|  |  |
| --- | --- |
| **Изображение выглядит как ночное небо  Автоматически созданное описание** | **автономная некоммерческая образовательная организация**  **высшего образования Центросоюза Российской Федерации**  **«Сибирский университет потребительской кооперации»** |

Кафедра информатики

**ОТЧЕТ ОБ ОЗНАКОМИТЕЛЬНОЙ** **ПРАКТИКЕ**

Место прохождения практики

АНОО ВО Центросоюза РФ СибУПК

Обучающегося(ейся) 2 курса

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(Фамилия И.О.)*

группы ПИБ-01

Руководитель практики

доцент кафедры информатики,

канд. пед. наук

***Колдунова Ирина Дмитриевна***

Оценка после защиты \_\_\_\_\_\_\_\_\_\_\_

Дата защиты 30.12.2021

Новосибирск

2021

|  |  |
| --- | --- |
| **Изображение выглядит как ночное небо  Автоматически созданное описание** | **автономная некоммерческая образовательная организация**  **высшего образования Центросоюза Российской Федерации**  **«Сибирский университет потребительской кооперации»** |

Кафедра информатики

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Обучающемуся(ейся) 2 курса очной формы обучения

направления подготовки 09.03.03 Прикладная информатика

направленность (профиль) Прикладная информатика в информационной сфере

ФИО обучающегося(ейся)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Шифр \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Вид практики: учебная

Тип практики: ознакомительная

Приказ о направлении на практику: № Сз-668-Э от 13 декабря 2021 г.

Срок прохождения практики с «17» декабря 2021г. по «30» декабря 2021г.

Место прохождения практики АНОО ВО Центросоюза РФ СибУПК

Срок сдачи студентом отчета на кафедру30.12.21

1. Перечень подлежащих разработке вопросов и общее направление работы:
2. Постановка задачи, краткое описание основных средств языка, которые предполагается использовать для решения поставленной задачи.
3. Разработка алгоритма решения поставленной задачи
4. Создание интерфейса приложения.
5. Тестирование приложения.

II. Объем отчета по практике \_\_\_\_\_\_\_\_\_\_ страниц (+приложение)

Задание выдано

Руководитель практики от университета Колдунова Ирина Дмитриевна, канд. пед. наук, доцент кафедры информатики СибУПК \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Задание получено

Обучающийся(аяся) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**Содержание**

[**Введение** 5](#_Toc90811128)

[**1.** **Теоретические основы разработки оконных приложений** 6](#_Toc90811129)

[**1.1.** **…** 6](#_Toc90811130)

[**1.2.** **…** 6](#_Toc90811131)

[**2.** **Разработка приложения** 7](#_Toc90811132)

[**2.1.** **Разработка сценария и алгоритма решения поставленной задачи** 7](#_Toc90811133)

[**2.2.** **Создание интерфейса приложения** 7](#_Toc90811134)

[**2.3.** **Тестирование приложения** 7](#_Toc90811135)

[**Заключение** 8](#_Toc90811136)

[**Библиографический список** 9](#_Toc90811137)

[**Приложения** 10](#_Toc90811138)

# Введение

Учебно-ознакомительная практика обеспечивает последовательность и непрерывность в формировании у студентов универсальных компетенций и навыков.

Целью данной практики является приобретение первоначального практического опыта, связанного с освоением специальности 09.03.03 Прикладная информатика, на основе разработки собственного приложения.

Учебно-ознакомительная практика ставит следующие задачи:

- обобщение, систематизация, конкретизация и закрепление теоретических знаний при разработке собственной программы, а также при помощи выполнения индивидуального задания;

- выполнение технических требований индивидуального знания;

- составление технической документации создания приложения информатизация прикладных процессов;

- овладение необходимым набором универсальных компетенций;

- получение практического опыта создания и тестирования собственного приложения;

- формирование практических умений: программирование на языке C# в соответствии с профилем подготовки, сбор необходимой информации для предметной области и т.д.

# Теоретические основы разработки оконных приложений

Язык C# был создан в 1998-2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework. C# относится к семье языков с C-подобным синтаксисом. Его синтаксис наиболее близок к C++ и Java. Учитывая опыт применения языков C++ и Java, разработчики C# исключили из него некоторые модели, вызывавшие при разработке приложений проблемные ситуации. Это в некоторой степени упростило язык и сделало его более надежным.

Приведем некоторые особенности C#:

• C# создавался параллельно с .Net Framework, поэтому в полной мере учитывает все возможности FCL и CLR;

• C# является полностью объектно-ориентированным языком, и даже типы, встроенные в язык, представлены классами;

• C# является наследником языков C/C++ и сохраняет лучшие черты этих популярных языков программирования. В то же время язык стал проще и надежнее;

• благодаря каркасу .Net Framework, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда CLR представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода;

• мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных.

# Платформа Microsoft .NET

Платформа Microsoft .NET представляет собой единую среду исполнения программ и поддержки их разработки.

Ключевыми задачами при построении платформы .NET являлись:

1. Поддержка разработки распределенных корпоративных приложений, включая серверные и мобильные компоненты, на базе Web-сервисов и XML.

2. Унификации библиотек функций для всех приложений, независимо от используемого языка программирования.

Платформа .NET основана на единой объектно-ориентированной модели; все сервисы, предоставляемые программисту платформой, оформлены в виде единой иерархии классов. Модель платформы .NET существенно упрощает разработку приложений по сравнению с программированием для Windows-платформ, где практически вся функциональность предоставлялась разработчику как неструктурированный набор функций в Windows API.

Проблема создания объектно-ориентированной надстройки над функциями Windows API решалась независимо в различных языках программирования. С появлением платформы .NET впервые в истории программирования применяется единая модель, позволяющая на равных пользоваться различными языками для создания приложений. Поскольку базовые классы .NET стали общие для всех систем программирования, то это означает, что изменилось соотношение язык - базовые функции. Если раньше каждый язык содержал свою библиотеку классов, представляющих надстройку над функциями Windows API, например MFC в VC++, то теперь язык программирования адаптируется к платформе. Библиотека классов MFC - это набор статических объектных модулей . Они подключаются к приложению на этапе компоновки исполняемого модуля программы и становятся при этом его неотъемлемой частью. В то же время .NET Class Library - это динамические библиотеки классов, которые являются составной частью операционной среды (специальным видом исполняемых модулей) и используются приложением только в момент его выполнения. Разработчики .NET лишены возможности выбора между статическими объектными библиотеками (LIB) и библиотеками классов (DLL) (за исключением тех, кто пишет на C/C++, которые занимают особое положение в средствах разработки .NET).

3. Повышения управляемости приложений с точки зрения эффективного использования ресурсов и безопасности.

# 1.2 Элемент ListBox

Иногда бывает необходимо вывести целый список данных, например, при составлении таблицы функции. В таких случаях для вывода можно использовать элемент ListBox, который соответствует списку. Рассмотрим пример его применения.

Пусть требуется составить таблицу функции 2 y = x на отрезке [a,b] с шагом h, причем концы отрезка и шаг должны определяться при выполнении программы. Для их ввода используем текстовые поля.

Создадим новое приложение. На форме расположим кнопку button1, три текстовых поля textBox1, textBox2, textBox3 для ввода 139 данных, четыре метки label1, label2, label3, label4 для размещения поясняющих надписей и список listBox1, как показано на рис.1

Создадим новое приложение. На форме расположим кнопку button1, три текстовых поля textBox1, textBox2, textBox3 для ввода 139 данных, четыре метки label1, label2, label3, label4 для размещения поясняющих надписей и список listBox1, как показано на рис.2

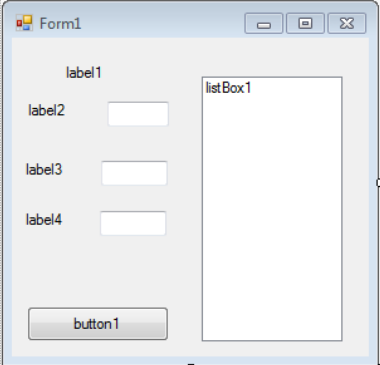


Рис.1. Форма с расположенными элементами

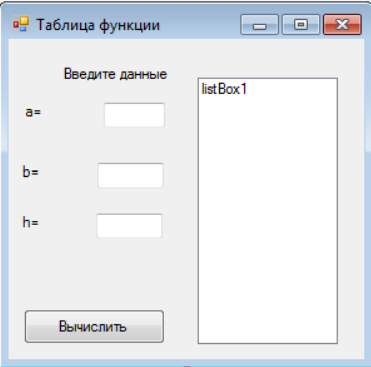


Рис.2. Форма после изменения свойства Text элементов

Теперь создадим обработчик события

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

{

double a = Convert.ToDouble(textBox1.Text);

double b = Convert.ToDouble(textBox2.Text);

double h = Convert.ToDouble(textBox3.Text);

decimal y;

string z;

for (double x = a; x < b + h / 2; x += h)

{

y = Math.Round((decimal)(x \* x),3);

z = "y(" + x.ToString() + ")=" + y.ToString();

listBox1.Items.Add(z);

}

}

Запустим приложение, введем данные и нажмем кнопку. Результат выполнения приложения приведен на рис.3.

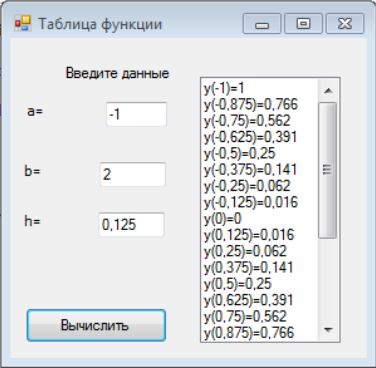


Рис.3. Результат выполнения приложения

Приведем некоторые комментарии к коду метода и полученному результату. Для переменной y мы использовали тип decimal, чтобы можно было применить метод Math.Round() для округления с точностью до тысячных. Члены списка могут принимать значения только типа string. Поэтому мы ввели переменную z типа string и присвоили ей требуемое значение. Последний оператор тела метода в коллекцию членов списка listBox1 добавляет новый член, т.е. новую строку, со значением 141 z. В результате добавляется следующее значение функции. По рис.3 видим, что при необходимости в окно списка добавляется полоса прокрутки.

Элемент ListBox можно использовать не только для вывода результата, но и для ввода данных. В этом случае на этапе конструирования интерфейса создается список с перечнем возможных вариантов некоторой переменной. После запуска приложения мышью выделяется нужный вариант и вызывается событие. Обработчик события выполняет соответствующие операторы с учетом выбранного члена списка. Рассмотрим пример.

Предположим, что у нас форма с тремя текстовыми полями, кнопкой и списком, содержащим две строки. В первой строке написано слово "сумма", а во второй – "произведение". Первые два текстовых поля предназначены для ввода двух чисел, по списку определяется вид операции. В третье текстовое поле должен выводиться результат выполнения выбранной операции.

Воспользуемся предыдущей формой, несколько подкорректировав расположения элементов и надписи. Более существенные преобразования выполним со списком. Приведем их. Выделим этот объект и в Окне свойств выберем свойство Items. Появится кнопка . Нажмем ее. В результате откроется окно Редактора коллекции строк списка (рис.4).

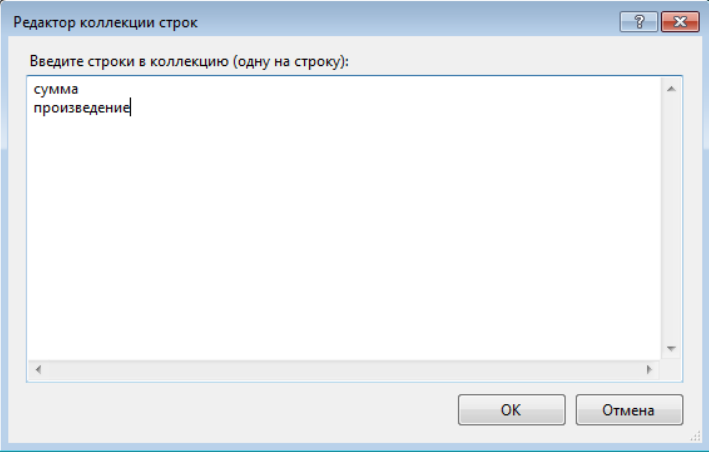


Рис.4. Окно редактора коллекции строк элемента ListBox

Впишем соответствующие слова и нажмем на кнопку Ок. Введенные слова отобразятся в списке listBox1. Получим форму вида, изображенного на рис.5.

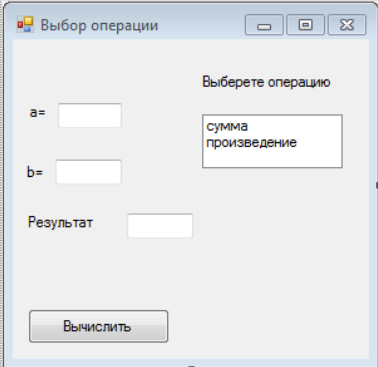


Рис.5. Форма после добавления строк в коллекцию

Теперь создадим обработчик

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

{

double a = Convert.ToDouble(textBox1.Text);

double b = Convert.ToDouble(textBox2.Text);

double c = 0;

string z = listBox1.SelectedItem.ToString();

if (z == "сумма")

c = a + b;

if (z == "произведение")

c = a \* b;

textBox3.Text = c.ToString();

}

Здесь в четвертой строке тела метода использовано свойство SelectedItem, которое соответствует выбранному члену списка. Этот член списка преобразуется в строку, и полученное значение присваивается переменной z. Затем в зависимости от значения z выполняется сложение или умножение.

Запустим приложение, В текстовые поля введем значения a и b, из списка выберем требуемую операцию и нажмем кнопку. На рис.6 показан результат при выборе произведения. Аналогично решается задача на сложение.

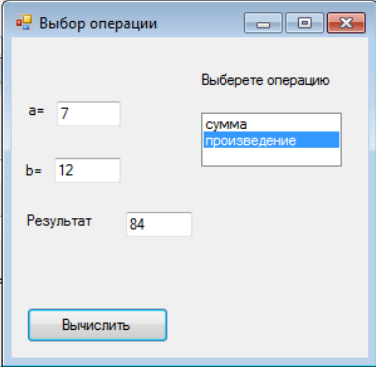


Рис.6. Пример вычисления произведения

# Разработка приложения

# 2.1Разработка сценария и алгоритма решения поставленной задачи

# 2.2Создание интерфейса приложения.

Создание пользовательского интерфейса с несколькими областями, аналогичный тому, который используется в Microsoft Outlook, со списком **папок**, областью **сообщений** и областью **предварительного просмотра**. Это расположение достигается, главным образом, путем закрепления элементов управления в форме.

При закреплении элемента управления вы определяете, к какому краю родительского контейнера прикрепляется элемент управления. Таким образом, если задать для свойства [Dock](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.splitcontainer.dock) значение [Right](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.dockstyle" \l "system-windows-forms-dockstyle-right), правый край элемента управления будет закреплен на правом краю родительского элемента управления. Кроме того, размер закрепленного края элемента управления будет меняться в соответствии с размерами контейнерного элемента управления. Дополнительные сведения о том, как работает свойство [Dock](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.splitcontainer.dock), см. в разделе [Практическое руководство. Закрепление элементов управления в формах Windows Forms](https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/how-to-dock-controls-on-windows-forms?view=netframeworkdesktop-4.8).

Эта процедура направлена на упорядочивание [SplitContainer](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.splitcontainer) и других элементов управления в форме, а не добавление функциональных возможностей для имитации приложения Microsoft Outlook.

Чтобы создать такой пользовательский интерфейс, необходимо разместить все элементы управления в элементе управления [SplitContainer](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.splitcontainer), который содержит элемент управления [TreeView](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.treeview) на панели слева. На расположенной справа панели элемента управления [SplitContainer](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.splitcontainer) находится второй элемент управления [SplitContainer](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.splitcontainer), а элемент управления [ListView](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.listview) размещен над элементом управления [RichTextBox](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.richtextbox). Эти элементы управления [SplitContainer](https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.splitcontainer) обеспечивают независимое изменение размера других элементов управления в форме. Вы можете адаптировать методы, описанные в этой процедуре, для создания собственных пользовательских интерфейсов.