без окончаний), окончания имен, фамилий и отчеств для мужского и женского варианта. Для второй части задания создаются массивы с буквами, цифрами и другими символами, из которых комбинируются учетные данные.

Система программирования: Microsoft Visual C#

Особенности реализации и возникшие трудности: необходимо записать каждый текстовый файл в кодировке Unicode. Затем подгрузить его командой, например, для генерации мужского имени:

```
string[] imyaM = File.ReadAllLines("Names.txt");
```

В этом коде каждая строка (т.е. каждое имя) записывается в символьный массив, в котором происходит дополнительная обработка. С помощью датчика случайных чисел по индексу массива в текстовое поле помещается случайно выбранная основа имени и случайно выбранное окончание:

```
richTextBox1.Text = imyaM[rnd.Next(0, 104)] + oconchaniya_imen_M[rnd.Next(0, 23)];
```

Аналогичным образом добавляются сформированные фамилия и отчество. Функция rnd.Next(0,104) означает выбор случайного числа из диапазона от 0 до 104, учитывая, что в текстовом файле, а значит, и в массиве содержится 105 строк.

Имена в файле не делятся на женские и мужские, они регулируются только окончаниями, например, от корня «ларис» можно получить мужские имена «Ларисур», «Ларисий» и женские «Лариса», «Ларисия», что представляет широкий диапазон для творчества.

Процедуры для создания логина и пароля почти одинаковы, различаются только количество элементов в символьных массивах (для пароля добавлено больше символов) и длиной результирующей комбинации. Создать массив символов из строки можно несколькими способами, в этом приложении использовалась конвертация:

```
char[] data = symbols.ToCharArray(0, symbols.Length);
```

Затем формируется последовательность из случайно взятых (по индексу) символов:

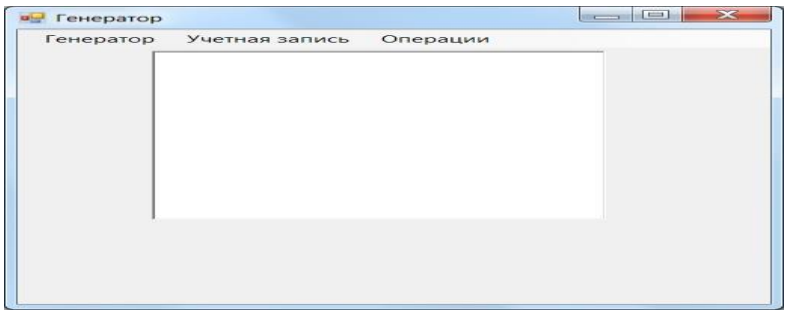
```
s += Convert.ToString(data[rnd.Next(0, (symbols.Length - 1))]);
```

В этом алгоритме не учитывается регистр букв (заглавные или строчные), нет проверки на похожесть некоторых символов, например, цифры «0» и буквы «О». Кроме того, такой пароль не поддерживает правила криптографии на устойчивость, но для демонстрационных целей он подходит.

Достоинства приложения: простой понятный интерфейс, большое количество комбинаций для имен и фамилий.

Недостатки приложения: невозможность добавления полученного имени и фамилии в новый файл, нет контроля на благозвучность имени, не учитывается частота появления буквы или цифры в логинах и паролях.

Разработка интерфейса. В приложении использовались следующие компоненты: расширенное текстовое поле для вывода результата (RichTextBox) и меню для выполнения всех операций в программе (menuStrip). Окончательно форма задачи имеет вид:



Задача 2.1 в

Постановка задачи. фай

Внешнее описание: по нажатию на кнопку с нужным полом генерируется либо мужское, либо женское полное имя.

Функциональная спецификация: текстовые файлы с шаблонами имен и их окончаний записываются в ресурсы проекта. Затем с помощью трех разных алгоритмов (комбинация основы и окончания из файла; инверсия букв в готовых именах из файла; набор последовательности из отдельных букв случайным образом) формируется строковый массив из трех имен. После этого с помощью датчика псевдослучайного числа выбирается одно имя и записывается в текстовое поле результата. Аналогичные операции осуществляются для фамилии и отчества. Два одинаковых алгоритма с разными исходными данными реализуются в обработчике соответствующей кнопки.

Система программирования: Microsoft Visual C#

Особенности реализации и возникшие трудности: для простоты рассмотрим алгоритм на примере генерирования мужского имени.

1) Комбинация данных из двух текстовых файлов:

string imyaM = Properties.Resources.Names; //создание текстовой переменной с данными из файла ресурсов

genmi[0] = imyaM1[rnd.Next(0, 257)].ToString() + oconchaniya_imen_M1[rnd.Next(0, 23)].ToString(); // формирование имени из случайно выбранных частей

При формировании отчества могут возникнуть трудности с изменением букв, например, «Валерий – Валерьевич». В программе не предусмотрены случаиисключения, поэтому можно получить некорректное отчество «Валеривич».

2) Инверсия букв из слов файле с последующим дописыванием окончания:

string[] imyaM22 = imyaM2.Split('\n'); // создается массив строк string resm = imyaM22[rnd.Next(0, linesCount)]; // выбирается случайное имя

if (resm[resm.Length - 1] == 'й') resm = resm.TrimEnd('й'); // удаление последней буквы «й» из имени

char[] resm2 = resm.ToCharArray(); // создание массива из отдельных букв основы Array.Reverse(resm2); // запись символов в обратном порядке string resm3 = new string(resm2); // создание новой текстовой переменной resm3 = resm3.Trim(); // удаление лишних пробелов в начале и конце слова resm3 = resm3.Substring(0, 1).ToUpper() + resm3.Substring(1, resm3.Length - 1); // выделение первой буквы слова, чтоб сделать ее прописной, а остальные буквы записать строчными if (resm3[resm3.Length - 1] == 'а' || resm3[resm3.Length - 1] == 'е' || resm3[resm3.Length - 1] == 'и' || resm3[resm3.Length - 1] == 'о')

resm3 = String.Concat(resm3, "й"); // добавление к имени буквы «й» последней, если основа заканчивается на гласную

genmi[1] = resm3; // запись сформированного имени во второй элемент массива

При формировании отчества могут возникнуть те же проблемы с некорректностью формирования окончания.

3) Формирование последовательности из символьных массивов с гласными и согласными буквами, учитывая их чередование:

string m1 = n[rnd.Next(0, 14)].ToString() + (char)s[rnd.Next(0, 17)] … // формирование первых двух букв имени

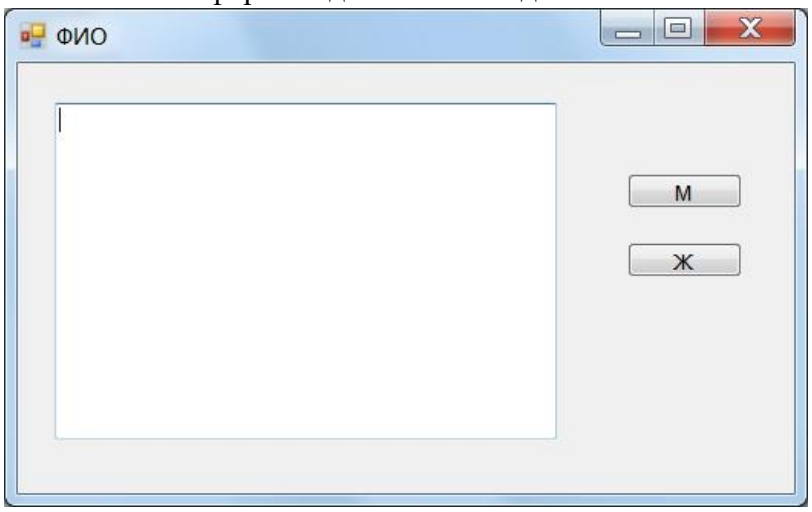
textBox1.Text= genmi[rnd.Next(0, 2)].ToString(); // выбор окончательного случайного имени из трех сформированных

Достоинства приложения: простой понятный интерфейс, большое количество комбинаций для имен и фамилий.

Недостатки приложения: невозможность добавления полученного имени и фамилии в новый файл, нет контроля на благозвучность имени, не согласованность основы с окончанием в отчествах.

Разработка интерфейса. В приложении использовались следующие компоненты: текстовое поле для вывода результата (TextBox) и две кнопки для реализации алгоритма (Button).

Окончательно форма задачи имеет вид:



Задача 2.2

Постановка задачи: генератор имен путем случайного выбора гласных и согласных букв алфавита.

Внешнее описание: щелчком по кнопке генерируется имя и фамилия, которые отображаются в статическом текстовом поле.

Функциональная спецификация: Используются 2 массива из гласных и согласных букв. Из каждого массива случайным образом выбирается по одной букве и записывается в одну строку.

Система программирования: Python

Особенности реализации и возникшие трудности: для случайного выбора числа из определенного промежутка используется функция randint библиотеки random.

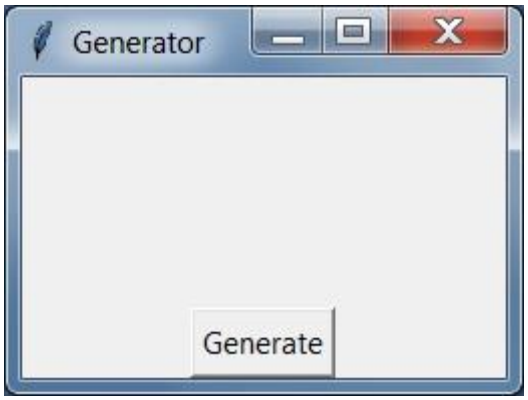
Сначала выбирается случайная длина слова – будущего имени, например, random.randint(3,8) – от 3 до 8 символов.

Это число будет конечным значением цикла в генераторе. Затем реализуется алгоритм чередования, когда гласная буква сменяется согласной и наоборот:

```
if(x == 0 or x % 2 == 0):
        firstName = firstName + bLetters[random.randint(0,18)]
else:           firstName = firstName + aLetters[random.randint(0,7)]
```

Аналогичным образом выглядит алгоритм для фамилии. Затем оба слова, записанные с прописной буквы, через пробел формируются в одной текстовой переменной для вывода.

Достоинства приложения: простой интерфейс, реализация основной функции.
Недостатки приложения: нет проверки на благозвучие имени.
Разработка интерфейса. В приложении использовались следующие виджеты: статическое текстовое поле для вывода (Label), кнопка генерации (Button). Окончательно форма задачи имеет вид:



Чтобы избавиться от предыдущего недостатка, изменим алгоритм на следующий:

1) Из букв случайным образом формируются два слога: а) гласная-согласнаягласная; б) согласная-гласная-согласная:

```
firstName=aLetters[random.randint(0,7)].upper()+bLetters[random.randint(0,18)]+aLetters[r
        andom.randint(0,7)]
```

2) В цикле эти слоги помещаются в два списка (по аналогии со списками из файлов): first_names.append(firstName)

3) Из каждого списка выбираются случайные слова, которые соединяются в одно имя:

```
first = random.sample(first_names, 1)[0] last = random.sample(last_names, 1)[0] name=first +
        last
```

Задача 3.2 а.

Постановка задачи. Текстовый редактор с дополнительными функциями. Внешнее описание: создание, открытие и сохранение файлов форматов .rtf, .txt. Работа с основными функциями форматирования и вставки текста. Функциональная спецификация: создание, открытие, сохранение, печать файла; работа с буфером обмена, вставка и показ даты и времени; изменение шрифта и цвета фона, поиск и замена слов; вывод количества символов и строк, вставка сгенерированных имен, фамилий и паролей. Система программирования: Microsoft Visual Basic.

Особенности реализации и возникшие трудности: по умолчанию текстовое поле в VB не поддерживает автоматическое контекстное меню, поэтому необходимо создавать его отдельно вручную. Для простоты реализации все операции над текстом проводятся только в пунктах основного меню. Отдельная проблема с

форматом файла. Для работы с rich text и plain text в этой среде предусмотрены разные методы, поэтому необходимый метод вызывается исходя из условия:

```
Try
RichTextBox1.LoadFile(OpenFileDialog1.FileName, RichTextBoxStreamType.RichText)
    Catch
RichTextBox1.LoadFile(OpenFileDialog1.FileName, RichTextBoxStreamType.PlainText)
    End Try
```

В этом фрагменте кода происходит попытка открыть .rtf документ. Если эта попытка не удачна, то документ открывается как .txt.

В программе реализованы основные функции работы с текстом, как в предыдущей задаче: открытие, сохранение, печать документа, работа с буфером обмена, поиск и замена слов, откат предыдущих действий, форматирование символов (шрифт, начертание, регистр, цвет).

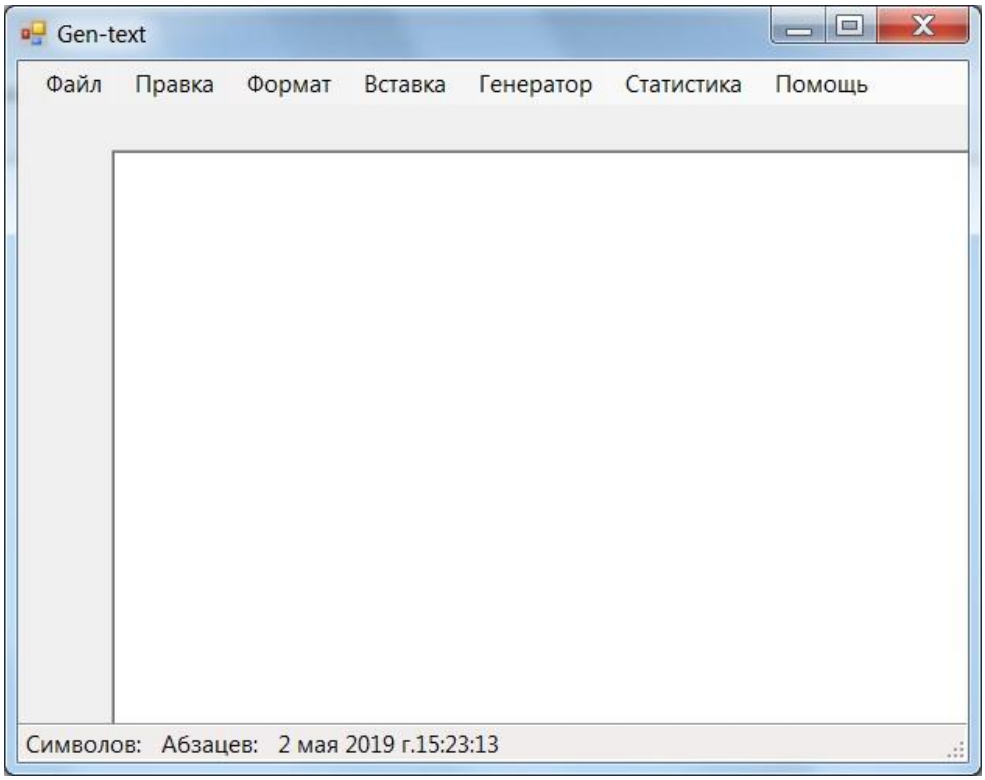
Кроме того, добавлена возможность вставки изображения, таблицы и списка, как в стандартном редакторе WordPad.

Дополнительно в программе реализован пункт меню «Генератор», в котором добавлены функции из задачи 3.1: вставка имени, фамилии, отчества, даты рождения, логина и пароля. И пункт «Статистика», позволяющий выводить числовую информацию для анализа символов.

Достоинства приложения: простой интерфейс, реализация основных и дополнительных функций текстового редактора с элементами генератора и статистических данных.

Недостатки приложения: нет быстрых кнопок для доступа к основным командам и контекстного меню.

Разработка интерфейса. В приложении использовались следующие компоненты: многострочное текстовое поле для основной работы (RichTextBox), основное меню (MenuStrip), стандартные диалоговые окна (OpenFileDialog, SaveFileDialog, FontDialog, ColorDialog, PrintDialog, PageSetupDialog, PrintDocument), таймер для отображения текущей даты и времени (Timer), строка состояния (StatusStrip). Окончательно форма задачи имеет вид:



Основные команды и процедуры кода приложения:

1) Сохранение файла:

```
If SaveFileDialog1.FileName <> "" Then RichTextBox1.SaveFile(SaveFileDialog1.FileName)
        RichTextBox1.Modified = False
```

2) Печать документа:

```
PrintDialog1.Document = PrintDocument1
    Dim result As DialogResult = PrintDialog1.ShowDialog()
    '   'Если сделан щелчок на OK, печатаем документ на принтере
    If result = DialogResult.OK Then
       PrintDocument1.Print()
    End If
```

3) Смена шрифта для выделенного текста:

```
FontDialog1.ShowDialog()
RichTextBox1.SelectionFont = FontDialog1.Font
```

4) Поиск слов:

```
Dim txt As String
    txt = InputBox("Поиск слов", "Найти?", "")
    RichTextBox1.Find(txt)
```

5) Замена одного слова на другое:

```
Dim txt, zam As String
    txt = InputBox("Поиск слов", "Найти?", "")      zam = InputBox("Заменить?",
        "Замена", "")
    RichTextBox1.Text = RichTextBox1.Text.Replace(txt, zam)
```

6) Вставить логин (как случайную комбинацию латинских букв):

```
Dim n() As String
    Dim i, k As Integer      Dim name As String      name = ""      ReDim n(26)
    Randomize()
    Const ab As String = "a b c d e f g h i j k l m n o p q r s t u v w x y z"      n = Split(ab, "
        ")      k = Int((6 * Rnd()) + 3)      For i = 1 To k
      name = name + n(Int(26 * Rnd()))      Next
    name = name.Substring(0, 1).ToUpper + name.Substring(1, name.Length - 1)
    RichTextBox1.Text += name + " "
```

7) Вставка случайного мужского имени из массива:

```
Randomize()
    Dim name1 As String() = New String(30) {"Андрей", "Борис", "Виктор", "Глеб",
"Денис", "Егор", "Александр", "Алексей", "Артем", "Арсений", "Владислав", "Дмитрий",
"Евгений", "Иван", "Игорь", "Илья", "Кирилл", "Максим", "Матвей", "Михаил",
        "Никита",
"Роман", "Руслан", "Сергей", "Тимофей", "Тимур", "Ярослав", "Антон", "Семён",
        "Николай", "Степан"}
    RichTextBox1.Text += name1(Int(30 * Rnd())) + " "
```

8) Вставка даты рождения, учитывая количество дней (28, 30, 31) в каждом месяце:

```
Dim dr As String
```

```
Dim y, m, d As Integer
Randomize()
y = Int(99 * Rnd() + 1910)        m = Int(12 * Rnd() + 1)
If m = 1 Or m = 3 Or m = 5 Or m = 7 Or m = 8 Or m = 10 Or m = 12 Then d = Int(31 *
      Rnd() + 1)
If m = 4 Or m = 6 Or m = 9 Or m = 11 Then d = Int(30 * Rnd() + 1)        If m = 2 Then d
      = Int(28 * Rnd() + 1)
dr = Str(d) + "." + Str(m) + "." + Str(y) + " "
RichTextBox1.Text += dr
```

9) Женские имена, а также фамилии, отчества, пароль генерируются аналогичными способами.

10) Подсчет авторских листов (1 авторский лист составляет 40000 знаков с пробелами):

```
Dim k As Integer        Dim s As Single        k = RichTextBox1.TextLength        s =
      Math.Round(k / 40000, 2)
MsgBox(Str(s) & " alk")
```

11) Стандартный метод подсчета строк в тексте Lines.Count не учитывает автоматический перенос текста на новую строку, поэтому правильнее назвать эту операцию подсчетом количества абзацев. Эта информация высвечивается в статус-строке при каждом обновлении текста, поэтому используется процедура RichTextBox1_TextChanged():

```
Dim st, k As Integer        k = RichTextBox1.TextLength        st = RichTextBox1.Lines.Count
ToolStripStatusLabel1.Text = "Символов: " & Str(k)
ToolStripStatusLabel2.Text = "Абзацев: " & Str(st)
```

12) Отображение в статус-строке текущей даты и времени формируется в процедуре Timer1_Tick():

```
ToolStripStatusLabel3.Text = Now.ToLongDateString & Now.ToLongTimeString
```

13) Прописные буквы:

```
RichTextBox1.SelectedText = RichTextBox1.SelectedText.ToUpper()
```

14) Подсчет количества слов, первое из которых выделено в тексте:

```
Dim txt, slovo As String        txt = RichTextBox1.Text        txt = txt.Replace(vbLf, " ")
      slovo = RichTextBox1.SelectedText
slovo = slovo.Trim
If slovo = "" Then MsgBox("Выделите нужное слово!")
Dim all() As String        all = txt.Split()        Dim i, k As Integer
For i = 0 To UBound(all)
  If all(i) = slovo Then k = k + 1
Next
MsgBox(Str(k))
```

15) Поиск в тексте чисел и их суммирование:

```
Dim txt As String
Dim summa As Single        txt = RichTextBox1.Text        Dim all() As String
      summa = 0        Dim i As Integer        txt = txt.Replace(vbLf, " ")        all =
      txt.Split()        For i = 0 To UBound(all)        If IsNumeric(all(i)) Then
      summa += all(i)
```

```
                RichTextBox1.SelectionStart = txt.IndexOf(all(i))
                RichTextBox1.SelectionLength = all(i).Length
                RichTextBox1.SelectionColor = Color.Blue
            End If
        Next
        MsgBox(Str(summa))
```

Для этой операции сначала все строки текста помещаются в переменную с единым разделителем «пробел». Затем из этой строки создается массив строк. Если элемент массива – число, то оно прибавляется к предыдущему. Каждая позиция числа в тексте выделяется, и цвет числа меняется на синий.

16) В выделенном тексте начертание меняется на курсив и обратно с курсива на обычный:

```
If RichTextBox1.SelectionFont.Italic Then
        RichTextBox1.SelectionFont = New Font(RichTextBox1.Font.FontFamily,
RichTextBox1.Font.Size, FontStyle.Regular)
    Else
        RichTextBox1.SelectionFont = New Font(RichTextBox1.Font.FontFamily,
RichTextBox1.Font.Size, FontStyle.Italic)
    End If
```

17) Вставка рисунка с помощью диалогового окна открытия файла:

```
OpenFileDialog1.ShowDialog()
    Dim bmp As New Bitmap(OpenFileDialog1.FileName)
    Clipboard.SetImage(bmp)        RichTextBox1.Paste()
```

18) Преобразование выделенного текста в маркированный список:

```
If Not RichTextBox1.SelectionBullet Then
        RichTextBox1.SelectionBullet = True
        RichTextBox1.SelectionIndent = 20
        RichTextBox1.SelectionHangingIndent = 15
        RichTextBox1.SelectionRightIndent = 20
    Else
        RichTextBox1.SelectionBullet = False          RichTextBox1.SelectionIndent = 0
        RichTextBox1.SelectionHangingIndent = 0
        RichTextBox1.SelectionRightIndent = 0
    End If
```

19) Вставка простой таблицы из одной строки и двух столбцов. Клавиша Enter добавляет новую строку к таблице. Код опирается на тэги формата .rtf:

```
    Dim table As String
    table = "\trowd\cellx1000\cellx2000\intbl \cell\intbl \cell\row"
    Dim Index As Integer
    Index = RichTextBox1.Rtf.LastIndexOf("}")
    RichTextBox1.Rtf = RichTextBox1.Rtf.Substring(0, Index) + table + "}"
```

20) Открытие стандартной программы Windows «Таблица символов» для вставки символа:

```
    Shell("CharMap.exe")
```

21) Дополнительная процедура PrintDocument1_PrintPage() для печати текстовой области со всем содержимым:

```
Dim printFont = New Font("Arial", 12)
    ' Отрисовываем текст
    e.Graphics.DrawString(RichTextBox1.Text, printFont, Brushes.Black,
e.MarginBounds, New StringFormat())
    ' Страниц больше нет
    e.HasMorePages = False
```

22) Справка создана с использованием стандартного диалогового окна MsgBox(), а пункт меню «О программе» с помощью вызова модального окна (новой формы).

Задача 3.2 б

Постановка задачи. Простой текстовый редактор с основными функциями.

Внешнее описание: создание, открытие и сохранение файлов формата .txt.

Работа с основными функциями форматирования и вставки текста. Функциональная спецификация: создание, открытие, сохранение файла; работа с буфером обмена, вставка и показ даты и времени; изменение шрифта и цвета, вставка рисунка. Система программирования: Python

Особенности реализации и возникшие трудности: в стандартной библиотеке Tkinter() нет многих виджетов для диалоговых окон: печать, выбор шрифта и т.д. Эти функции можно реализовать либо через системные библиотеки Windows API, либо через другие модули Python (например, PyQt5). В данном приложении рассмотрим только выбор нескольких шрифтов из выпадающего списка, созданного вручную.

Каждая операция реализована с помощью отдельной функции, имя которой записывается в качестве имени команды для кнопки или пункта меню, или пункта контекстного меню.

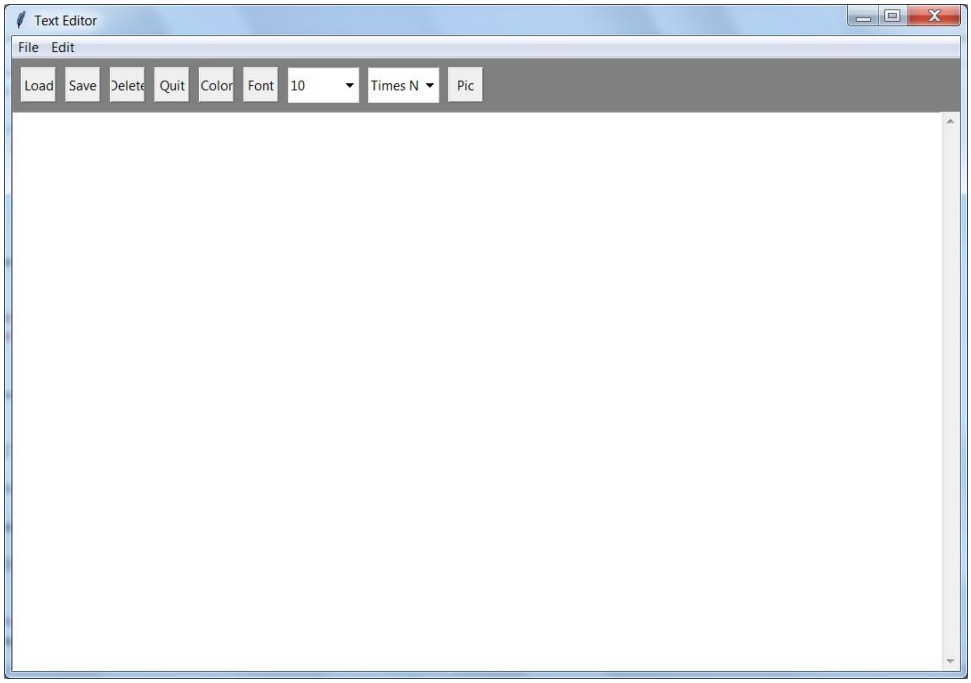
Открытие файла с изображением: эта функция корректно работает только при непосредственном вводе имени файла. Если использовать диалоговое окно FileDialog, то необходимо функции реализовывать как методы внутри класса. В противном случае имя файла нельзя использовать в других частях программы, если процедура открытия была реализована в функциообработчике.

Достоинства приложения: простой интерфейс, реализация основных функций текстового редактора.

Недостатки приложения: нет большинства функций, используемых в классических текстовых редакторов (смена шрифта и начертания, печать документа).

Разработка интерфейса. В приложении использовались следующие виджеты: многострочное текстовое поле для основной работы (Text), основное меню (Menu), стандартные диалоговые окна (FileDialog, ColorChooser), кнопки быстрого доступа (Button), выпадающие списки для имени и размера шрифта (Combobox).

Окончательно форма задачи имеет вид:



Основные команды и процедуры кода приложения:

1) Открытие файла:

```
fn = filedialog.Open(root, filetypes = [('*.txt files', '.txt')]).show()     if fn == '':         return
    textbox.delete('1.0', 'end')
    textbox.insert('1.0', open(fn, 'rt').read())
```

Для использования диалогового окна его необходимо импортировать из библиотеки: from tkinter import filedialog. Если выбран текстовый файл, то его открыть в режиме чтения - 'rt'.

2) Сохранение файла:

```
fn = filedialog.SaveAs(root, filetypes = [('*.txt files', '.txt')]).show()     if fn == '':         return
        if not fn.endswith(".txt"):
        fn+=".txt"
    open(fn, 'wt').write(textbox.get('1.0', 'end'))
```

Это же диалоговое окно с методом SaveAs() позволяет сохранить текст из области под файловым именем.

3) Удаление текста (или создание нового файла):

```
answer=messagebox.askyesno("Подтверждение", message="Вы хотите удалить текст?")
        if answer==True:
        textbox.delete(0.0, END)
```

Для этой операции используется диалоговое окно messagebox, которое аналогично добавляется в список импортируемых виджетов, к нему применяется метод askyesno(), и при положительном ответе содержимое текстовой области удаляется.

4) Смена цвета символов при помощи диалогового окна:

```
(rgb, hx) = colorchooser.askcolor()     textbox.config(foreground=hx)
```

Если “foreground” заменить на bg (background), то изменится цвет фона текстовой области.

5) Смена шрифта:

Т.к. готового виджета диалогового окна в стандартной библиотеке не существует, то в качестве аргументов функции используются данные, заранее выбранные в поле комбинированного списка:

textbox.config(font=(fname.get(), fsize.get()))

При этом в строку импорта виджетов добавляется “font”, а в основной программе при описании виджетов добавляется Combobox, например:

fname=StringVar() #создание текстовой переменной

combo2 = Combobox(panelFrame) #инициализация поля со списком внутри рамки

combo2['values'] = ('Times New Roman', 'Monotype Corsiva', 'Courier', 'Arial', 'Impact', 'Comic Sans MS') #перечень элементов списка – названий для шрифтов combo2.current(0) #установка курсора на первом элементе списка fname=combo2 #присваивание выделенного элемента списка текстовой переменной combo2.place(x=400,y=10,width=80,height=40) #размещение поля со списком на форме

6) Копирование текста в буфер обмена: textbox.event_generate('<<Copy>>')

7) Добавление даты и времени к тексту (необходимо импортировать соответствующие модули):

textbox.insert(END, datetime.now())

8) Формирование контекстного меню:

pmenu.tk_popup(event.x_root, event.y_root)

…

pmenu = Menu(textbox)

pmenu.add_command(label='Cut', accelerator='Ctrl+X', command=cut)

… textbox.bind('<Button-3>', popup)

9) Формирование пунктов основного меню, например, «Файл»:

menuBar = Menu(root) fileMenu = Menu(menuBar)

fileMenu.add_command(label="New", accelerator='Ctrl+N', command=delete_text)

…

menuBar.add_cascade(label="File", menu=fileMenu)

10) Формирование одной из кнопок с командой (на примере открытия файла):

loadBtn = Button(panelFrame, text = 'Load') loadBtn.bind("<Button-1>", LoadFile)

loadBtn.place(x = 10, y = 10, width = 40, height = 40)

11) Формирование текстовой области внутри рамки с полосами прокрутки (scrollbar):

textFrame.pack(side = 'bottom', fill = 'both', expand = 1) textbox = Text(textFrame, font='Arial 14', wrap='word') scrollbar = Scrollbar(textFrame) scrollbar['command'] = textbox.yview textbox['yscrollcommand'] = scrollbar.set textbox.pack(side = 'left', fill = 'both', expand = 1) scrollbar.pack(side = 'right', fill = 'y')

12) Вставка изображения в текстовую область:

Для работы с изображениями формата .jpg необходимо подключить дополнительную библиотеку Python Image Library (PIL):

from PIL import Image, ImageTk

В процедуре-обработчике для кнопки написать команду вставки изображения в текстовое поле в указанную позицию курсора:

textbox.image_create(INSERT,image=img)
В основной программе добавить строку с явным указанием имени файла:
img = ImageTk.PhotoImage(Image.open("smile.jpg"))

Задача 3.2 в

Постановка задачи. Текстовый редактор с поддержкой вычислений. Внешнее описание: создание, открытие и сохранение файлов форматов .rtf, .txt. Работа с основными функциями форматирования и вставки текста. Функциональная спецификация: создание, открытие, сохранение, печать файла; работа с буфером обмена; изменение шрифта и цвета фона, поиск и замена слов; вывод количества совпадений, слов, строчных и прописных букв, подсчет суммы, произведения, разности, частного и среднего арифметического выделенных в тексте чисел. **Система программирования: Microsoft Visual C#.**

Особенности реализации и возникшие трудности: часть функций дублируется в пунктах меню и на панели кнопок простым копированием в обработчиках событий. Операции «Открыть файл» и «Сохранить как» реализуются с помощью стандартных диалоговых окон. Для операции «Сохранить файл под существующим именем» необходимо задать переменную класса, в которой будет храниться имя ранее открытого или сохраненного файла.

Фрагмент кода:

```
using System;
using System.IO;
using System.Linq;
using System.Windows.Forms;

namespace TextEditorWithCalculations
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      // Создание нового файла
      private void CreateNewFile()
      {
         richTextBox1.Clear();
      }

      // Открытие файла
      private void OpenFile()
      {
         OpenFileDialog openFileDialog1 = new OpenFileDialog();
         openFileDialog1.Filter = "Text Files|.txt|Rich Text Files|.rtf";
         if (openFileDialog1.ShowDialog() == DialogResult.OK)
         {
            string fileText = File.ReadAllText(openFileDialog1.FileName);
            richTextBox1.Text = fileText;
         }
```

```
        }

        // Сохранение файла
        private void SaveFile()
        {
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();
            saveFileDialog1.Filter = "Text Files|.txt|Rich Text Files|.rtf";
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Text);
            }
        }

        // Остальные функции добавления форматирования, вычислений, поиска и
            замены слов, буфера обмена, вывода статистики и т.д. должны быть
            реализованы аналогично.

        private void createToolStripMenuItem_Click(object sender, EventArgs e)
        {
            CreateNewFile();
        }

        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            OpenFile();
        }

        private void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveFile();
        }
    }
}
```

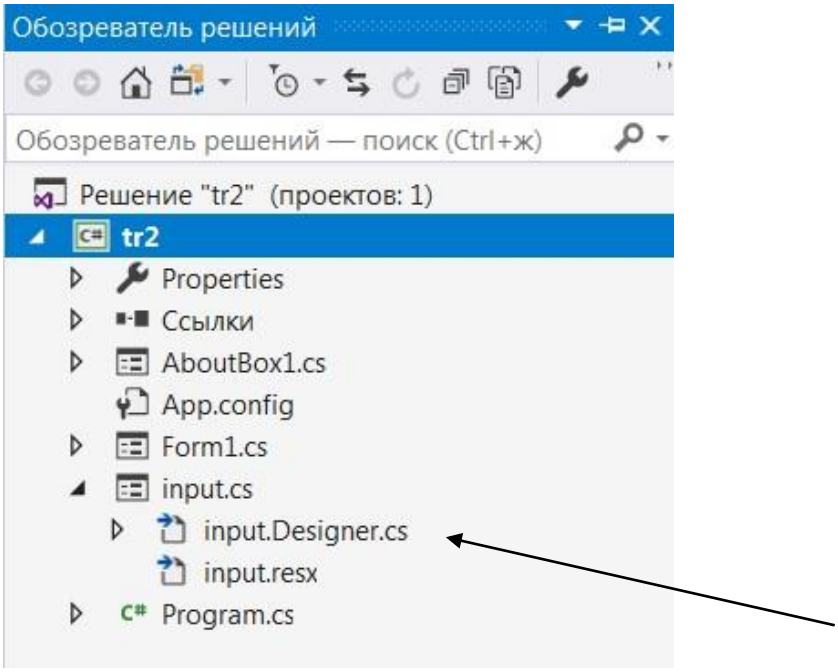
Если имя файла – пусто (т.е. он не был открыт или сохранен), то управление передается функции-обработчику сохранения файла под именем.

К сожалению, в C# нет диалоговых окон для поиска и замены текста и инструментов для ввода значений (по типу InputBox), поэтому для реализации этих операций используется 3 основных способа:

1) Подключить соответствующий модуль из Visual Basic.
2) Создать пользовательский класс с необходимыми элементами управления и методами.
3) Создать новую форму и разрешить обмен данными между основной формой и созданной.

Воспользуемся в данной задаче последним способом.

После создания формы (в примере она называется input) перейти в обозреватель решений и выбрать файл для изменения input.Designer.cs:



Компоненты, служащие для обмена с другой формой, объявить, как public:

```
public System.Windows.Forms.TextBox textBox1;
```

Теперь содержимое текстового поля доступно для основной формы. Метод ShowDialog() позволяет вывести на экран вторую форму в качестве модальной, т.е. автоматически расположенной поверх экрана. Закрытие модальной формы не влияет на основную.

Рассмотрим фрагмент кода с поиском слова в тексте:

```
    string txt;
        input input1 = new input(); //инициализация формы с вводом слова
            input1.Owner = this; //передача прав собственника текущей (основной) форме
            input1.ShowDialog(); //открытие модальной формы
        txt = input1.textBox1.Text; //присваивание значения из поля модальной формы
        MainText.Find(txt); //встроенный метод поиска
```

Для замены производятся аналогичные действия, только либо создается новая модальная форма с двумя текстовыми полями, либо созданная форма вызывается два раза. Затем для замены применяется метод replace(). Работа с буфером обмена аналогична, как в других системах. Например:

```
MainText.Copy();
```

Работа с изменением начертания символов производится с помощью подобных методов:

```
SetSelectionFont(FontStyle.Bold);
```

Рассмотрим подробно функции подсчета в тексте:

1) Количество совпадений среди слов:

```
string[] arrString = MainText.Text.Split(); //массив строк с любым разделителем из
            текстового поля
        var a = from aa in arrString group aa by aa.Substring(aa.IndexOf(aa));
//выделение одинаковой подстроки из каждой строки          string output = "";
            foreach (var item in a)
        {               output += $@"{item.Key} встречается:
            {item.Count()}{Environment.NewLine}";
```

```
            /*интерполяция строк @$. Этот символ позволяет указывать переменные,
              окруженные фигурными скобками, прямо в строках без использования
              конкатенации или форматирования.*/
        }
        MessageBox.Show(output, "Количество совпадений");
```

2) Количество слов в тексте:

```
string str;          int k = 0;          str = MainText.Text;
        string[] words = str.Split(new char[] { },
          StringSplitOptions.RemoveEmptyEntries); //преобразование текста в массив строк
          с удалением пустых строк
        foreach (string s in words) //цикл по элементам
        {            k++;
        }

        MessageBox.Show($"{k}", "Количество слов");
```

3) Количество прописных букв в тексте (фрагмент функции):

```
foreach (char bukva in stroka)
        {
          if (Char.IsUpper(bukva))                lb++;
}
```

4) Арифметические операции (на примере сложения):

```
try
        { decimal result = array[0];
          for (int i = 1; i < array.Count; i++)

            result += array[i];
            MessageBox.Show($" {result}", "Сумма");
                }
        catch (Exception)
        {
        }
```

Предварительно необходимо объявить массив, в котором будут храниться выбранные числа, и разрешить множественное выделение. Числа из текста выделаются с помощью комбинации клавиши Ctrl и движения мыши, поэтому весь метод выделения числа из текста и помещение его в числовой массив осуществляется с помощью соответствующих функций: - MainText_KeyDown() – разрешение множественного выделения и определения цвета выделения;

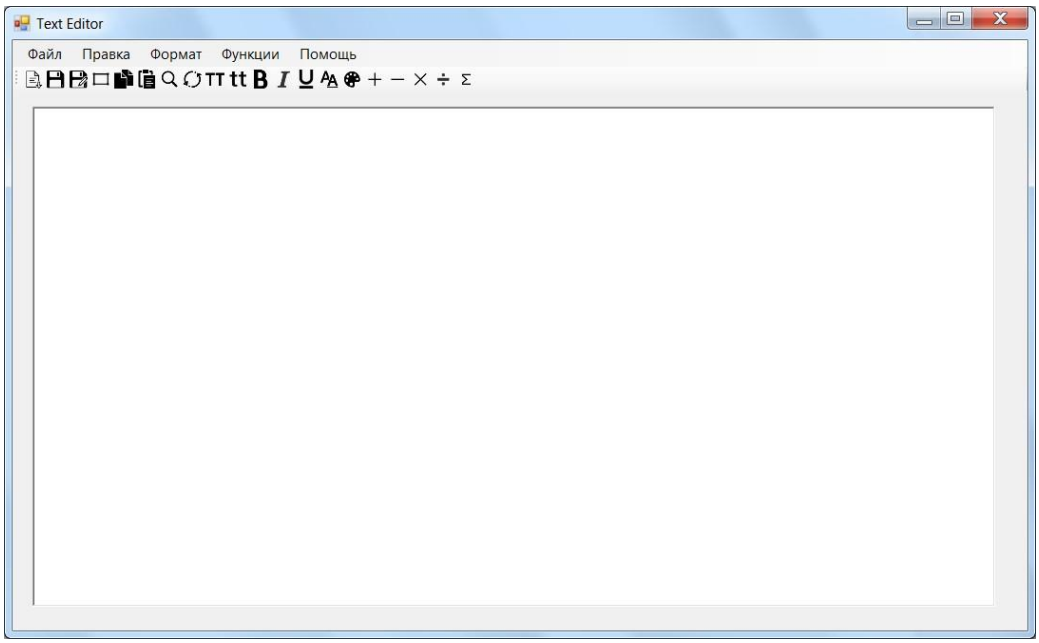
- MainText_MouseUp() – добавление при нажатой клавиши ctrl выделенного числа в массив;
- MainText_MouseClick() – восстановление первоначальных значений и обнуление массива.

Достоинства приложения: простой интерфейс, реализация основных и дополнительных функций текстового редактора с подсчетом числовых значений.

Недостатки приложения: нет контекстного меню.

Разработка интерфейса. В приложении использовались следующие компоненты: многострочное текстовое поле для основной работы (RichTextBox=MainText), основное меню (MenuStrip), панель кнопок (toolStrip - button), стандартные диалоговые окна (OpenFileDialog, SaveFileDialog, FontDialog, ColorDialog, PrintDialog, PageSetupDialog, PrintDocument.

Окончательно форма задачи имеет вид:



Для добавления пункта меню «О программе» использовалась встроенная форма. Она вызывается с помощью пункта: Проект – Добавить форму Windows – Окно «О программе». Данные для этой формы подставляются из окна: Проект – Свойства – Приложение – Сведения о сборке…

Подгружается это модальное окно в основной форме с помощью команды:

```
new AboutBox1().ShowDialog();
```

Задача 4.1.

Постановка задачи. Создание тестирующей программы для проверки знаний. Вариант №2. Все вопросы и варианты ответов находятся на одной форме. После прохождения теста в текстовом окне по нажатию кнопки высвечивается оценка.

Внешнее описание: каждый вопрос с вариантами ответа размещается на отдельной панели, кнопка «Оценка» неактивна до тех пор, пока в текстовом поле не введено имя.

Функциональная спецификация: В программе предусмотрена защита на повторное нажатие радиокнопки с вариантом ответа во избежание неверного результата. В текстовом поле «оценка» показывается не только значение, но и количество правильных ответов, а также уровень знаний.

Система программирования: Visual C#.

Особенности реализации и возникшие трудности: подсчет правильных ответов можно осуществить двумя способами:

1) через переменную-счетчик наращивания для каждого нажатия радиокнопки (как в предыдущей задаче);

2) сопоставление правильного варианта с элементом массива.

Первый способ быстрей компилируется, но код программы занимает много места из-за повторяющихся операторов. Второй способ работает медленней из-за инициализации массивов, но код программы намного короче, поэтому воспользуемся вторым способом.

Для этого создадим два числовых массива – один пустой с будущими номерами ответов, второй заполненный с правильными вариантами ответов, например: answersRight = new int[] { 1, 2, 2, 3, 4 }.

В функции обработки кнопки подсчета результата переменная-счетчик увеличивается внутри цикла:

```
for (int i = 0; i < answers.Length; i++)
        {
          if (answers[i] == answersRight[i])
          {
            countRightA++;              } }
```

В этой же функции производится подсчет количества баллов по схеме, которое затем выводится в текстовую область:

```
double level = (double)countRightA / answers.Length * 100;          string grade = "";
          if (level <= 50)              grade = "2";
```

Функция обработки щелчка по радиокнопке упрощена, благодаря тому, что можно обращаться к родительскому классу RadioButton, из которого выделяется необходимый индекс. Например, числа в названии radioButtonQ1Answer1 означают вопрос №1 и ответ №1.

Фрагмент кода:

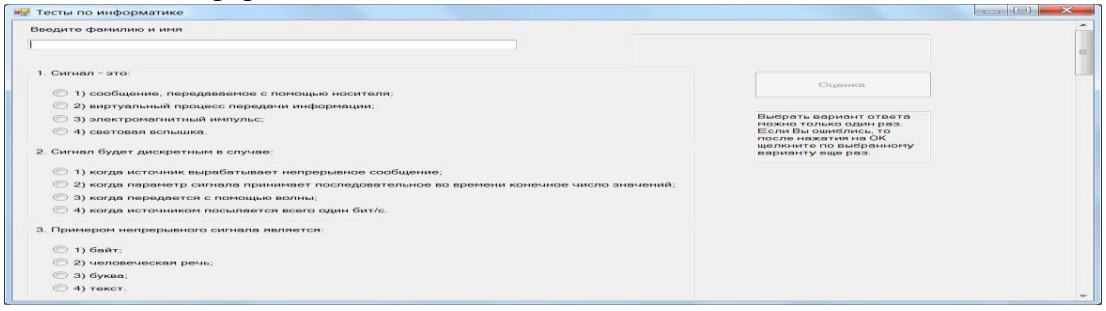
```
int indexQ = Convert.ToInt32((((RadioButton)sender).Parent.Name).Substring(16)); //
          выделение из имени радиокнопки номера вопроса
        int indexA = Convert.ToInt32((((RadioButton)sender).Name).Substring(12 +
          indexQ.ToString().Length + 6)); // выделение из имени номера ответа          if
          (answers[indexQ - 1] != 0)
        {
          MessageBox.Show("Выбирать вариант можно только один раз!"); //защита от
          повторного нажатия
          answers[indexQ - 1] = 0;
          ((RadioButton)sender).Checked = false;
        }          else          {
          answers[indexQ - 1] = indexA; //заполнение массива номеров ответов индексами
          нажатых радиокнопок.
        }
```

Достоинства приложения: одно окно для всех вопросов, возможность проходить тест не по порядку, защита от повторного нажатия.

Недостатки приложения: для большого количества вопросов приходится использовать полосы прокрутки.

Разработка интерфейса. В приложении использовались следующие компоненты: текстовые поля для ввода данных и вывода результатов (TextBox), панели для вопросов и ответов (GroupBox), радиокнопки выбора ответа (RadioButton), кнопка вывода результата (Button).

Окончательно форма задачи имеет вид:



Использованные материалы и Интернет-ресурсы

Основные источники:

1. Михеева Е.В. Практикум по информационным технологиям в профессиональной деятельности: учеб. пособие. - М.: Издательский центр «Академия», 2005. - 256 с.
2. Михеева Е.В. Титова О.И. Информационные технологии в профессиональной деятельности экономиста и бухгалтера: учеб. пособие. -М.: ОИЦ «Академия», 2008. -208 с.
3. Михеева ЕВ. Информационные технологии в профессиональной деятельности: учеб. пособие. - М.: ОИЦ «Академия», 2010. - 384 с.
4. Титоренко Г.А. Автоматизированные информационные технологии в экономике: учебник.- М.:ЮНИТИ, 2004.-399с.
5. Журин А.А. Самый современный самоучитель работы на компьютере. М.: ООО «Издательство ЛСТ»,2004 г.
6. Семёнов М.И. Автоматизированные технологии в экономике. Учебник – М.: Финансы и статистика, 2000 г.
7. Соколова Н.Я. Информационное обеспечение профессиональной деятельности. Рабочая тетрадь – М.: 2003 г.
8. Хаймер Л.А. Интеллектуальные информационные системы. Учебное пособие, 2002 г.

Дополнительные источники:

1. Байдаков В., Дранищев В. И. др. 1С:Предприятие 8.1. Руководство пользователя. - М.: Фирма «1С», 2008. -303 с.
2. Безека СВ. Создание презентаций в Ms PowerPoint 2007. - СПб.: ПИТЕР, 2010. - 275 с.
3. Харитонов С.А., Чистов Д.В. Хозяйственные операции в 1С:Бухгалтерия 8, Задачи, решения, результаты. - М.: 1С-Паблишинг, 2008. - 463 с.
4. Электронный ресурс: MS Office 2007 Электронный видео учебник. Форма доступа: http:// gigasize.ru.
5. Электронный ресурс: Российское образование. Федеральный портал. Форма доступа: http:// www.edu.ru/fasi.