Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Филиал ФГАОУ ВО «ЮУрГУ (НИУ)» в г. Златоусте

Факультет техники и технологии

Кафедра электрооборудования и автоматизации производственных процессов

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
_____ Ю.С. Сергеев
_____ 2019 г.

РАЗРАБОТКА УЧЕБНОГО ПОСОБИЯ К ПРАКТИЧЕСКИМ ЗАНЯТИЯМ ПО ДИСЦИПЛИНЕ «ПРИКЛАДНОЕ ПРОГРАММИРОВАНИЕ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ ЮУрГУ– 13.03.02.19.291.00.00 ПЗ ВКР

Консультант
Безопасность жизнедеятельности
доцент., к.т.н.
_____ С.Н. Трофимова
_____2019 г.

Руководитель работы
доцент., к.т.н.
_____С.А. Петрищев
_____2019 г.

Экономическая часть
доцент., к.т.н.
_____ С.А. Петрищев
_____2019 г.

Автор работы
студент группы ФТТ-533
_____ А.О. Ткачев
_____2019 г.

Нормоконтролер
ст. преподаватель
_____ О.В. Терентьев
_____ 2019 г.

Златоуст 2019

АННОТАЦИЯ

Ткачев А.О. Разработка учебного пособия к практическим занятиям по дисциплине «Прикладное программирование». - Златоуст: ЮУрГУ, филиал в г. Златоусте, кафедра ЭАПП, 73 с., 13 ил. Библиографический список - 18 наименований. 8 листов ф. А1.

В данной выпускной квалификационной работе произведен выбор платформы для выполнения практических занятий, разработаны практические занятия по дисциплине, созданы методические материалы для проведения практических занятий.

В работе рассматриваются такие темы, как линейные, разветвляющиеся и циклические алгоритмы, работа со строковым типом данных, одномерные и многомерные массивы, графики функций. По каждой теме рассмотрены теоретические и практические примеры, задания для самостоятельной работы.

В организационно-экономическом разделе произведен расчет всех расходов связанных с разработкой учебного пособия. Рассчитана полная себестоимость проекта на разработку учебного пособия к практическим занятиям по дисциплине «Прикладное программирование».

В разделе безопасность жизнедеятельности рассмотрены вопросы: охрана труда; противопожарная безопасность; расчет искусственного освещения.

Использование материалов работы планируется к внедрению в учебный процесс бакалавров направления 130302 «Энергетика и электротехника» путем использования современных цифровых технологий.

					13.03.02.19.291.00.00 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Ткачев А. О.			Разработка учебного пособия к практическим занятиям по дисциплине «Прикладное программирование» Пояснительная записка	Лит.	Лист	Листов
Провер.		Петрищев С.А.				Д	4	73
Т. Контр.		Вигриянов П.Г.				Филиал ФГАОУ ВО «ЮУрГУ (НИУ)» в г. Златоуст Кафедра ЭАПП		
Н. Контр.		Терентьев О.В.						
Утверд.		Сергеев Ю.С.						

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Язык C Sharp - новый язык программирования, разработанный компанией Microsoft под платформу .NET. Другие языки программирования были созданы до появления платформы .NET. Язык C Sharp специально создавался под эту платформу, и поэтому в нем отсутствуют проблемы совместимости с предыдущими версиями языка.

Разработка приложений на языке C Sharp ведется в платформе Visual Studio.Net, куда помимо C Sharp встроены такие языки программирования, как Visual Basic.NET и Visual C++.

NET Runtime - «Среда выполнения». В этой среде выполняется код, полученный после компиляции программы, написанной на C Sharp. Эта среда выполнения построена не на ассемблере (код, который является родным для процессора), а на промежуточном коде. Поэтому для данной «Среды выполнения» возможно использование нескольких языков программирования. В теории программа, написанная для этой среды, может быть выполнена любой операционной системой, в которой NET Runtime установлена, и пока для этого существует только одна ОС - Windows.

Целью выпускной квалификационной работы является повышение качества обучения бакалавров направления 130302 «Энергетика и электротехника» путем использования современных цифровых технологий.

Задачи:

- сравнение платформ и языков;
- выбор основных практических занятий;
- создание указаний к практическим занятиям;
- разработка индивидуальных заданий.

Объект: практические занятия по дисциплине «Прикладное программирование».

Предмет: указание к практическим занятиям по дисциплине «Прикладное программирование».

1 ВЫБОР ПЛАТФОРМЫ ДЛЯ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

1.1 Критерии выбора платформы

Выбор платформы представляет собой чрезвычайно сложную задачу, которая состоит из двух частей:

- определение сервиса, который должен обеспечиваться платформой;
- определение уровня сервиса, который может обеспечить данная платформа.

Существует несколько причин, в силу которых достаточно сложно оценить возможности платформы с выбранным набором компонентов, которые включаются в систему:

- подобная оценка прогнозирует будущее: предполагаемую комбинацию устройств, будущее использование программного обеспечения, будущих пользователей;
- конфигурация аппаратных и программных средств связана с определением множества разнородных по своей сути компонентов системы, в результате чего сложность быстро увеличивается;
- скорость технологических усовершенствований аппаратных средств, функциональной организации системы, операционных систем очень высокая и постоянно растет. Ко времени, когда какой-либо компонент широко используется и хорошо изучен, он часто рассматривается как устаревший;
- доступная потребителю информация об аппаратном обеспечении, операционных системах, программном обеспечении носит общий характер. Структура аппаратных средств, на базе которых работают программные системы, стала настолько сложной, что эксперты в одной области редко являются таковыми в другой.[9]

Выбор платформы определяется рядом критериев. К ним относятся:

- отношение стоимость-производительность;
- надежность и отказоустойчивость;
- масштабируемость;
- совместимость и мобильность программного обеспечения.

1.2. Отношение стоимость-производительность

Появление любого нового направления в вычислительной технике определяется требованиями компьютерного рынка. Поэтому у разработчиков компьютеров нет одной единственной цели. Суперкомпьютеры стоят дорого, т. к. для достижения поставленных целей при проектировании высокопроизводительных конструкций приходится игнорировать стоимостные характеристики. Другим крайним примером может служить низкостоимостная конструкция, где производительность принесена в жертву для достижения низкой стоимости. К этому направлению относятся персональные компьютеры. Между этими двумя крайни-

ми направлениями находятся конструкции, основанные на отношении стоимость-производительность, в которых разработчики находят баланс между стоимостными параметрами и производительностью. Типичными примерами такого рода компьютеров являются мини-компьютеры и рабочие станции.

1.3. Надежность и отказоустойчивость

Самой главной характеристикой платформы является надежность. Повышение надежности основано на принципе предотвращения неисправностей путем снижения интенсивности отказов и сбоев за счет применения электронных схем и компонентов с высокой и сверхвысокой степенью интеграции, снижения уровня помех, облегченных режимов работы схем, обеспечение тепловых режимов их работы, а также за счет совершенствования методов сборки аппаратной части персонального компьютера.

Введение отказоустойчивости требует избыточного аппаратного и программного обеспечения. Структура многопроцессорных и многомашинных систем приспособлена к автоматической реконфигурации и обеспечивает возможность продолжения работы системы после возникновения неисправностей. Понятие надежности включает не только аппаратные средства, но и программное обеспечение. Главной целью повышения надежности систем является целостность хранимых в них данных.

1.4. Масштабируемость

Масштабируемость должна обеспечиваться архитектурой и конструкцией компьютера, а также соответствующими средствами программного обеспечения.

Добавление каждого нового процессора в действительно масштабируемой системе должно давать прогнозируемое увеличение производительности и пропускной способности при приемлемых затратах. В действительности реальное увеличение производительности трудно оценить заранее, поскольку оно в значительной степени зависит от динамики поведения прикладных задач.

Возможность масштабирования системы определяется не только архитектурой аппаратных средств, но и зависит от заложенных свойств программного обеспечения. Простой переход, например, на более мощный процессор может привести к перегрузке других компонентов системы. Это означает, что действительно масштабируемая система должна быть сбалансирована по всем параметрам.

1.5. Совместимость и мобильность программного обеспечения

В настоящее время одним из наиболее важных факторов, определяющих современные тенденции в развитии информационных технологий, является ориентация компаний-поставщиков компьютерного оборудования на рынок прикладных программных средств. Это объясняется, прежде всего, тем, что для конечного пользователя, в конце концов, важно программное обеспечение, позволяющее

решить его задачи, а не выбор той или иной аппаратной платформы. Переход от однородных сетей программно-совместимых компьютеров к построению неоднородных сетей, включающих компьютеры разных фирм-производителей, в корне изменил и точку зрения на саму сеть: из сравнительно простого средства обмена информацией она превратилась в средство интеграции отдельных ресурсов-мощную распределенную вычислительную систему, каждый элемент которой лучше всего соответствует требованиям конкретной прикладной задачи. Этот переход выдвинул ряд новых требований:

Во-первых, такая вычислительная среда должна позволять гибко менять количество и состав аппаратных средств и программного обеспечения в соответствии с меняющимися требованиями решаемых задач.

Во-вторых, она должна обеспечивать возможность запуска одних и тех же программных систем на различных аппаратных платформах, т. е. обеспечивать мобильность программного обеспечения.

В-третьих, эта среда должна гарантировать возможность применения одних и тех же человеко-машинных интерфейсов на всех компьютерах, входящих в неоднородную сеть.

1.6. Языки программирования

Язык программирования - это система обозначений, служащая для точного описания программ или алгоритмов для ЭВМ. Языки программирования являются искусственными языками. От естественных языков они отличаются ограниченным числом “слов” и очень строгими правилами записи команд (операторов). Поэтому при применении их по назначению они не допускают свободного толкования выражений, характерного для естественного языка.

Можно сформулировать ряд требований к языкам программирования и классифицировать языки по их особенностям.

Основные требования, предъявляемые к языкам программирования:

- наглядность - использование в языке по возможности уже существующих символов, хорошо известных и понятных как программистам, так и пользователям ЭВМ;

- единство - использование одних и тех же символов для обозначения одних и тех же или родственных понятий в разных частях алгоритма. Количество этих символов должно быть по возможности минимальным;

- гибкость - возможность относительно удобного, несложного описания распространенных приемов математических вычислений, с помощью имеющегося в языке ограниченного набора изобразительных средств;

- модульность - возможность описания сложных алгоритмов в виде совокупности простых модулей, которые могут быть составлены отдельно и использованы в различных сложных алгоритмах;

- однозначность - недвусмысленность записи любого алгоритма. Отсутствие ее могло бы привести к неправильным ответам при решении задач.

Объектно-ориентированные языки (Object Pascal, C++, Java, Objective Caml. и др.). Руководящая идея объектно-ориентированных языков заключается в стремлении связать данные с обрабатывающими эти данные процедурами в единое целое - объект.

Объектно-ориентированный подход использует следующие базовые понятия:

- объект;
- свойство объекта;
- метод обработки;
- событие;
- класс объектов.

Объект - совокупность свойств (параметров) определенных сущностей и методов их обработки (программных средств).

Свойство - это характеристика объекта и его параметров. Все объекты наделены определенными свойствами, совокупность которых выделяют (определяют) объект.

Метод - это набор действий над объектом или его свойствами.

Событие - это характеристика изменения состояния объекта.

Класс - это совокупность объектов, характеризующихся общностью применяемых к ним методов обработки или свойств.

C# (C Sharp) - «Си Шарп»: объектно-ориентированный язык программирования, о разработке которого в 2000 г. объявила фирма Microsoft . По своему характеру он напоминает языки C++ и Java и предназначен для разработчиков программ, использующих языки C и C++ для того, чтобы они могли более эффективно создавать Интернет - приложения. Указывается, что C Sharp будет тесно интегрирован с языком XML.

Delphi - является «наследником» языка Паскаль; основные операторы в этих языках одинаковы. Но Delphi имеет средство для работы с различными графическими объектами (создания форм, кнопок, меню), а также для обработки сложных структур данных. Поэтому он очень популярен при разработке различных Windows- приложений.

Паскаль - процедурно-ориентированный язык программирования высокого уровня, разработанный в конце 1960-х гг. Никлаусом Виртом, первоначально для обучения программированию в университетах. Назван в честь французского математика XVII века Блеза Паскаля. В своей начальной версии Паскаль имел довольно ограниченные возможности, поскольку предназначался для учебных целей, однако последующие его доработки позволили сделать его хорошим универсальным языком, широко используемым, в том числе для написания больших и сложных программ.

Бейсик (Basic) - язык программирования высокого уровня. Первоначально предназначался для обучения программированию. Отличается простотой, легко усваивается начинающими программистами благодаря наличию упрощенных конструкций языка Фортран и встроенных математических функций, алгоритмов и операторов. Существует множество различных версий Бейсика, которые не

полностью совместимы друг с другом. Некоторые реализации Бейсика включают средства обработки данных и наборов данных. Большинство версий Бейсика используют интерпретатор, который преобразует его компоненты в машинный код и позволяет запускать программы без промежуточной трансляции. Некоторые более совершенные версии Бейсика позволяют использовать для этой цели трансляторы.

Java - объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем, приобретённой компанией Oracle). Приложения Java обычно компилируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине (JVM) независимо от компьютерной архитектуры. Для семи разных задач время выполнения на Java составляет в среднем в полтора-два раза больше, чем для C/C++, в некоторых случаях Java быстрее, а в отдельных случаях в 7 раз медленнее. С другой стороны, для большинства из них потребление памяти Java-машиной было в 10-30 раз больше, чем программой на C/C++. Также примечательно исследование, проведённое компанией Google, согласно которому отмечается существенно более низкая производительность и большее потребление памяти в тестовых примерах на Java в сравнении с аналогичными программами на C++. Программы, написанные на Java, имеют репутацию более медленных и занимающих больше оперативной памяти, чем написанные на языке C Sharp.[13]

1.7. Обоснование выбора платформы и языка

Языков программирования на сегодняшний день очень много, все они разные и предназначены для решения различных задач. C Sharp отлично подходит для быстрого написания настольных приложений с удобным интерфейсом. Кроме того, он относится к одному из языков технологии ASP.NET для разработки веб-приложений. Он востребован и перспективен, отлично подходит для того, чтобы с него начинать изучение программирования.

На сегодняшний день Windows Forms все еще остается платформой для многих бизнес-приложений, ориентированных на работу с данными. Довольно часто в приложениях можно встретить формы, которые предназначены для ввода или редактирования объектов с большим количеством зависимых свойств.

В Windows Forms имеется множество возможностей, которые упрощают и ускоряют реализацию общих задач, таких как создание диалоговых окон, печать, добавление справки и документации, а также локализация приложений на различных языках. Кроме того, в Windows Forms применяется эффективная система безопасности .NET Framework. Благодаря ней можно создавать более надежные приложения.

Базовый курс программирования, построенный на основе языка C Sharp, позволит студентам быстрее стать востребованными специалистами-профессионалами.

Предметной областью решаемой задачи является разработка проектов Windows Forms в языке C Sharp.

Язык программирования C Sharp в настоящее время является одним из наиболее распространенных средств разработки в рамках объектно-ориентированного подхода.

Язык C Sharp, как средство обучения программированию, обладает рядом несомненных достоинств. Он хорошо организован, строг, большинство его конструкций логичны и удобны. Развитые средства диагностики и редактирования кода делают процесс программирования приятным и эффективным. Мощная библиотека классов платформы .NET берет на себя массу рутинных операций, что дает возможность решать более сложные задачи, используя готовые «строительные блоки». Все это позволяет расценивать C Sharp как перспективную замену языков Паскаль, BASIC и C при обучении программированию. [15]

Вывод раздела один:

- среда разработки Visual Studio.NET предоставляет мощные и удобные средства написания, корректировки, компиляции, отладки и запуска приложений, использующих .NET-совместимые языки;

- приложение в процессе разработки называется проектом. Проект объединяет все необходимое для создания приложения: файлы, папки, ссылки и прочие ресурсы. Среда Visual Studio.NET позволяет создавать проекты различных типов.

2 РАЗРАБОТКА ПРАКТИЧЕСКИХ ЗАНЯТИЙ ПО ДИСЦИПЛИНЕ

2.1. Структура приложения C Sharp

Проект создается перед началом программирования. Он содержит все исходные материалы для приложения, такие как файлы исходного кода, ресурсов, значки, ссылки на внешние файлы, на которые опирается программа, и данные конфигурации, такие как параметры компилятора.

Часто используется более глобальное понятие - решение (solution). Оно содержит один или несколько проектов, один из которых может быть указан как стартовый проект. Исполнение решения начинается со стартового проекта.

При создании простейшей C Sharp программы в Visual Studio создается папка решения, в которой для каждого проекта создается подпапка проекта, а уже в ней - другие подпапки с результатами компиляции приложения.

Проект - это основная единица, с которой работает студент. При создании проекта можно выбрать его тип, а Visual Studio создаст каркас проекта в соответствии с выбранным типом.

По своим «внешним» проявлениям консольные напоминают приложения DOS, запущенные в Windows. Тем не менее, это настоящие Win32-приложения, которые под DOS работать не будут. Для консольных приложений доступен Win32 API, а кроме того, они могут использовать консоль - окно, предоставляемое системой, которое работает в текстовом режиме и в которое можно вводить данные с клавиатуры. Особенность консольных приложений в том, что они работают не в графическом, а в текстовом режиме.

Проект в Visual Studio состоит из файла проекта (файл с расширением .csproj), одного или нескольких файлов исходного текста (с расширением .cs), файлов с описанием окон формы (с расширением. designer.cs), файлов ресурсов (с расширением .resx), а также ряда служебных файлах.

В файле проекта находится информация о модулях, составляющих данный проект, входящих в него ресурсах, а также параметров построения программы. Файл проекта автоматически создается и изменяется средой Visual Studio и не предназначен для ручного редактирования.

Файл исходного текста - программный модуль, предназначен для размещения текстов программ. В этом файле студент размещает текст программы, написанный на языке C Sharp. Модуль имеет структуру, представленную на рисунке 2.1.

```
// Раздел подключенных пространств имен
using System;
// Пространство имен нашего проекта
name space MyFirstApp
  {
   // Класс окна
   public partial class Form1 : Form
       {
       // Методы окна
       public Form1()
           {
            Initialize Component();
            }
        }
   }
```

Рисунок 2.1 - Структура программы

В разделе подключения пространств имен (каждая строка которого располагается в начале файла и начинается ключевым словом using) описываются используемые пространства имен. Каждое пространство имен включает в себя классы, выполняющие определенную работу, например, классы для работы с сетью располагаются в пространстве System.Net, а для работы с файлами - в System.IO. Большая часть пространств, которые используются в обычных проектах, уже подключена при создании нового проекта, но при необходимости можно дописать дополнительные пространства имен.

Чтобы не происходило конфликтов имен классов и переменных, классы нашего проекта также помещаются в отдельное пространство имен. Определяется оно ключевым словом namespace, после которого следует имя пространства (обычно оно совпадает с именем проекта).

Внутри пространства имен помещаются наши классы - в новом проекте это класс окна, который содержит все методы для управления
поведением окна. Нужно обратить внимание, что в определении класса присутствует ключевое слово partial, это говорит о том, что в исходном тексте представлена только часть класса, с которой мы работаем непосредственно, а служебные методы для обслуживания окна скрыты в другом модуле (при желании их тоже можно посмотреть, но редактировать вручную не рекомендуется).

Внутри класса располагаются переменные, методы и другие элементы программы. Фактически, основная часть программы размещается внутри класса при создании обработчиков событий.

При компиляции программы Visual Studio создает исполняемые exe-файлы в каталоге bin.[11]

2.1.1 Работа с проектом

Проект в Visual Studio состоит из многих файлов, и создание сложной программы требует хранения каждого проекта в отдельной папке. При создании нового

проекта Visual Studio по умолчанию сохраняет его в отдельной папке. Желательно создать для себя свою папку со своей фамилией внутри папки своей группы, чтобы все проекты хранились в одном месте. После этого можно запускать Visual Studio и создавать новый проект.

После создания проекта сохранить его в подготовленной папке: Файл → Сохранить все. При внесении значительных изменений в проект следует еще раз сохранить проект той же командой, а перед запуском программы на выполнение среда обычно сама сохраняет проект на случай какого-либо сбоя. Для открытия существующего проекта используется команда Файл → Открыть проект, или можно найти в папке файл проекта с расширением .sln и сделать на нем двойной щелчок.

2.2 Описание данных

Особенное значение в C Sharp, имеют типы данных поскольку это строго типизированный язык. Это значит, что все операции подвергаются строгому контролю со стороны компилятора на соответствие типов, причем недопустимые операции не компилируются. Строгая проверка типов позволяет предотвратить ошибки и повысить надежность программ. Для обеспечения контроля типов все переменные, выражения и значения должны принадлежать к определенному типу. Такого понятия, как бестиповая переменная, допустимая в ряде скриптовых языков, в C Sharp вообще не существует. Более того, тип значения определяет те операции, которые разрешается выполнять над ним. Операция, разрешенная для одного типа данных, может оказаться недопустимой для другого.В таблице 2.1 представлена структура тип данных.

Таблица 2.1 - Структура тип данных

Типы данных				
Числовые		Символьные	Логические	Прочие
Целочисленные byte, sbyte ushort,short uint, int ulong,long	С плавающей точкой float double decimal	char string	bool	object

Целочисленные типы. В C Sharp определены девять целочисленных типов: char, byte, sbyte, short, ushort, int, uint, long и ulong. Тип char может хранить числа, но чаще используется для представления символов. Остальные восемь целочисленных типов предназначены для числовых расчетов.

Некоторые целочисленные типы могут хранить как положительные, так и отрицательные значения (sbyte, short, int и long), другие же - только положительные (char, byte, ushort, uint и ulong).

Типы с плавающей точкой. Такие типы позволяют представлять числа с дробной частью. В C Sharp имеются три разновидности типов данных с плавающей точкой: float, double и decimal. Первые два типа представляют числовые значения с одинарной и двойной точностью, вычисления над ними выполняются аппаратно и поэтому быстро. Тип decimal служит для представления чисел с плавающей точкой высокой точности без округления, характерного для типов float и double. Вычисления с использованием этого типа выполняются программно и поэтому более медленны.

Числа, входящие в выражения, C Sharp по умолчанию считает целочисленными. Поэтому следующее выражение будет иметь значение 0, ведь если 3 нацело разделить на 4, то получится как раз 0:

```
double x = 3 / 4;
```

Чтобы этого не происходило, в подобных случаях нужно явно указывать тип чисел с помощью символов - модификаторов: f для float и d для double. Приведенный выше пример правильно будет выглядеть так:

```
double x = 3d / 4d;
```

Иногда в программе возникает необходимость записать числа в экспоненциальной форме. Для этого после мантиссы числа записывается символ «e» и сразу после него - порядок. Например, число $3{,}5 \cdot 10^{-3}$ будет записано в программе следующим образом:

```
3.5e-3
```

Символьные типы. В C Sharp символы представлены не 8 - разрядным кодом, как во многих других языках программирования, а 16 - разрядным кодом, который называется юникодом (Unicode). В юникоде набор символов представлен настолько широко, что он охватывает символы практически из всех естественных языков на свете.

Основным типом при работе со строками является тип string, задающий строки переменной длины. Тип string представляет последовательность из нуля или более символов в кодировке Юникод. Хранение текста происходит в виде последовательной доступной только для чтения коллекции объектов char.

Логический тип данных. Тип bool представляет два логических значения: «истина» и «ложь». Эти логические значения обозначаются в C Sharp зарезервированными словами true и false соответственно. Следовательно, переменная или выражение типа bool будет принимать одно из этих логических значений.[10]

Самыми популярными данными являются - переменные и константы. Переменная - это ячейка памяти, которой присвоено некоторое имя, и это имя используется для доступа к данным, расположенным в данной ячейке. Для каждой переменной задается тип данных - диапазон всех возможных значений для данной переменной. Объявляются переменные непосредственно в тексте программы. Лучше всего

сразу присвоить им начальное значение с помощью знака присвоения «=» (переменная = значение):

```
int x; // Только объявление
int y = 9; // Объявление и инициализация
```

Для того чтобы присвоить значение символьной переменной, достаточно заключить это значение (т. е. символ) в одинарные кавычки:

```
char ch;    // Только объявление
char symbol = 'u'; // Объявление и инициализация
```

Частным случаем переменных являются константы. Константы - это переменные, значения которых не меняются в процессе выполнения программы. Константы описываются как обычная переменная, только с ключевым словом const впереди:[13]

```
const int a = 15;
```

2.3 Ввод/вывод данных в программу

Рассмотрим один из способов ввода данных через элементы, размещенные на форме. TextBox - используют для ввода данных, через обращение к его свойству Text - хранит в себе строку введенных символов. Поэтому данные можно считать таким образом:

```
private void button2_Click(object sender,EventArgs c)
{
    string z = textBox2.Text;
}
```

Однако со строкой символов трудно производить арифметические операции, поэтому лучше всего при вводе числовых данных перевести строку в целое или вещественное число. Для этого у типов int и double существуют методы Parse для преобразования строк в числа. С этими числами можно производить различные арифметические действия. Таким образом, предыдущий пример можно переделать следующим образом:

```
private void button2_Click(object sender, EventArgs c)
{
    string z = textBox2.Text;
    int x = int.Parse(s);
    int y = x*x;
}
```

В языках программирования в дробных числах чаще всего используется точка, например: «29.7». Однако в C Sharp методы преобразования строк в числа (вроде double.Parse() или Convert.ToFloat()) учитывают региональные настройки Windows, в которых в качестве десятичной точки используется символ запятой (например,«29,7»). Поэтому в полях TextBox в формах следует вводить дробные числа с запятой, а не с точкой. В противном случае преобразование не выполнится, а программа остановится с ошибкой.[12]

Перед выводом числовые данные следует преобразовать назад в строку. Для этого у каждой переменной существует метод ToString(), который возвращает в результате строку с символьным представлением значения. Вывод данных можно осуществлять в элементы TextBox или Label, используя свойство Text. Например:

```
private void button7_Click(object sender,EventArgs y)
{
    string s = textBox1.Text;
    int x = int. Parse(z)
    int y = b*b;
    labe77.Text = b.ToString();
}
```

2.4 Арифметические действия и стандартные функции

При вычислении выражения, стоящего в правой части оператора присвоения, могут использоваться арифметические операции:

1) умножение (*);
2) сложение (+);
3) вычитание (–);
4) деление (/);
5) остаток от деления (%).

Для задания важных операций могут использоваться круглые скобки ( ), а также могут использоваться стандартные математические функции, представленные методами класса Math:

- Math.Sin(a)– вычислить синус;
- Math.Sinh(a)– вычислить гиперболический синус;
- Math.Cos(a) – вычислить косинус (аргумент задается в радианах);
- Math.Atan(a) – вычислить арктангенс (аргумент задается в радианах);
- Math.Log(a)– вычислить натуральный логарифм;
- Math.Exp(a)– вычислить экспоненту;
- Math.Pow(x, y) – вычисление переменной x в степень y;
- Math.Sqrt(a)– извлечение квадратного корня;
- Math.Abs(a)– вычисление модуль числа;
- Math.Truncate(a)– вычисление целой части числа;
- Math.Round(a)– округление числа.

В тригонометрических функциях все аргументы задаются в радианах.

2.4.1 Логические переменные и операции над ними

Переменные логического типа описываются посредством служебного слова bool. Они могут принимать только два значения - False (ложь) и True (истина). Результат False (ложь) и True (истина) возникает при использовании операций сравнения > (больше), < (меньше), ! = (не равно), >= (больше либо равно), <= (меньше либо равно), = = (равно).

Описываются логические переменные следующим образом:

```
bool z;
```

В языке C Sharp имеются логические операции, применяемые к переменным логического типа. Это операции логического отрицания (!), логическое И (&&) и логическое ИЛИ (||). Операция логического отрицания является унарной операцией. Результат операции ! есть False, если операнд истинен, и True, если операнд имеет значение «ложь». Так,

!True → False (неправда есть ложь),

!False → True (не ложь есть правда).

Результат операции логическое И (&&) есть истина, только если оба ее операнда истинны, и ложь во всех других случаях. Результат операции логическое ИЛИ (||) есть истина, если какой-либо из ее операндов истинен, и ложен только тогда, когда оба операнда ложны.[10]

2.5 Условные операторы

Операторы ветвления позволяют изменить порядок выполнения операторов в программе. К операторам ветвления относятся условный оператор if и оператор выбора switch.

Условный оператор if используется для разветвления процесса обработки данных на два направления. Он может иметь одну из форм: сокращенную или полную.

Форма сокращенного оператора if:

```
if (A) V;
```

где A - логическое или арифметическое выражение, истинность которого проверяется; V - оператор.

При выполнении сокращенной формы оператора if сначала вычисляется выражение A, затем проводится анализ его результата: если V истинно, то выполняется оператор V формы оператора if можно либо выполнить оператор V, либо пропустить его.

Форма полного оператора if:

```
if (A) V1; else V2;
```

где A - логическое или арифметическое выражение, истинность которого проверяется; V1, V2 - операторы.

При выполнении полной формы оператора if сначала вычисляется выражение A, затем анализируется его результат: если A истинно, то выполняется оператор V1, а оператор V2 пропускается; если A ложно, то выполняется оператор V2, а V1 - пропускается. Таким образом, с помощью полной формы оператора if можно выбрать одно из двух альтернативных действий процесса обработки данных.

Для примера вычислим значение функции:

$$y(x) = \begin{cases} \sin(x), \text{ если } x \le a \\ \cos(x), \text{ если } a < x < b \\ \operatorname{tg}(x), \text{ если } x \ge b \end{cases}$$

Указанное выражение может быть запрограммировано в виде:

```
                if (a < = b)
              y = Math.Sin(x);
if ((a > b) && (a < b)) y = Math.Cos(x);
                if (a > = b)
           y = Math.Sin(x) / Math.Cos(x);
```

или с помощью оператора else:

```
                if (a < = b)
              y = Math.Sin(x);
                    else
                  if(a < b)
              y = Math.Cos(x);
                    else
           y = Math.Sin(x) / Math.Cos(x);
```

В Си - подобных языках программирования, к которым относится и C Sharp, операция сравнения представляется двумя знаками равенства, например:

```
if (a == b)
```

Оператор выбора switch предназначен для разветвления процесса вычислений по нескольким направлениям. Формат оператора:

```
switch (<выражение>)
                {
case <константное_выражение_4>:
          [<оператор 4>];
        <оператор перехода>;
case <константное_выражение_4>: [<оператор 4>];
```

```
                <оператор перехода>;
                        ...
        case <константное_выражение_m>: [<оператор m>];
                <оператор перехода>; [default:
                        <оператор>;]
                        }
```

Выражение, записанное в квадратных скобках, является необязательным элементом в операторе switch. Если оно отсутствует, то может отсутствовать и оператор перехода.

Выражение, стоящее за ключевым словом switch, должно иметь арифметический, символьный или строковый тип. Все константные выражения должны иметь разные значения, но их тип должен совпадать с типом выражения, стоящим после switch или приводиться к нему. Ключевое слово case и расположенное после него константное выражение называют также меткой case.

Выполнение оператора начинается с вычисления выражения, расположенного за ключевым словом switch. Полученный результат сравнивается с меткой case. Если результат выражения соответствует метке case, то выполняется оператор, стоящий после этой метки, за которым обязательно должен следовать оператор перехода: break, goto, return и т. д. При использовании оператора break происходит выход из switch и управление передается оператору, следующему за switch. Если же используется оператор goto, то управление передается оператору, помеченному меткой, стоящей после goto. Оператор return завершает выполнение текущего метода.[14]

Если ни одно выражение case не совпадает со значением оператора switch, управление передается операторам, следующим за необязательной подписью default. Если подписи default нет, то управление передается за пределы оператора switch.

Пример использования оператора switch:

```
int dayOfWeek = 7;
switch (dayOfWeek)
        {
        case 1:
MessageBox.Show("Monday");
 break;
        case 2:
MessageBox.Show("Tuesday");
 break;
        case 3:
MessageBox.Show("Wednesday");
 break;
        case 4:
MessageBox.Show("Thursday");
 break;
        case 5:

MessageBox.Show("Friday");
```

```
                break;
                        case 6:
                Message-
Box.Show("Saturday"); break;
                        case 7:
        MessageBox.Show("sunday");
                break;
                        default:
    MessageBox.Show("Неверное значение!");
                    break;
                        }
```

Поскольку на момент выполнения оператора switch в этом примере переменная dayOfWeek равнялась 7, то будут выполнены операторы из блока case 7.

В ряде случаев оператор switch можно заменить несколькими операторами if, однако не всегда такая замена будет легче для чтения. Например, приведенный выше пример можно переписать так:

```
int dayOfWeek = 7;

if (dayOfWeek ==1)
MessageBox.Show("Monday");
        else
    if (dayOfWeek == 2)
        else
MessageBox.Show("Tuesday");
        else
    if (dayOfWeek == 3)
MessageBox.Show("Wednesday");
        else
    if (dayOfWeek == 4)
MessageBox.Show("Thursday");
        else
    if (dayOfWeek == 5)
 MessageBox.Show("Friday");
        else
    if (dayOfWeek == 6)
MessageBox.Show("Saturday");
        else
    if (dayOfWeek == 7)
 MessageBox.Show("sunday");
        else
MessageBox.Show("Неверное значение!");
```

2.5.1 Кнопки - переключатели

При создании программ в Visual Studio для организации разветвлений часто используются элементы управления в виде кнопок - переключателей (RadioButton). Состояние такой кнопки (включено - выключено) визуально отражается на форме, а в программе можно узнать его с помощью свойства Checked: если кнопка

включена, это свойство будет содержать True, в противном случае False. Если пользователь выбирает один из вариантов переключателя в группе, все остальные автоматически отключаются.

Группируются радиокнопки с помощью какого-либо контейнера - часто это бывает элемент GroupBox. Радиокнопки, размещенные в разных контейнерах, образуют независимые группы, представленные на рисунке 2.2.

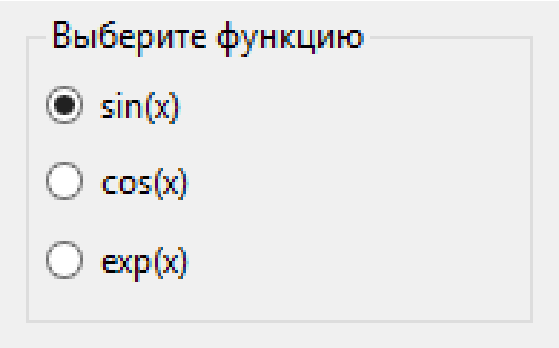


Рисунок 2.2 - Группа радиокнопок

```
if (radioButton5.Checked)
MessageBox.Show("Выбрана функция: синус");
else
if (radioButton4.Checked)
MessageBox.Show("Выбрана функция: косинус");
else
if (radioButton5.Checked)
MessageBox.Show("Выбрана функция: экспонента");
```

2.6 Операторы организации циклов

Под циклом понимается многократное выполнение одних и тех же операторов при различных значениях промежуточных данных. Число повторений может быть задано в явной или неявной форме.

К операторам цикла относятся: цикл с предусловием while, цикл с постусловием do while, цикл с параметром for и цикл перебора foreach. Рассмотрим некоторые из них.

2.6.1 Цикл с предусловием

Оператор цикла while организует выполнение одного оператора (простого или составного) неизвестное заранее число раз. Формат цикла while:

```
while (X) Y;
```

где X - выражение, истинность которого проверяется (условие завершения цикла); Y - тело цикла - оператор (простой или составной).

Перед каждым выполнением тела цикла анализируется значение выражения X: если оно истинно, то выполняется тело цикла, и управление передается на повторную проверку условия X; если значение X ложно - цикл завершается и управление передается на оператор, следующий за оператором Y.

Если результат выражения X окажется ложным при первой проверке, то тело цикла не выполнится ни разу. Отметим, что если условие X во время работы цикла не будет изменяться, то возможна ситуация зацикливания, то есть невозможность выхода из цикла. Поэтому внутри тела должны находиться операторы, приводящие к изменению значения
выражения B так, чтобы цикл мог корректно завершиться.

В качестве иллюстрации выполнения цикла while рассмотрим программу вывода целых чисел от 1 до n по нажатию кнопки на форме:

```
private void button1_Click(object sender, EventArgs e)
                                {
        int m  = 10;     // Количество повторений цикла
            int u  = 2;       // Начальное значение
          while (u <= m) // Пока u меньше или равно  m
                                {
    MessageBox.Show(i.ToString()); // Показываем u
                u++; // Увеличиваем u на 2
                                }
                                }
```

2.6.2 Цикл с постусловием

Оператор цикла do while также организует выполнение одного оператора (простого или составного) неизвестное заранее число раз. Однако в отличие от цикла while условие завершения цикла проверяется после выполнения тела цикла. Формат цикла do while:

```
do S while (B);
```

где B - выражение, истинность которого проверяется (условие завершения цикла); S - тело цикла - оператор (простой или блок).

Сначала выполняется оператор S, а затем анализируется значение выражения B: если оно истинно, то управление передается оператору S, если
ложно - цикл завершается, и управление передается на оператор, следующий за условием B. Так как условие B проверяется после выполнения тела цикла, то в любом случае тело цикла выполнится хотя бы один раз.[11]

В операторе do while, так же как и в операторе while, возможна ситуация зацикливания в случае, если условие Bвсегда будет оставаться истинным.

2.6.3 Цикл с параметром

Цикл с параметром имеет следующую структуру:

```
for (<инициализация>; <выражение>; <модификация>)
                    <оператор>;
```

Инициализация используется для объявления и/или присвоения начальных значений величинам, используемым в цикле в качестве параметров (счетчиков). В этой части можно записать несколько операторов, разделенных запятой. Областью действия переменных, объявленных в части инициализации цикла, является цикл и вложенные блоки. Инициализация выполняется один раз в начале исполнения цикла.

Выражение определяет условие выполнения цикла: если его результат истинен, цикл выполняется. Истинность выражения проверяется перед каждым выполнением тела цикла, таким образом, цикл с параметром реализован как цикл с предусловием. В блоке выражение через запятую можно записать несколько логических выражений, тогда запятая равносильна операции логическое И (&&).

Модификация выполняется после каждой итерации цикла и служит обычно для изменения параметров цикла. В части модификация можно записать несколько операторов через запятую.

Оператор (простой или составной) представляет собой тело цикла. Любая из частей оператора for (инициализация, выражение, модификация, оператор) может
отсутствовать, но точку с запятой, определяющую позицию пропускаемой части, надо оставить

Пример формирования строки, состоящей из чисел от 0 до 9, разделенных пробелами:

```
string y = ""; // Инициализация строки
for (int t = 0; t <= 15; t++) // Перечисление всех чисел
y += t.ToString() + " "; // Добавляем число и пробел
MessageBox.Show(y.ToString()); // Показываем результат
```

Данный пример работает следующим образом. Сначала вычисляется начальное значение переменной t. Затем, пока значение t меньше или равно 15, выполняется тело цикла, и затем повторно вычисляется значение t Когда значение t становится больше 15, условие - ложно и управление передается за пределы цикла.[15]

2.6.4 Средства отладки программ

Практически в каждой написанной программе после запуска обнаруживаются ошибки.

Ошибки первого уровня (ошибки компиляции) связаны с неправильной записью операторов (орфографические, синтаксические). При обнаружении ошибок компилятор формирует список, который отображается по завершению компиляции (рисунок 2.3). При этом возможен только запуск программы, которая была успешно скомпилирована для предыдущей версии программы.

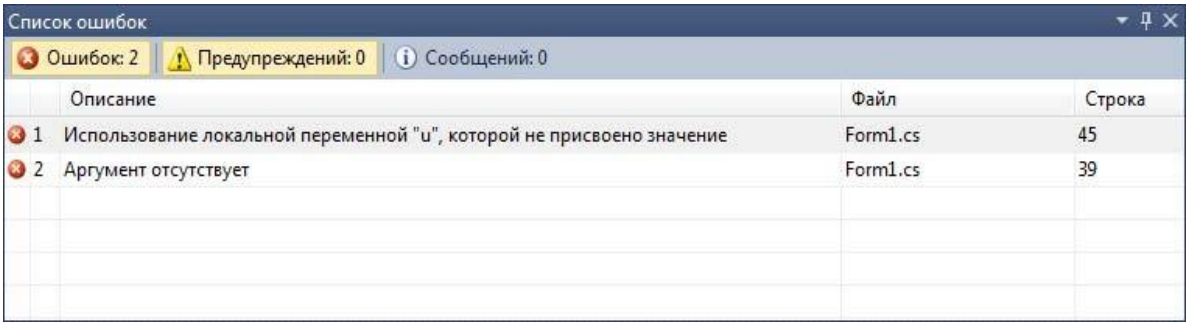


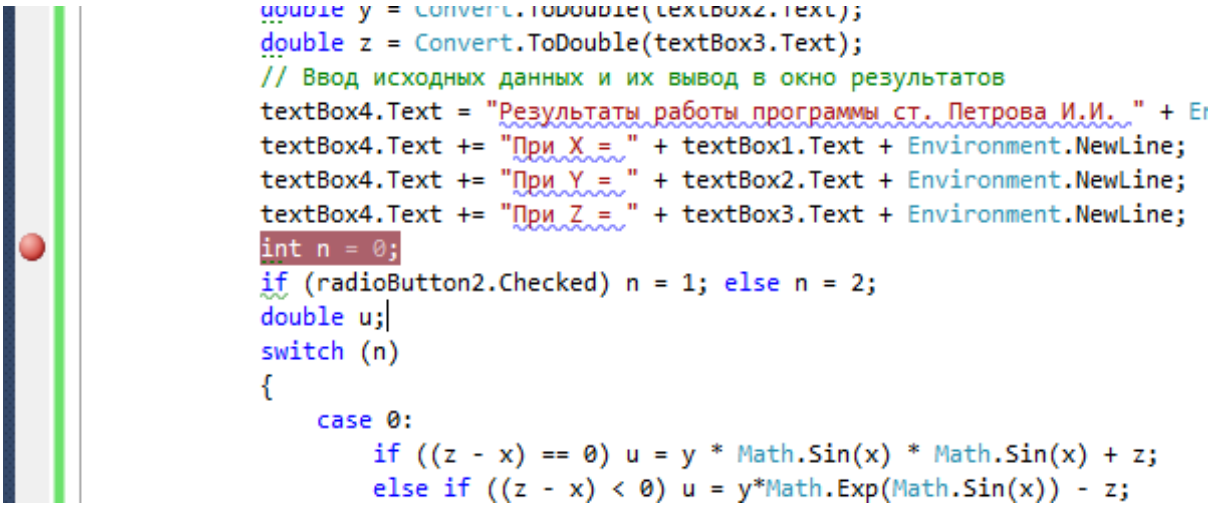
Рисунок.2.3 - Окно со списком ошибок компиляции

При выявлении ошибок компиляции в нижней части экрана появляется текстовое окно (смотреть рисунок 2.3), содержащее сведения обо всех ошибках, найденных в проекте. Каждая строка этого окна содержит имя файла, в котором найдена ошибка, номер строки с ошибкой и характер ошибки. Для быстрого перехода к интересующей ошибке необходимо дважды щелкнуть на строке с ее описанием. Следует обратить внимание на то, что одна ошибка может повлечь за собой другие, которые исчезнут при ее исправлении. Поэтому необходимо исправлять их последовательно, сверху вниз и, после исправления каждой - компилировать программу снова.

Ошибки второго уровня (ошибки выполнения) связаны с ошибками выбранного алгоритма решения или с неправильной программной реализацией алгоритма. Эти ошибки проявляются в том, что результат расчета оказывается неверным либо происходит переполнение, деление на ноль и др. Поэтому перед использованием отлаженной программы ее надо протестировать, т. е. сделать просчеты при таких комбинациях исходных данных, для которых заранее известен результат. Если тестовые расчеты указывают на ошибку, то для ее поиска следует использовать встроенные средства отладки среды программирования.

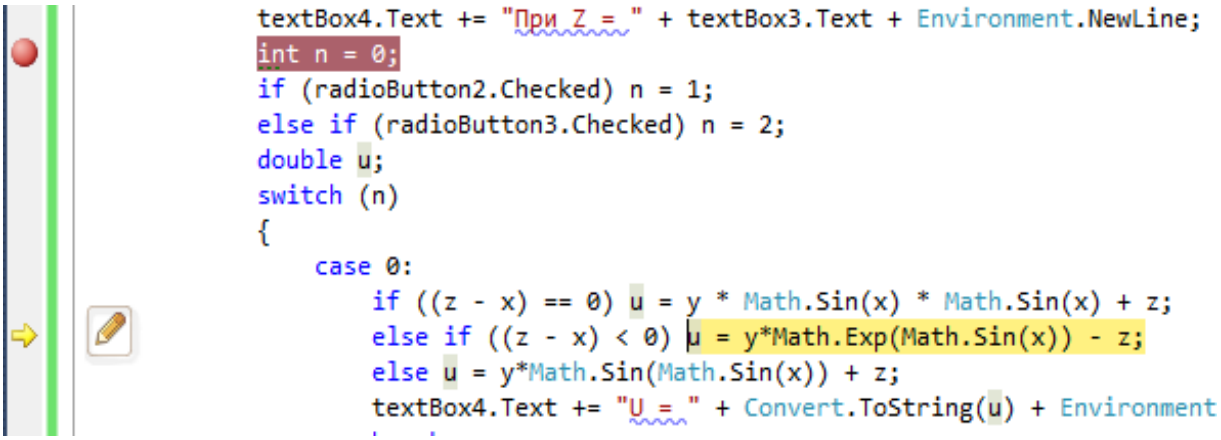
В простейшем случае для локализации места ошибки рекомендуется поступать следующим образом. В окне редактирования текста установить точку останова перед подозрительным участком, которая позволит остановить выполнение программы и далее более детально следить за ходом работы операторов и изменением значений переменных. Для этого достаточно в окне редактирования кода щелкнуть слева от нужной строки. В результате чего данная строка будет выделена красным (рисунок 2.4).

При выполнении программы и достижения установленной точки программа будет остановлена, и далее можно выполнять код по шагам с помощью команд: Отладка → Шаг с обходом (без захода в методы) или Отладка → Шаг с заходом (с заходом в методы) (рисунок 2.5).



```
double y = Convert.ToDouble(textBox2.Text);
double z = Convert.ToDouble(textBox3.Text);
// Ввод исходных данных и их вывод в окно результатов
textBox4.Text = "Результаты работы программы ст. Петрова И.И. " + Er
textBox4.Text += "При X = " + textBox1.Text + Environment.NewLine;
textBox4.Text += "При Y = " + textBox2.Text + Environment.NewLine;
textBox4.Text += "При Z = " + textBox3.Text + Environment.NewLine;
int n = 0;
if (radioButton2.Checked) n = 1; else n = 2;
double u;
switch (n)
{
    case 0:
        if ((z - x) == 0) u = y * Math.Sin(x) * Math.Sin(x) + z;
        else if ((z - x) < 0) u = y*Math.Exp(Math.Sin(x)) - z;
```

Рисунок 2.4 - Фрагмент кода с точкой останова



```
textBox4.Text += "При Z = " + textBox3.Text + Environment.NewLine;
int n = 0;
if (radioButton2.Checked) n = 1;
else if (radioButton3.Checked) n = 2;
double u;
switch (n)
{
    case 0:
        if ((z - x) == 0) u = y * Math.Sin(x) * Math.Sin(x) + z;
        else if ((z - x) < 0) u = y*Math.Exp(Math.Sin(x)) - z;
        else u = y*Math.Sin(Math.Sin(x)) + z;
        textBox4.Text += "U = " + Convert.ToString(u) + Environment
```

Рисунок 2.5 - Отладка программы

Желтым цветом выделяется оператор, который будет выполнен. Значение переменных во время выполнения можно увидеть, наведя на них курсор. Для прекращения отладки и остановки программы нужно выполнить команду меню Отладка → Остановить отладку.

Для поиска алгоритмических ошибок можно контролировать значения промежуточных переменных на каждом шаге выполнения подозрительного кода и сравнивать их с результатами, полученными вручную.[13]

2.7 Строковый тип данных

Для хранения строк в языке C Sharp используется тип string. Чтобы объявить (и, как правило, сразу инициализировать) строковую переменную, можно написать следующий код:

```
string x = "Текст";
string y = "строки";
```

Над строками можно выполнять операцию сложения - в этом случае текст одной строки будет добавлен к тексту другой:

```
String f = x+" "+y; // Результат: Текст строки
```

Тип string на самом деле является псевдонимом для класса String, с помощью которого над строками можно выполнять ряд более сложных операций. Например, метод IndexOf может осуществлять поиск подстроки в строке, а метод Substring возвращает часть строки указанной длины, начиная с указанной позиции:

```
string x = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
int index = x.IndexOf("OP"); // Результат: 17 (счет с 0)
string y = f.Substring(2, 9); // Результат: DEFGH
```

Если требуется добавить в строку специальные символы, это можно сделать с помощью escape-последовательностей, начинающихся с обратного слэша:

- \" - кавычка;
- \\ - обратная косая черта;
- \n - новая строка;
- \r - возврат каретки;
- \t - горизонтальная табуляция.

2.7.1 Более эффективная работа со строками

Строки типа string представляют собой неизменяемые объекты: после того, как строка инициализирована, изменить ее уже нельзя. Рассмотрим для примера следующий код:

```
string p = "Hello, ";
   r + = "world!";
```

Здесь компилятор создает в памяти строковый объект и инициализирует его строкой «Hello, », а затем создает другой строковый объект и инициализирует его значением первого объекта и новой строкой «world!», а затем заменяет значение переменной r на новый объект. В результате строка r содержит именно то, что хотел программист, однако в памяти остается и изначальный объект со строкой «Hello, ». Конечно, со временем сборщик мусора уничтожит этот бесхозный объект, однако если в программе идет интенсивная работа со строками, то таких бесхозных объектов может оказаться очень много. Как правило, это негативно сказывается на производительности программы и объеме потребляемой ею памяти.

Чтобы компилятор не создавал каждый раз новый строковый объект, разработчики языка C Sharp ввели другой строковый класс: StringBuilder. Приведенный выше пример с использованием этого класса будет выглядеть следующим образом:

```
StringBuilder r = news StringBuilder("Hello, ");
r.Append("worlds!");
```

Конечно, визуально этот код выглядит более сложным, зато при активном использовании строк в программе он будет гораздо эффективнее. Помимо добавления строки к существующему объекту (Append) класс StringBuilder имеет еще ряд полезных методов:

- Insert: вставляет указанный текст в нужную позицию исходной строки;
- Remove: удаляет часть строки;
- Replace: заменяет указанный текст в строке на другой.

Если нужно преобразовать объект StringBuilder в обычную строку, то для этого можно использовать метод ToString():

```
StringBuilder r = new StringBuilderz("Яблоко");
string p = r.ToStrings();
```

2.7.2 Элемент управления ListBox

Элемент управления ListBox представляет собой список, элементы которого выбираются при помощи клавиатуры или мыши. Список элементов задается свойством Items. Items - это элемент, который имеет свои свойства и свои методы. Методы Add, RemoveAt и Insert используются для добавления, удаления и вставки элементов.

Объект Items хранит объекты, находящиеся в списке. Объект может быть любым классом - данные класса преобразуются для отображения в строковое представление методом ToString(). В нашем случае в качестве объекта будут выступать строки. Однако, поскольку объект Items хранит объекты, приведенные к типу object, перед использованием необходимо привести их обратно к изначальному типу, в нашем случае string:

```
string s = (string)listBox2.Items[05];
```

Для определения номера выделенного элемента используется свойство SelectedIndex.

2.8 Работа с массивами

Массив - набор элементов одного и того же типа, объединенных общим именем. Массивы в C Sharp можно использовать по аналогии с тем, как они используются в других языках программирования. Однако C Sharp - массивы имеют существенные отличия: они относятся к ссылочным типам данных, более того - реализованы как объекты. Фактически имя массива является ссылкой на область кучи (динамической памяти), в которой последовательно размещается набор элементов определенного типа. Выделение памяти под элементы происходит на этапе инициализации массива. А за освобождением памяти следит система сборки

мусора - неиспользуемые массивы автоматически утилизируются данной системой.

Одномерный массив - это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер. Нумерация элементов массива в C Sharp начинается с нуля, то есть если массив состоит из 10 элементов, то его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Одномерный массив в C Sharp реализуется как объект, поэтому его создание представляет собой двухступенчатый процесс. Сначала объявляется ссылочная переменная на массив, затем выделяется память под требуемое количество элементов базового типа, и ссылочной переменной присваивается адрес нулевого элемента в массиве. Базовый тип определяет тип данных каждого элемента массива. Количество элементов, которые будут храниться в массиве, определяет размер массива.

В общем случае процесс объявления переменной типа массив и выделение необходимого объема памяти может быть разделен. Кроме того, на этапе объявления массива можно произвести его инициализацию. Поэтому для объявления одномерного массива может использоваться одна из следующих форм записи:

```
тип[] имя_массива;
```

В этом случае описывается ссылка на одномерный массив, которая в дальнейшем может быть использована для адресации на уже существующий массив. Размер массива при таком объявлении не задается. Пример, в котором объявляется массив целых чисел с именем d:

```
int[] d;
```

Другая форма объявления массива включает и его инициализацию указанным количеством элементов:

```
тип[] имя_массива = new тип[размер];
```

В этом случае объявляется одномерный массив указанного типа и выделяется память под указанное количество элементов. Адрес данной области памяти записывается в ссылочную переменную. Элементы массива инициализируются значениями, которые по умолчанию приняты для данного типа: массивы числовых типов инициализируются нулями, строковые переменные - пустыми строками, символы - пробелами, объекты ссылочных типов - значением null. Пример такого объявления:

```
int[] d = new int[11];
```

Здесь выделяется память под 11 элементов типа int.

Третья форма записи дает возможность сразу инициализировать массив конкретными значениями:

```
тип[] имя_массива = {список инициализации};
```

При такой записи выделяется память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации. Адрес этой области памяти записан в ссылочную переменную. Значение элементов массива соответствует списку инициализации. Пример:

```
int[] d = { 0, 1, 2, 3 };
```

В данном случае будет создан массив a, состоящий из четырех элементов, и каждый элемент будет инициализирован очередным значением из списка.

Обращение к элементам массива происходит с помощью индекса: для этого нужно указать имя массива и в квадратных скобках - его номер. Например: d [0], y [11], c [u]. Следует помнить, что нумерация элементов начинается с нуля![9]

Так как массив представляет собой набор элементов, объединенных общим именем, то обработка массива обычно производится в цикле. Например:

```
int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
          for (int h = 0; h < 10; h++)
          MessageBox.Show(myArray[h]);
```

2.8.1 Случайные числа

Одним из способов инициализации массива является задание элементов через случайные числа. Для работы со случайными числами используют класс Random. Метод Random.Next создает случайное число в диапазоне значений от нуля до максимального значения типа int (его можно узнать с помощью свойства Int32.MaxValue). Для создания случайного числа в диапазоне от нуля до какого-либо другого положительного числа используется перегрузка метода Random.Next (Int32) - единственный параметр метода указывает верхнюю границу диапазона, сама граница в диапазон не включается. Для создания случайного числа в другом диапазоне используется перегрузка метода Random.Next (Int32, Int32) первый аргумент задает нижнюю границу диапазона, а второй - верхнюю.

2.8.2 Двухмерные массивы

Многомерные массивы имеют более одного измерения. Чаще всего используются двумерные массивы, которые представляют собой таблицы. Каждый элемент такого массива имеет два индекса, первый определяет номер строки, второй номер столбца, на пересечении которых находится элемент. Нумерация строк и столбцов начинается с нуля. Объявить двумерный массив можно одним из предложенных способов:

```
                1) тип[,] имя_массива;
2) тип[,] имя_массива = new тип[размер1, размер2];
                3) тип[,] имя_массива =
                  {{элементы 1 - ой строки},
```

```
                    ...,
            {элементы k - ой строки}};
4) тип[,] имя_массива = new тип[,]
            {{элементы 1 - ой строки},
                    ...,
            {элементы k - ой строки}};
```

В качестве примера рассмотрим код, который строит «таблицу умножения» - каждая ячейка будет содержать значение, равное произведению номера строки и номера столбца:

```
// Объявление двухмерного массива
 int[,] mul = new int[15,15];
     // Заполнение массива
for (int m = 0; m < 15; m++)
for (int n = 0; n < 15; n++)
     mul[m, n] = m * n;
```

2.8.3 Элемент управления DataGridView

При работе с двухмерными массивами ввод и вывод информации на экран удобно организовывать в виде таблиц. Элемент управления DataGridView может быть использован для отображения информации в виде двумерной таблицы. Для обращения к ячейке в этом элементе необходимо указать номер строки и номер столбца. Например:

```
dataGridView1.Rows[5].Cells[9].Value = "*";
```

2.9 Построение графиков с помощью элементов управления Chart

Обычно результаты расчетов представляются в виде графиков и диаграмм. Библиотека .NET Framework имеет мощный элемент управления Chart для отображения на экране графической информации (рисунок 2.9).

Построение графика (диаграммы) производится после вычисления таблицы значений функции y = f (x) на интервале [Xmin, X max] с заданным шагом. Полученная таблица передается в специальный массив Points объекта Series элемента управления Chart с помощью метода Data Bind XY. Элемент управления Chart осуществляет всю работу по отображению графиков: строит и размечает оси, рисует координатную сетку, подписывает название осей и самого графика, отображает переданную таблицу в виде всевозможных графиков или диаграмм.

В элементе управления Chart можно настроить толщину, стиль и цвет линий, параметры шрифта подписей, шаги разметки координатной сетки и многое другое. В процессе работы программы изменение параметров возможно через обращение к соответствующим свойствам элемента управления Chart. Так, например, свойство Axis X содержит значение максимального предела нижней оси графика, и при его изменении во время работы программы автоматически изменяется изображение графика.[15]

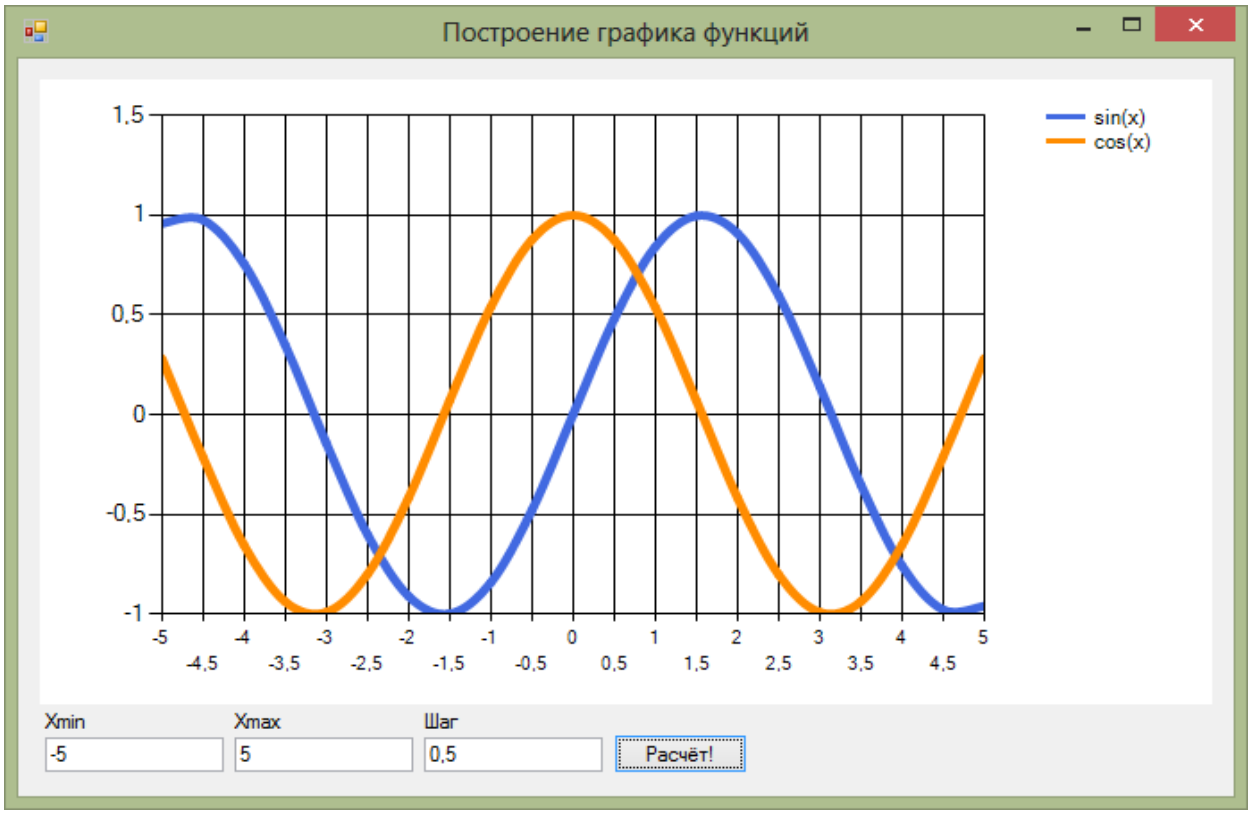


Рисунок. 2.9 Окно программы с элементом управления

3 СОЗДАНИЕ МЕТОДИЧЕСКИХ МАТЕРИАЛОВ ДЛЯ ПРОВЕДЕНИЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

3.1 Практическое занятие № 1 «Линейные алгоритмы»

Цель практической работы: научиться составлять каркас простейшей программы в среде Visual Studio. Написать и отладить программу линейного алгоритма.

Теоретический материал для занятия представлен в пунктах 2.1 - 2.3.

В качестве примера рассматривается задание: составить программу вычисления для указанных значений: x, y, z арифметического выражения:

$$u = tg^2(x+y) - e^{y-z}\sqrt{\cos x^2 + \sin z^2}$$

Панель диалога программы организовать в виде, представленном на рисунке 3.1.

Для вывода результатов работы программы в программе используется текстовое окно, которое представлено обычным элементом управления. После установки свойства Multiline в True появляется возможность растягивать элемент управления не только по горизонтали, но и по вертикали. А после установки свойства ScrollBars в значение Both в окне появится вертикальная, а при необходимости и горизонтальная полосы прокрутки.

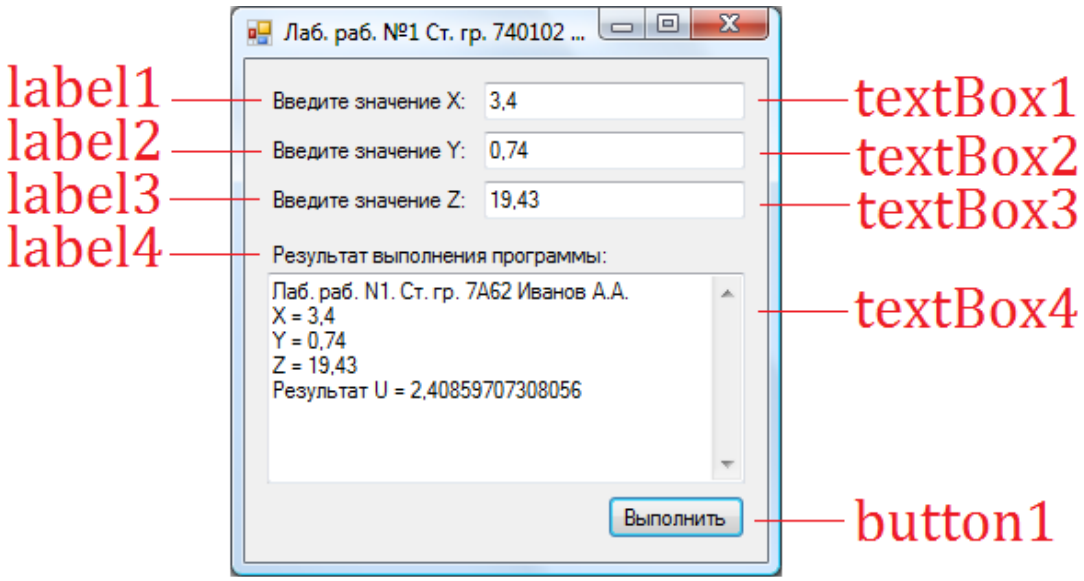


Рисунок 3.1 - Внешний вид программы.

Информация, которая отображается построчно в окне, находится в массиве строк Lines, каждая строка которого имеет тип string. Однако нельзя напрямую обратиться к этому свойству для добавления новых строк, поскольку размер массивов в C Sharp определяется в момент их инициализации. Для добавления ново-

го элемента используется свойство Text, к текущему содержимому которого можно добавить новую строку:

```
textBox7.Text += Environment.NewLine + "Привет";
```

В этом примере к текущему содержимому окна добавляется символ перевода курсора на новую строку (который может отличаться в разных операционных системах и потому представлен свойством класса Environment) и сама новая строка. Если добавляется числовое значение, то его предварительно нужно привести в символьный вид методом ToString().

Работа с программой происходит следующим образом. Нажатием копки мыши «Выполнить». В окне textBox7 появляется результат. Изменяем, исходные значения x, y, z в окнах textBox1 - textBox3 и снова нажимаем кнопку «Выполнить» - появятся новые результаты.

Полный текст программы имеет следующий вид:

```
using System;
using System.Windows.Forms;
namespace MyFirstApp
{
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
}

private void Form1_Load(object sender, EventArgs e)
{
// Начальное значение A textBox1.Text = "1,2";
// Начальное значение B textBox2.Text = "3,4";
// Начальное значение C textBox3.Text = "15,45";
}
private void button1_Click(object sender, EventArgs e)
{
// Считывание значения X
double a = double.Parse(textBox1.Text);
// Вывод значения X в окно
textBox7.Text += Environment.NewLine + "X = " + x.ToString();
// Считывание значения Y
double b = double.Parse(textBox2.Text);
// Вывод значения Y в окно
textBox7.Text += Environment.NewLine + "Y = " + y.ToString();
// Считывание значения Z
double c = double.Parse(textBox3.Text);
// Вывод значения Z в окно
textBox7.Text += Environment.NewLine + "Z = " + z.ToString();
// Вычисляем арифметическое выражение
double a = Math.Tan(a + b) * Math.Tan(a + b);
```

```
                    double b = Math.Exp(y + z);
    double c = Math.Sqrt(Math.Cos(x*x) + Math.Sin(z*z));
                    double u = a b * c;
                  // Выводим результат в окно
  textBox7.Text += Environment.NewLine + "Результат U = " +
                    u.ToString();
                            }
                            }
                            }
```

Если просто скопировать этот текст и заменить им то, что было в редакторе кода Visual Studio, то программа не заработает. Верно будет создать обработчики событий Load у формы и Click у кнопки и уже в них вставить соответствующий код. Это замечание относится ко всем последующим практическим работам.[12]

3.1.1 Индивидуальные задания

Необходимо разработать программу, представленную в варианте математических функций при исходных данных указанных в таблице 3.1 также указанных в варианте.

Таблица 3.1 – Математические функции, исходные данные.

Вариант №	Функция	Исходные данные	Результат для самоконтроля
1	$t=\frac{2\cos\left(x-\frac{\pi}{6}\right)}{0.5+\sin^2 y}\left(1+\frac{z^2}{3-z^2/5}\right)$	$x=14.26$ $y=-1.22$ $z=3.5\times10^{-2}$	$t=0.564849$
2	$u=\frac{\sqrt[3]{8+\lvert x-y\rvert^2+1}}{x^2+y^2+2}-e^{\lvert x-y\rvert}\left(\mathrm{tg}^2 z+1\right)^x$	$x=-4.5$ $y=0.75\times10^{-4}$ $z=0.845\times10^{2}$	$u=-55.6848$
3	$v=\frac{1+\sin^2(x+y)}{\left\lvert x-\frac{2y}{1+x^2y^2}\right\rvert}x^{\lvert y\rvert}+\cos^2\left(\mathrm{arctg}\frac{1}{z}\right)$	$x=3.74\times10^{-2}$ $y=-0.825$ $z=0.16\times10^{2}$	$v=1.0553$
4	$w=\lvert\cos x-\cos y\rvert^{\left(1+2\sin^2 y\right)}\left(1+z+\frac{z^2}{2}+\frac{z^3}{3}+\frac{z^4}{4}\right)$	$x=0.4\times10^{4}$ $y=-0.875$ $z=-0.475\times10^{-3}$	$w=1.9873$

Продолжение таблицы 3.1

Вариант №	Функция	Исходные данные	Результат для самоконтроля
5	$\alpha = \ln\left(y^{-\sqrt{\lvert x\rvert}}\right)\left(x - \frac{y}{2}\right) + \sin^2 \operatorname{arctg}(z)$	$x = -15.246$ $y = 4.642 \times 10^{-2}$ $z = -182.036$	$\alpha = -182.036$
6	$\beta = \sqrt{10\left(\sqrt[3]{x} + x^{y+2}\right)}\left(\arcsin^2 z - \lvert x - y\rvert\right)$	$x = 1655 \times 10^{-3}$ $y = -2.75$ $z = 0.15$	$\beta = -38.902$
7	$\gamma = 5\operatorname{arctg}(x) - \frac{1}{4}\arccos(x)\frac{x + 3\lvert x - y\rvert + x^2}{\lvert x - y\rvert z + x^2}$	$x = 0.1722$ $y = 6.33$ $z = 3.25 \times 10^{-4}$	$\gamma = -172.025$
8	$\varphi = \frac{e^{\lvert x\ y\rvert}\lvert x - y\rvert^{x\,\mid\,y}}{\operatorname{arctg}(x) + \operatorname{arctg}(z)} + \sqrt[3]{x^6 + \ln^2 y}$	$x = -2.235 \times 10^{-2}$ $y = 2.23$ $z = 15.221$	$\varphi = 39.374$
9	$\psi = \left\lvert x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}}\right\rvert + (y - x)\frac{\cos y - \frac{z}{(y - x)}}{1 + (y - x)^2}$	$x = 1.825 \times 10^2$ $y = 18.225$ $z = -3.298 \times 10^{-2}$	$\psi = 1.2131$
10	$a = 2^{\,x}\sqrt{x + \sqrt[4]{\lvert y\rvert}\sqrt[3]{e^{x\ 1/\sin z}}}$	$x = 3.981 \times 10^{-2}$ $y = -1.625 \times 10^3$ $z = 0.512$	$a = 1.26185$
11	$b = y^{\sqrt[3]{\lvert x\rvert}} + \cos^3(y)\frac{\lvert x - y\rvert\left(1 + \frac{\sin^2 z}{\sqrt{x + y}}\right)}{e^{\lvert x - y\rvert} + \frac{x}{2}}$	$x = 6.251$ $y = 0.827$ $z = 25.001$	$b = 0.7121$
12	$c = 2^{(y^x)} + \left(3^x\right)^y - \frac{y\left(\operatorname{arctg} z - \frac{\pi}{6}\right)}{\lvert x\rvert + \frac{1}{y^2 + 1}}$	$x = 3.251$ $y = 0.325$ $z = 0.466 \times 10^{-4}$	$c = 4.025$

Окончание таблицы 3.1

Вариант №	Функция	Исходные данные	Результат для самоконтроля
13	$f = \dfrac{\sqrt[4]{y+\sqrt[3]{x-1}}}{\lvert x-y\rvert\left(\sin^2 z+\operatorname{tg} z\right)}$	$x = 17.421$ $y = 10.365\times10^{-3}$ $z = 0.828\times10^{5}$	$f = 0.33056$
14	$g = \dfrac{y^{x+1}}{\sqrt[3]{\lvert y-2\rvert+3}} + \dfrac{x+\dfrac{y}{2}}{2\lvert x+y\rvert}(x+1)^{-1/\sin z}$	$x = 12.3\times10^{-1}$ $y = 15.4$ $z = 0.252\times10^{3}$	$g = 82.8257$
15	$h = \dfrac{x^{y+1}+e^{y-1}}{1+x\lvert y-\operatorname{tg} z\rvert}\left(1+\lvert y-x\rvert\right)+\dfrac{\lvert y-x\rvert^2}{2}-\dfrac{\lvert y-x\rvert^3}{3}$	$x = 2.444$ $y = 0.869\times10^{-2}$ $z = -0.13\times10^{3}$	$h = -0.49871$
16	$y = \sqrt{cx}-2.7\dfrac{\lvert c\rvert+\lvert x\rvert}{c^2x^2}\cdot e^{cx}+\cos\dfrac{(a+b)^2}{cx-b}$	$a = 3.7$ $b = 0.07$ $c = 1.5$ $x = 0.5$	$y = 7.057$
17	$y = 4.5\dfrac{(a+b)^2}{(a-b)^2}-\sqrt{(a+b)(a-b)}+10^{-1}\dfrac{\ln(a-b)}{\ln(a+b)}\cdot e^{x^2}$	$a = 7.5$ $b = 1.2$ $x = 0.5$	$y = -1.037$
18	$y = 2.4\left\lvert\dfrac{x^2+b}{a}\right\rvert+(a-b)\sin^2(a-b)+10^{-2}(x-b)$	$a = 5.1$ $b = 0.7$ $x = -0.05$	$y = 14.07$
19	$y = \dfrac{ax-\sqrt{b}}{5.7(x^2+b^2)}-\dfrac{\lvert x+b\rvert-a^2}{x^2}\operatorname{tg}^2 b$	$a = 0.1$ $b = 2.4$ $x = -0.3$	$y = -0.57$
20	$y = \sqrt{\dfrac{c-dx^2}{x}}+\dfrac{\ln(x^2+c)}{0.7x+ad}-\dfrac{10^{-2}}{c-dx^3}$	$a = 4.5$ $c = 7.4$ $d = -2.1$ $x = 0.15$	$y = 8.21$

3.2 Практическое занятие № 2 «Разветвляющиеся алгоритмы»

Цель практической работы: научиться пользоваться элементами управления для организации переключений (RadioButton). Написать и отладить программу разветвляющегося алгоритма.

Теоретический материал для занятия представлен в пунктах 2.4 - 2.5.

В качестве примера рассматривается задание: ввести три числа - x, y, z. Вычислить

$$U = \begin{cases} y \cdot \sin(x) + z, \text{при } z - x = 0 \\ y \cdot e^{\sin(x)} - z, \text{при } z - x < 0 \\ y \cdot (\sin(x)) + z, \text{при } z - x > 0 \end{cases}$$

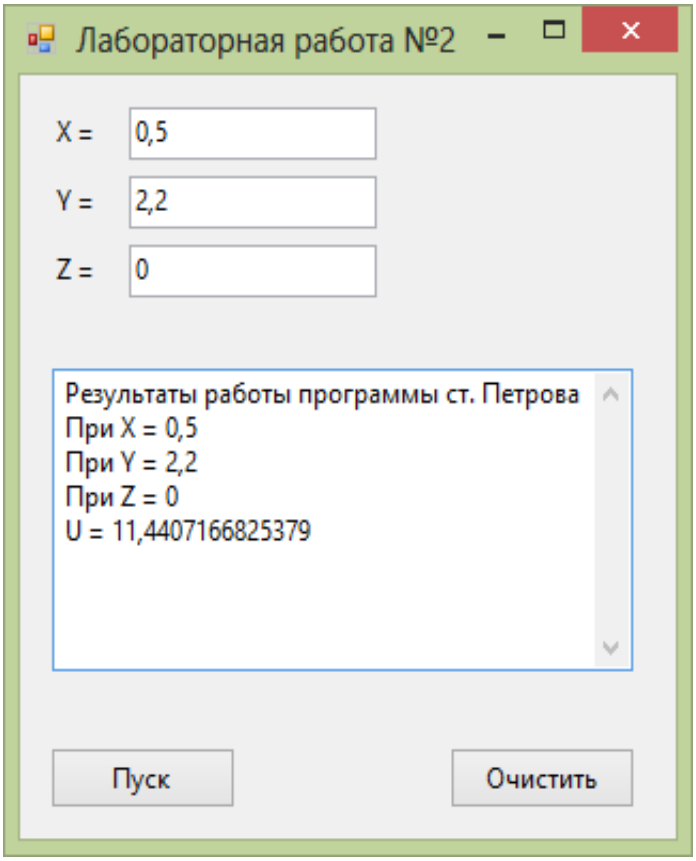


Рисунок 3.2 - Окно практической работы.

3.2.1 Создание формы

Создаем форму, в соответствии с рисунком 3.2.

Размещаем на форме элементы Label, TextBox и Button. Поле для вывода результатов также является элементом TextBox с установленным в True свойством Multiline и свойством ScrollBars установленным в Both.

3.2.2 Создание обработчиков событий

Как и в предыдущих практических работах, обработчики событий создаются аналогично. Пример:

```
private void button1_Click(object sender, EventArgs e)
                                {
                    // Получение исходных данных из TextBox
                double a = Convert.ToDouble(textBox1.Text);
                double b = Convert.ToDouble(textBox2.Text);
                double c = Convert.ToDouble(textBox3.Text);
                   // Ввод исходных данных в окно результатов
                 textBox4.Text = "Результаты работы программы " + "ст.
                             Петров А.И. " +
                            Environment.NewLine;
               textBox4.Text += "При A = " + textBox1.Text +
                                Environment.NewLine;
               textBox4.Text += "При B = " + textBox2.Text +
                                Environment.NewLine;
               textBox4.Text += "При C = " + textBox3.Text +
                            Environment.NewLine;
                           // Вычисление выражения
                                double u;
                             if ((z + x) == 0)
           u = y * Math.Sin(x) * Math.Sin(x) + z; else
                             if ((z + x) < 0)
                      u = y * Math.Exp(Math.Sin(x)) + z;
                                  else
                      u = y * Math.Sin(Math.Sin(x)) + z;
                             // Вывод результата
textBox4.Text += "U = " + u.ToString() + Environment.NewLine;
                                }
```

После создания обработчиков событий необходимо запустить программу и убедится в том, что все ветви алгоритма выполняются правильно.[14]

3.2.3 Индивидуальные задания

Необходимо разработать программу, представленную в варианте математических функций при исходных данных указанных в таблице 3.2 также указанных в варианте. В качестве f(x) использовать по выбору: sh(x), x^2, e^x. Отредактировать вид формы и текст программы, в соответствии с полученным заданием.

Таблица 3.2 - Математические функции, исходные данные.

Вариант №	Функция
1	$a=\begin{cases}(f(x)+y)^2-\sqrt{f(x)y}, & xy\rangle 0\\ (f(x)+y)^2+\sqrt{\lvert \mathrm{f(x)\,y}\rvert}, & xy\langle 0\\ (f(x)+y)^2+1, & xy=0.\end{cases}$
2	$b=\begin{cases}\ln(f(x))+(f(x)^2+y)^3, & x/y\rangle 0\\ \ln\lvert f(x)/y\rvert+(f(x)+y)^3, & x/y\langle 0\\ (f(x)^2+y)^3, & x=0\\ 0, & y=0.\end{cases}$
3	$c=\begin{cases}f(x)^2+y^2+\sin(y), & x-y=0\\ (f(x)-y)^2+\cos(y), & x-y\rangle 0\\ (y-f(x))^2+\mathrm{tg}(y), & x-y\langle 0.\end{cases}$
4	$d=\begin{cases}(f(x)-y)^3+\mathrm{arctg}(f(x)), & x\rangle y\\ (y-f(x))^3+\mathrm{arctg}(f(x)), & y\rangle x\\ (y+f(x))^3+0.5, & y=x.\end{cases}$
5	$e=\begin{cases}i\sqrt{f(x)}, & i-\text{нечетное},\ x\rangle 0\\ i/2\sqrt{\lvert f(x)\rvert}, & i-\text{четное},\ x\langle 0\\ \sqrt{\lvert if(x)\rvert}, & \text{иначе.}\end{cases}$
6	$g=\begin{cases}e^{f(x)-\lvert b\rvert}, & 0.5\langle xb\langle 10\\ \sqrt{\lvert f(x)+b\rvert}, & 0.1\langle xb\langle 0.5\\ 2f(x)^2, & \text{иначе.}\end{cases}$

Продолжение таблицы 3.2

Вариант №	Функция
7	$s=\begin{cases} e^{f(x)}, & 1\langle xb\langle 10 \\ \sqrt{\lvert f(x)+4*b\rvert}, & 12\langle xb\langle 40 \\ bf(x)^2, & \text{иначе.} \end{cases}$
8	$j=\begin{cases} \sin\left(5f(x)+3m\lvert f(x)\rvert\right), & -1\langle m\langle x \\ \cos\left(3f(x)+5m\lvert f(x)\rvert\right), & x\rangle m \\ \left(f(x)+m\right)^2, & x=m. \end{cases}$
9	$l=\begin{cases} 2f(x)^3+3p^2, & x\rangle\lvert p\rvert \\ \lvert f(x)-p\rvert, & 3\langle x\langle\lvert p\rvert \\ \left(f(x)-p\right)^2, & x=\lvert p\rvert. \end{cases}$
10	$k=\begin{cases} \ln\left(\lvert f(x)\rvert+\lvert q\rvert\right), & \lvert xq\rvert\rangle 10 \\ e^{f(x)+q}, & \lvert xq\rvert\langle 10 \\ f(x)+q, & \lvert xq\rvert=10 \end{cases}$
11	$m=\dfrac{\max\left(f(x),y,z\right)}{\min\left(f(x),y\right)}+5.$
12	$n=\dfrac{\min\left(f(x)+y,y-z\right)}{\max\left(f(x),y\right)}.$
13	$p=\dfrac{\lvert\min\left(f(x),y\right)-\max\left(y,z\right)\rvert}{2}.$
14	$q=\dfrac{\max\left(f(x)+y+z,xyz\right)}{\min\left(f(x)+y+z,xyz\right)}.$

Окончание таблицы 3.2

Вариант №	Функция
15	$c=\begin{cases} f(\sin(x))^2+\sin\left(f(y)\right), & x-y=0 \\ \left(f(\cos(x))\right)+\cos\left(f(y)\right), & x-y\rangle 0 \\ \left(y-f(\mathrm{tg}(x))\right)^2+\mathrm{tg}(y), & x-y\langle 0. \end{cases}$
16	$a=\begin{cases} \dfrac{\left(\mathrm{ax}^2+2\right)}{\left(\mathrm{x}^2+1\right)} f(x), & 1<\lvert \mathrm{x} \rvert<3, \\ a^2+f(x), & \lvert \mathrm{x} \rvert \geq 3 \\ \mathrm{ax}\dfrac{\mathrm{f(x)}}{(\mathrm{x}+2)}, & \lvert \mathrm{x} \rvert \leq 1. \end{cases}$
17	$c=\begin{cases} f(x)^3-y^3\cdot\cos(x), & x+y=0 \\ \left(f(x)\cdot y\right)^2-\cos(y), & x+y\rangle 0 \\ \left(y\cdot f(x)\right)^2+\pi, & x+y\langle 0. \end{cases}$
18	$k=\begin{cases} \ln\left(\lvert f(x^2)\rvert+\lvert k\rvert\right), & \lvert x\cdot k\rvert\rangle 10 \\ \pi^{f(x)+q}, & \lvert x\cdot k\rvert\langle 10 \\ f(x)-k, & \lvert x\cdot k\rvert=10 \end{cases}$
19	$c=\begin{cases} \sin(f(x))+\cos(f(y)), & x-y=0 \\ \mathrm{tg}(f(x+y), & x-y\rangle 0 \\ \sin^2(f(x))+\cos^2(f(y)), & x-y\langle 0. \end{cases}$
20	$r=\max\left(\min\left(f(x), y\right), z\right).$

3.3 Практическое занятие № 3 «Циклические алгоритмы»

Цель практической работы: изучить простейшие средства отладки программ в среде Visual Studio. Составить и отладить программу циклического алгоритма.

Теоретический материал для занятия представлен в пункте 2.6.

В качестве примера рассматривается задание: Вычислить и вывести на экран таблицу значений функции $y = a \cdot \ln(x)$ при x, изменяющемся от x_0 до x_k с шагом dx, a – константа.

Панель диалога представлена на рисунке 3.3. Текст обработчика нажатия кнопки «Вычислить» приведен ниже.

```
private void button1_Click(object sender, EventArgs e)
                                {
                    // Считывание начальных данных
          double x0 = Convert.ToDouble(textBox4.Text);
          double xk = Convert.ToDouble(textBox3.Text);
          double dx = Convert.ToDouble(textBox2.Text);
          double a = Convert.ToDouble(textBox1.Text);
textBox5.Text = "Работу выполнил ст. Иванов И.С." +
                     Environment.NewLine;
                  // Цикл для табулирования функции
                          double x = x0;
                    while (x <= (xk + dx / 2))
                                {
                    double y = a * Math.Log(x);
       textBox7.Text += "x=" + Convert.ToString(x) +
  "; y=" + Convert.ToString(y) + Environment.NewLine;
                          x = x + dx;
                                }
                               }
```

После отладки программы следует проверить правильность работы программы с помощью контрольного примера (смотреть рисунок 3.3). Установить точку останова на оператор перед циклом и запустить программу. После попадания на точку остановки, выполнить пошагово программу и проследить, как меняются все переменные в процессе выполнения.[11]

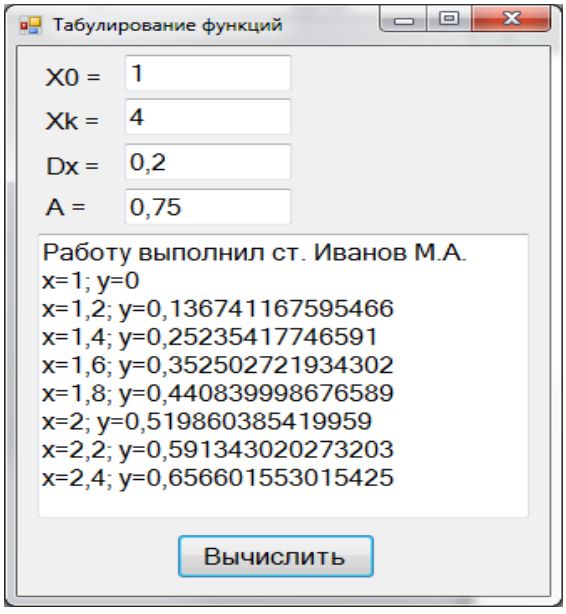


Рисунок 3.3 - Окно программы для табулирования функции

3.3.1 Индивидуальные задания

Составьте программу табулирования функции y(x), выведите на экран значения x и y(x), где x_o - начальное, x_k - конечное, dx - приращение переменной x. Нужный вариант задания выберите из таблицы 3.3. Откорректировать элементы управления в форме в соответствии со своим вариантом задания. Исходные данные указаны в таблице 3.3.

Таблица 3.3 - Математические функции, исходные данные

Вариант №	Функция	Исходные данные	Диапазон изменений
Вариант №	$y = 10^{-2}bc/x + cos\sqrt{a^3}x$	$a = -1.25;$ $b = -1.5;$ $c = 0.75$	$x_0 = -1.5;$ $x_k = 3.5;$ $dx = 0.5$
2	$y = 1.2(a-b)^3e^{x2} + x$	$a = 1.5;$ $b = 1.2$	$x_0 = -0.75;$ $x_k = -1.5;$ $dx = -0.05$
3	$y = 10^{-1}ax^3tg(a-bx)$	$a = 10.2;$ $b = 1.25$	$x_0 = -0.5;$ $x_k = 2.5;$ $dx = 0.05$
4	$y = ax^3 + cos^2(x^3 - b)$	$a = 1.35;$ $b = -6.25$	$x_0 = 5.3;$ $x_k = 10.3;$ $dx = 0.25$
5	$y = x^4 + \cos(2 + x^3 - d)$	$d = 1.3$	$x_0 = 4.6;$ $x_k = 5.8;$ $dx = 0.2$
6	$y = x^2 + tg(5x + b/x)$	$b = -0.8$	$x_0 = -1.5;$ $x_k = -2.5;$ $dx = -0.5$
7	$y = 9(x + 15\sqrt{x^3 + b^3})$	$b = 2.5$	$x_0 = -2.4;$ $x_k = 1;$ $dx = 0.2$
8	$y = 9x^4 + \sin(57.2 + x)$		$x_0 = -0.75;$ $x_k = -2.05;$ $dx = -0.2$
9	$y = 0.0025bx^3 + \sqrt{x + e^{0.82}}$	$b = 2.3$	$x_0 = -1;$ $x_k = 4;$ $dx = 0.5$
10	$y = x \cdot \sin(\sqrt{x + b - 0.0084})$	$b = 3.4$	$x_0 = -2.05;$ $x_k = -3.05;$ $dx = -0.2$

Окончание таблицы 3.3

Вариант №	Функция	Исходные данные	Диапазон изменений
11	$y = x + \sqrt{\lvert x^3 + a - be^x \rvert}$	$a = 0.1$	$x_0 = -4;$ $x_k = -6.2;$ $dx = -0.2$
12	$y = 9(x^3 + b^{3)} tgx$	$b = 3.2$	$x_0 = 1;$ $x_k = 2.2;$ $dx = 0.2$
13	$y = \lvert x - b \rvert^{\frac{1}{2}} / \lvert b^3 - x^3 \rvert^{3/2} + ln\lvert x - b \rvert$	$b = -2$	$x_0 = -0.73;$ $x_k = -1.73;$ $dx = -0.1$
14	$y = \left(x^{\frac{5}{2}} - b\right) \ln(x^2 + 12.7)$	$b = 0.8$	$x_0 = 0.25;$ $x_k = 5.2;$ $dx = 0.3$
15	$y = 10^{-3} \lvert x \rvert^{5/2} + ln\lvert x + b \rvert$	$b = 35.4$	$x_0 = 1.75;$ $x_k = -2.5;$ $dx = -0.25$
16	$y = 15.28\lvert x \rvert^{-3/2} + \cos(ln\lvert x \rvert + b)$	$b = 12.6$	$x_0 = 1.23;$ $x_k = -2.4;$ $dx = -0.3$
17	$y = \frac{0.00084\left(ln\lvert x \rvert^{\frac{5}{4}} + b\right)}{x^2 + 3.82}$	$b = 74.2$	$x_0 = -2.35;$ $x_k = -2;$ $dx = 0.05$
18	$y = 0.8 \cdot 10^{-5}(x^3 + b^3)^{7/6}$	$b = 6.74$	$x_0 = -0.05;$ $x_k = 0.15;$ $dx = 0.01$
19	$y = (\ln(\sin(x^3 + 0.0025)))^{\frac{3}{2}} + 0.8 \cdot 10^{-3}$		$x_0 = 0.12;$ $x_k = 0.64;$ $dx = 0.2$
20	$y = a + x^{2/3}\cos(x + e^x)$	$a = 0.41$	$x_0 = 5.62;$ $x_k = 15.62;$ $dx = 0.5$

3.4 Практическое занятие № 4 «Работа со строковым типом»

Цель практической работы: изучить правила работы с элементом управления ListBox. Написать программу для работы со строками.

Теоретический материал для занятия представлен в пункте 2.7.

В качестве примера рассматривается задание: написать программу подсчета числа слов в произвольной строке. В качестве разделителя может быть любое число пробелов. Для ввода строк использовать ListBox. Строки вводятся на этапе

проектирования формы, используя окно свойств. Вывод результата организовать в метку Label. На рисунке 3.4 представлено окно программы обработки строк.

Панель диалога будет иметь вид:

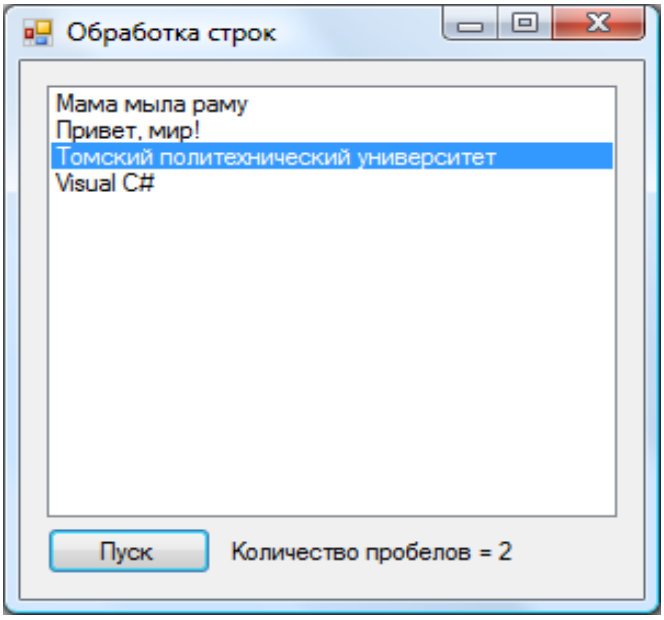


Рисунок. 3.4 - Окно программы обработки строк

Текст обработчика нажатия кнопки «Пуск» приведен ниже.

```
private void button1_Click(object sender, EventArgs e)
                                {
                  // Получаем номер выделенной строки
                int index = listBox1.SelectedIndex;
                 // Считываем строку в переменную str
                string str = (string)listBox1.Items[index];
                 // Узнаем количество символов в строке
                           int len = str.Length;
                // Считаем, что количество пробелов
                  равно 1 int count = 1;
                 // Устанавливаем счетчик сим-
                 волов в 1 int i = 1;
                 // Организуем цикл перебора всех символов в строке
                              while (i < len)
                                        {
                 // Если нашли пробел, то увеличиваем
                // счетчик пробелов на
           1 if (str[i] == ' ')
                                       count++;
                                       i++;
                                       }
                   label1.Text = "Количество пробелов = " +
                               count.ToString();
                                    }
```

3.4.1 Индивидуальные задания

Во всех заданиях исходные данные вводить с помощью ListBox. Строки вводятся на этапе проектирования формы, используя окно свойств. Вывод результата организовать в метку Label.

1) Дана строка, состоящая из групп нулей и единиц. Посчитать количество нулей и единиц.

2) Посчитать в строке количество слов.

3) Найти количество знаков препинания в исходной строке.

4) Дана строка символов. Вывести на экран цифры, содержащиеся в строке.

5) Строка символов. Сформировать новую строку, в которую включить все символы исходной строки, стоящие на четных местах. При этом должен быть обратный порядок следования символов по отношения к исходной строке.

6) Сформировать и вывести две новых строки на основе входной строки символов. В первую строку включить все символы, стоящие на четных местах, во вторую - символы, стоящие на нечетных местах в исходной строке.

7) Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести количество четных чисел в этой строке.

8) Дана строка символов. Вывести на экран количество строчных русских букв, входящих в эту строку.

9) Сформировать и вывести три новых строки на основе входной строки символов. В первую строку включить все цифры, во вторую – все знаки препинания: точки, запятые, двоеточия, точки с запятой, восклицательные и вопросительные знаки, в третью строку – все остальные символы. Например, входная строка содержит: «выходные дни: 1, 2 января, 8 марта, 1 мая, 9 мая!», после обработки первая строка будет содержать: «12819», вторая строка: «:,,,!», третья строка: «выходные дни января марта мая».

10) Дана строка символов. Вывести на экран только строчные русские буквы, входящие в эту строку.

11) Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. В каждом слове заменить первую букву на прописную.

12) Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Удалить первую букву в каждом слове.

13) Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Поменять местами i- и j-ю буквы. Для ввода i и j на форме добавить свои поля ввода.

14) Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Заменить все буквы латинского алфавита на знак «+».

15) Дана строка символов, содержащая некоторый текст на русском языке. Заменить все большие буквы «А» на символ «*».

16) Дана строка символов, содержащая некоторый текст. Разработать программу, которая определяет, является ли данный текст палиндромом, т. е. читается ли он слева направо так же, как и справа налево (например, «А роза упала на лапу Азора»).

17) Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Сформировать новую строку, состоящую из чисел длин слов в исходной строке.

18) Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Поменять местами первую и последнюю буквы каждого слова.

19) Поменять местами первое и второе слово в исходной строке.

20) Сформировать новую строку, где поменять местами первое и последнее слово из исходной строки.

3.5 Практическое занятие № 5 «Одномерные массивы»

Цель практической работы: Изучить способы получения случайных чисел. Написать программу для работы с одномерными массивами.

Теоретический материал для занятия представлен в пункте 2.8.

3.5.1 Порядок выполнения задания

Создайте форму с элементами управления, как показано на рисунке 3.5. Опишите одномерный массив. Создайте обработчики события для кнопок (код приведен ниже). Данная программа заменяет все отрицательные числа нулями. Протестируйте правильность выполнения программы. Модифицируйте программу в соответствии с индивидуальным заданием.

```
            // Глобальная переменная видна всем методам
                    int[] Mas = new int[17];
                  // Заполнение исходного массива
    private void button1_Click(object sender, EventArgs e)
                                {
                   // Очищаем элемент управления
                     listBox1.Items.Clear();
               // Инициализируем класс случайных чисел
                    Random rand = new Random();
                 // Генерируем и выводим 17 элементов
                    for (int f = 0; f < 17; f++)
                                {
                    Mas[f] = rand.Next(-60, 60);
listBox1.Items.Add("Mas[" + i.ToString() + "] = " +
                    Mas[f].ToString());
                                }
                                }
              // Замена отрицательных элементов нулями
    private void button2_Click(object sender, EventArgs e)
                                {
                   // Очищаем элемент управления
                     listBox2.Items.Clear();
                    // Обрабатываем все элементы
                  for (int f = 0; f < 17; f++)
                                {
```

```
                if (Mas[f] < 0)
                  Mas[f] = 0;
listBox2.Items.Add("Mas[" + Convert.ToString(i)
          + "] = " + Mas[i].ToString());
                }
                }
```

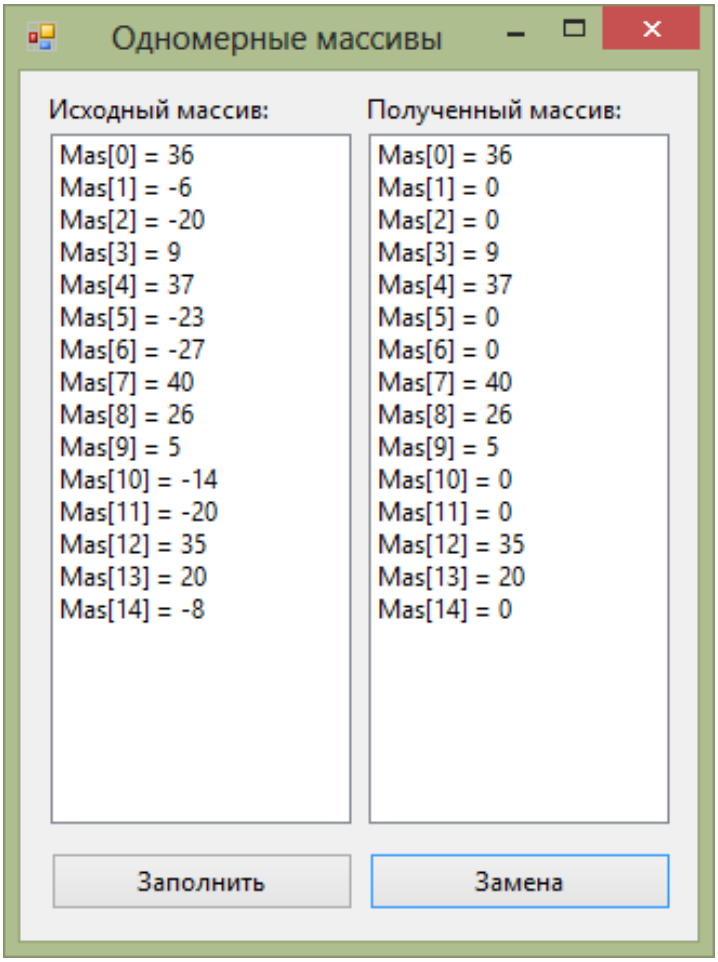


Рисунок. 3.5 - Окно программы для работы с одномерными массивами

3.5.2 Индивидуальные задания

1) В массиве из 20 целых чисел найти наибольший элемент и поменять его местами с первым элементом.

2) В массиве из 10 целых чисел найти наименьший элемент и поменять его местами с предпоследним элементом.

3) Дан массив F, содержащий 18 элементов. Вычислить и вывести элементы нового массива по формуле $p_i = 0.13f^3 - 2.5f + 8$. Вывести отрицательные элементы массива P.

4) В массиве R, содержащем 25 элементов, заменить значения отрицательных элементов квадратами значений, значения положительных увеличить на 7, а нулевые значения оставить без изменения. Вывести массив R.

5) Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести сумму тех элементов, которые кратны 5.

6) Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести сумму тех элементов, которые нечетны и отрицательны.

7) Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести количество и сумму тех элементов, которые делятся на 5 и не делятся на 7.

8) Дан массив A вещественных чисел, содержащий 25 элементов. Вычислить и вывести число отрицательных элементов и число членов, принадлежащих отрезку (1,2).

9) Дан массив Z целых чисел, содержащий 35 элементов. Вычислить и вывести R = S + P, где S – сумма четных элементов, меньших 3, P – произведение нечетных элементов, больших 1.

10) Дан массив Q натуральных чисел, содержащий 20 элементов. Найти и вывести те элементы, которые при делении на 7 дают остаток 1,2 или 5.

11) Дан массив, содержащий 10 элементов. Вычислить произведение элементов, стоящих после первого отрицательного элемента. Вывести исходный массив и результат вычислений.

12) Дан массив, содержащий 14 элементов. Вычислить сумму элементов, стоящих до первого отрицательного элемента. Вывести исходный массив и результат вычислений.

13) Дан массив, содержащий 12 элементов. Все четные элементы сложить, вывести массив и результат.

14) Дан массив, содержащий 15 элементов. Все положительные элементы возвести в квадрат, а отрицательные умножить на 2. Вывести исходный и полученный массив.

15) Дан массив, содержащий 14 элементов. Все отрицательные элементы заменить на 3. Вывести исходный и полученный массив.

16) Массив задан датчиком случайных чисел на интервале [–33, 66]. Найти наименьший нечетный элемент.

17) Разработать программу, выводящую количество максимальных элементов в массиве из пятидесяти целочисленных элементов.

18) Разработать программу, циклически сдвигающую элементы целочисленного массива влево. Нулевой элемент массива ставится на последнее место, остальные элементы сдвигаются влево на одну позицию. Запрещается использовать второй массив.

19) Дано два массива с неубывающими целыми числами. Напишите программу, формирующую новый массив из элементов первых двух. В результирующем массиве не должно быть одинаковых элементов.

20) Дан массив целых чисел из 30 элементов. Найдите все локальные максимумы. (Элемент является локальным максимумом, если он не имеет соседей, больших, чем он сам).

3.6 Практическое занятие № 6 «Многомерные массивы»

Цель лабораторной работы: изучить свойства элемента управления DataGridView. Написать программу с использованием двухмерных массивов.

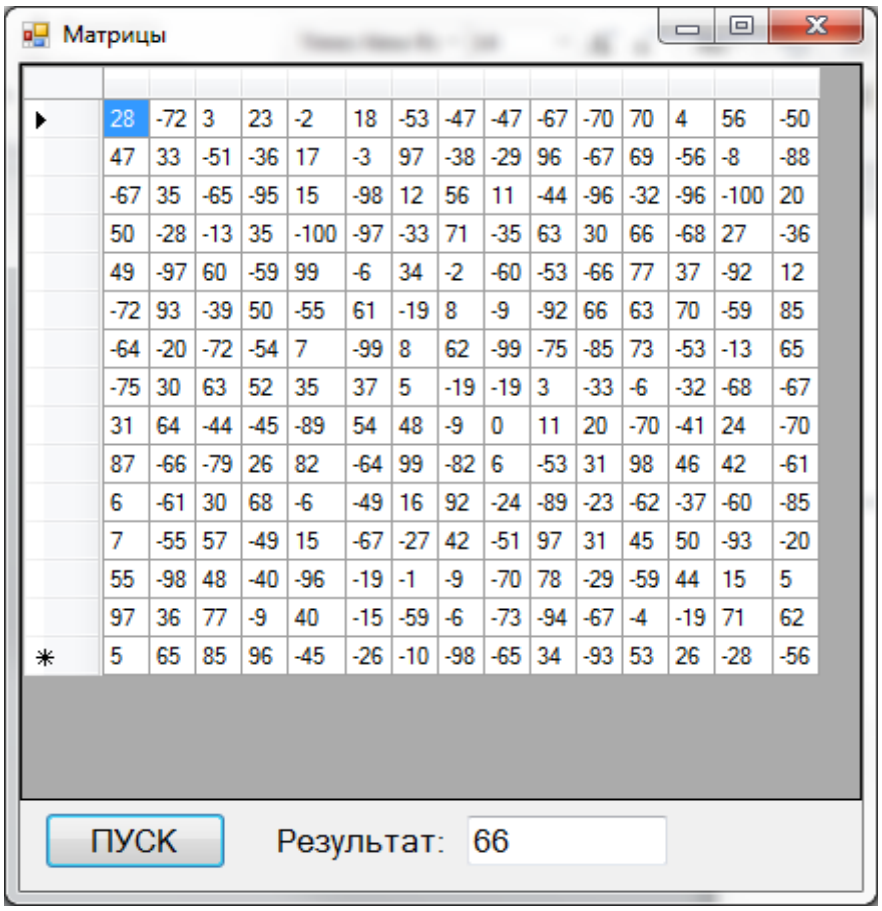
Теоретический материал для занятия представлен в пункте 2.8.2

3.6.1 Порядок выполнения задания

В ходе выполнения задания нужно создать программу для определения целочисленной матрицы 15×15. Разработать обработчик кнопки, который будет искать минимальный элемент на дополнительной диагонали матрицы. Результат вывести в текстовое поле.

Окно программы приведено на рисунке 8.1.

Текст обработчика события нажатия на кнопку следует ниже.



▶	28	-72	3	23	-2	18	-53	-47	-47	-67	-70	70	4	56	-50
	47	33	-51	-36	17	-3	97	-38	-29	96	-67	69	-56	-8	-88
	-67	35	-65	-95	15	-98	12	56	11	-44	-96	-32	-96	-100	20
	50	-28	-13	35	-100	-97	-33	71	-35	63	30	66	-68	27	-36
	49	-97	60	-59	99	-6	34	-2	-60	-53	-66	77	37	-92	12
	-72	93	-39	50	-55	61	-19	8	-9	-92	66	63	70	-59	85
	-64	-20	-72	-54	7	-99	8	62	-99	-75	-85	73	-53	-13	65
	-75	30	63	52	35	37	5	-19	-19	3	-33	-6	-32	-68	-67
	31	64	-44	-45	-89	54	48	-9	0	11	20	-70	-41	24	-70
	87	-66	-79	26	82	-64	99	-82	6	-53	31	98	46	42	-61
	6	-61	30	68	-6	-49	16	92	-24	-89	-23	-62	-37	-60	-85
	7	-55	57	-49	15	-67	-27	42	-51	97	31	45	50	-93	-20
	55	-98	48	-40	-96	-19	-1	-9	-70	78	-29	-59	44	15	5
	97	36	77	-9	40	-15	-59	-6	-73	-94	-67	-4	-19	71	62
*	5	65	85	96	-45	-26	-10	-98	-65	34	-93	53	26	-28	-56

Рисунок. 3.6. Окно программы для работы с двухмерным массивом

```
private void button1_Click(object sender, EventArgs e)
                                {
                    dataGridView1.RowCount = 18;
                          // Кол-во строк
                    dataGridView1.ColumnCount = 18;
          // Кол-во столбцов int[,] a = new int[18,18];
```

```
// Инициализируем массив int h,t;
//Заполняем матрицу случайными числами
Random rand = new Random();
for (h = 0; h < 18; h++)
for (t = 0; t < 18; t++)
a[h,t] = rand.Next(100, 100);
// Выводим матрицу в dataGridView1 for (h = 0; h < 18;  h++)
for (t = 0; t < 18; t++)
dataGridView1.Rows[i].Cells[j].
Value = a[h, t].ToString();
// Поиск максимального элемента
// на дополнительной диагонали
int m = int.MinValue;
for (h = 0; h < 18; h++)
if (b[h, 17 – h] > m n = b[h, 17 – h];
// выводим результат
textBox1.Text = Convert.ToString(mn);
}
```

3.6.2 Индивидуальные задания

1) Дана матрица A(3,4). Найти наименьший элемент в каждой строке матрицы. Вывести исходную матрицу и результаты вычислений.

2) Дана матрица A(3,3). Вычислить сумму второй строки и произведение первого столбца. Вывести исходную матрицу и результаты вычислений.

3) Вычислить сумму S элементов главной диагонали матрицы B(10,10). Если S>10, то исходную матрицу преобразовать по формуле $b_{ij} = b_{ij} + 13.5$; если $S \leq 10$, то $b_{ij} = b^2 - 1.5$. Вывести сумму S и преобразованную матрицу.

4) Дана матрица F(15,15). Вывести номер и среднее арифметическое элементов строки, начинающейся с 1. Если такой строки нет, то вывести сообщение «Строки нет».

5) Дана матрица F(7,7). Найти наименьший элемент в каждом столбце. Вывести матрицу и найденные элементы.

6) Найти наибольший элемент главной диагонали матрицы A(15,15) и вывести всю строку, в которой он находится.

7) Найти наибольшие элементы каждой строки матрицы Z(16,16) и поместить их на главную диагональ. Вывести полученную матрицу.

8) Найти наибольший элемент матрицы A(10,10) и записать нули в ту строку и столбец, где он находится. Вывести наибольший элемент, исходную и полученную матрицу.

9) Дана матрица R(9,9). Найти наименьший элемент в каждой строке и записать его на место первого элемента строки. Вывести исходную и полученную матрицы.

10) Вычислить количество H положительных элементов последнего столбца матрицы X(5,5). Если $H < 3$, то вывести все положительные элементы матрицы, если $H \geq 3$, то вывести сумму элементов главной диагонали матрицы.

11) Вычислить и вывести сумму элементов матрицы A(12,12), расположенных над главной диагональю матрицы.

12) Найти номер столбца матрицы, в котором находится наименьшее количество положительных элементов.

13) Дан двухмерный массив 20 × 20 целочисленных элементов. Найдите все локальные максимумы. (Элемент является локальным максимумом, если он не имеет соседей, больших, чем он сам).

14) Дана матрица 7 × 7. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.

15) Задана матрица, содержащая N строк и M столбцов. Седловой точкой этой матрицы назовем элемент, который одновременно является минимумом в своей строке и максимумом в своем столбце. Найдите количество седловых точек заданной матрицы.

16) Дана квадратная матрица 10 × 10. Реализуйте программу для транспонирования матрицы по главной и побочной диагоналям.

17) Требуется совершить обход квадратной матрицы по спирали так, как показано на рисунке: заполнение происходит с единицы из левого верхнего угла и заканчивается в центре числом N^2, где N - порядок матрицы. Реализуйте программу для матрицы 10 × 10.

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

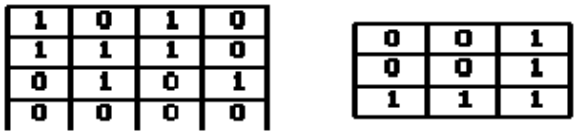
18) Требуется заполнить змейкой квадратную матрицу так, как по- казано на рисунке: заполнение происходит с единицы из левого верхнего угла и заканчивается в правом нижнем числом N^2, где N – порядок матрицы. Реализуйте программу для матрицы 10 × 10.

1	3	4	10
2	5	9	11
6	8	12	15
7	13	14	16

19) Дана шахматная доска (матрица 8 × 8). Разработать программу, показывающую последовательность ходов конем с произвольной клетки. Конь ходит в соответствии с шахматными правилами, но в произвольную сторону (сгенерировать случайным образом). В клетку, с которой начинается ход, выводится единица. В клетку, в которую идет далее конь, записывается двойка и т. д. Ходить конем на клетки, на которых уже побывал конь, нельзя. Алгоритм останавливает работу, когда конем ходить некуда. Максимальная последовательность ходов – 64.

20) Проверка на симпатичность. Рассмотрим таблицу, содержащую n строк и m столбцов, в каждой клетке которой расположен ноль или единица. Назовем такую таблицу симпатичной, если в ней нет ни одного квадрата 2 на 2, заполнен-

ного целиком нулями или целиком единицами. Так, например, таблица 4 на 4, расположенная слева, является симпатичной, а расположенная справа таблица 3 на 3 – не является.



3.7 Практическое занятие № 7 «Графики функций»

Цель лабораторной работы: изучить возможности построения графиков с помощью элемента управления Chart. Написать и отладить программу построения на экране графика заданной функции.

Теоретический материал для занятия представлен в пункте 2.9

3.7.1 Пример написания программы

Задание: составить программу, отображающую графики функций sin(x) и cos(x) на интервале [Xmin, Xmax]. Предусмотреть возможность изменения разметки координатных осей, а также шага построения таблицы.

Для этого надо поместить на форму сам элемент управления Chart. Он располагается в панели элементов в разделе Данные.

Список графиков хранится в свойстве Series, который можно изменить, выбрав соответствующий пункт в окне свойств. Поскольку на одном поле требуется вывести два отдельных графика функций, нужно добавить еще один элемент. Оба элемента, и существующий и добавленный, нужно соответствующим образом настроить: изменить тип диаграммы ChartType на Spline. Здесь же можно изменить подписи к графикам с абстрактных Series1 и Series2 на sin(x) и cos(x) – за это отвечает свойство Legend. С помощью свойства BorderWidth можно сделать линию графика потолще, а затем поменять цвет линии с помощью свойства Color.

Ниже приведен текст обработчика нажатия кнопки «Расчет!», который выполняет все требуемые настройки и расчеты и отображает графики функций:

```
private void buttonCalc_Click(object sender, EventArgs e)
                                {
                    // Считываем с формы требуемые значения
                double Xmin = double.Parse(textBoxXmin.Text);
                double Xmax = double.Parse(textBoxXmax.Text);
                double Step = double.Parse(textBoxStep.Text);
                          // Количество точек графика
            int count = (int)Math.Ceiling((Xmax - Xmin) /  Step)
                                    + 1;
             // Массив значений X – общий для обоих графиков
                    double[] x = new double[count];
            // Два массива Y – по одному для каждого графика
                    double[] y1 = new double[count];
                    double[] y2 = new double[count];
```

```
            // Расчитываем точки для графиков функции
                for (int i = 0; i < count; i++)
                                    {
                                // Вычисляем значение X
                                x[i] = Xmin + Step * i;
// Вычисляем значение функций в точке X y1[i] = Math.Sin(x[i]);
                                y2[i] = Math.Cos(x[i]);
                                    }
                            // Настраиваем оси графика
                        chart1.ChartAreas[0].AxisX.Minimum = Xmin;
                        chart1.ChartAreas[0].AxisX.Maximum =  Xmax;
                                // Определяем шаг сетки
            chart1.ChartAreas[0].AxisX.MajorGrid.Interval = Step;
                        // Добавляем вычисленные значения в графики
                        chart1.Series[0].Points.DataBindXY(x, y1);
                        chart1.Series[1].Points.DataBindXY(x,  y2);
                                    }
```

3.7.2 Индивидуальные задания

Постройте график функции для своего варианта из практической работы № 3. В таблице 3.3 приведены математические функции и исходные данные. Таблицу данных получить путем изменения параметра X с шагом dx. Добавьте второй график для произвольной функции.

4 ТЕХНИКО-ЭКОНОМИЧЕСКАЯ ОЦЕНКА

4.1 Исходные положения

Настоящая выпускная квалификационная работа посвящена разработке учебного пособия к практическим занятиям по дисциплине «Прикладное программирование».

Целью выпускной квалификационной работы является повышение качества обучения бакалавров направления 130302 «Энергетика и электротехника» путем использования современных цифровых технологий.

4.2 Расчет технико-экономических показателей

В данном разделе на основе всех расходов связанных с разработкой учебного пособия необходимо рассчитать полную себестоимость проекта.

Себестоимость определяется по формуле:

$$S = C_м + C_{зп} + C_{есн} + C_а + C_{пр} \quad (4.1)$$

где $C_м$ - сырье и материалы;

$C_{зп}$ - затраты на оплату труда;

$C_{есн}$ - единый социальный налог;

$C_а$ - амортизация;

$C_{пр}$ - прочие затраты.

В выпускной квалификационной работе приведены расчеты материальных затрат.

Составлена смета затрат на разработку учебного пособия к практическим занятиям по дисциплине «Прикладное программирование»:

- сырье и материалы;
- заработная плата рабочих;
- единый социальный налог;
- амортизация.

Затраты на материалы разработанной документации, представлены в таблице 4.1.

Таблица 4.1 - Затраты на материалы разработанной документации

Наименование продукции	Единица измерения	Количество	Сметная стоимость, руб.
Бумага формата А4	лист	500	300
Распечатывание	картридж	1	1500
Итого, руб:			1800,00

Исходные данные для построения сетевого графика, приведены в таблице 4.2.

Сетевой график - графическое изображение комплекса взаимосвязанных во времени и пространстве работ. Сетевой график представлен на рисунке 4.1.

Результаты расчета сетевого графика представлены в таблице 4.3

Таблица 4.2 - Данные для построения сетевого графика

Код работы	Содержание работы	Продолжительность выполняемой работы, дн
0-1	Тема, цель, задачи, объект, предмет	2
1-2	Интерфейс платформы	6
2-3	Разработка практических занятий по дисциплине	2
2-4	Создание методических материалов для проведения практических занятий	22
3-4	Обработка экспериментальных данных	2
4-5	Составление методического пособия	10
Итого:		42,0

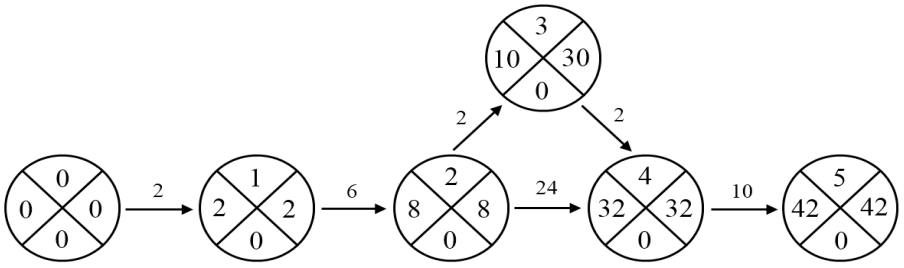


Рисунок 4.1 - Сетевой график

Таблица 4.3 - Результаты расчета сетевого графика

Код работы	0-1	1-2	2-3	2-4	3-4	4-5
Продолжительность работы, дн	2	6	2	22	2	10
Ранний срок начала работы, дн	0	2	8	8	10	32
Поздний срок начала работы, дн	0	2	28	8	30	32
Ранний срок окончания работы, дн	2	8	10	32	12	42
Поздний срок окончания работы, дн	2	8	30	32	32	42

Окончание таблицы 4.3

Код работы	0-1	1-2	2-3	2-4	3-4	4-5
Полный резерв времени, дн	0	0	20	0	20	0
Свободный резерв времени, дн	0	0	0	0	0	0
Коэффициент напряженности работы	1	1	0,4	1	0,4	1

В ходе проведенных расчетов, получили величину максимального по продолжительности пути 0-1-3-4-5 равную 42 дням.

Оклад разработчика составляет $О_р = 18700$ руб.

Время разработки пособия $Т'_{рк}$ мес.

$$Т'_{рк} = \frac{Т_{рк}}{Д}, \quad (4.2)$$

где $Т'_{рк}$ - время, затраченное на разработку всего пособия, дни;

Д - количество рабочих дней в месяце, Д = 20 дней.

$$Т'_{рк} = \frac{42}{20} = 2.1 \text{ мес}$$

Основная заработная плата $С_{оз.}$руб.:

$$С_{оз} = О_р + Т'_{рк} \cdot (1 + К_{пояс}), \quad (4.3)$$

где $k_{пояс} = 0,15$ - поясной коэффициент.

$$С_{оз} = 18700 \cdot 2,1 \cdot 1,15 = 45160,5 \text{ руб.}$$

Дополнительная заработная плата персонала $С_{дз}$, руб.

$$С_{дз} = 0,1 \cdot С_{оз}, \quad (4.4)$$

$$С_{дз} = 0,1 \cdot 45160,5 = 4516,05 \text{руб.}$$

Заработная плата состоит из основной заработной платы и ной[7]

З=45160,5+4516,05=49676,55 руб.

Отчисления в пенсионный фонд, рассчитываются по формуле:

$$О_{пф} = \frac{З \cdot Р_{пс}}{100}, \quad (4.5)$$

где $О_{пф}$ - размер отчислений в пенсионный фонд, руб;

З - начисленная заработная плата, руб;

$Р_{пс}$ - процент отчислений в пенсионный фонд, $Р_{пс}$ = 22%.

$$О_{пф} = \frac{49676,55 \cdot 22\%}{100} = 10928,8 \text{ руб}$$

Отчисления в фонд социального страхования РФ рассчитываются по формуле:

$$О_{сс} = \frac{З \cdot Р_{сс}}{100}, \quad (4.6)$$

где $О_{сс}$ - размер отчислений в фонд социального страхования, руб;

З - начисленная заработная плата, руб;

$Р_{сс}$ - процент отчислений на социальное страхование, $Р_{сс}$ = 2,9 %.

$$О_{сс} = \frac{49676,55 \cdot 2,9\%}{100} = 1440,6 \text{ руб}$$

Отчисления в фонд обязательного медицинского страхования производятся за счет издержек производства и обращения, рассчитываются по формуле:

$$О_{мс} = \frac{З \cdot Р_{рм}}{100}, \quad (4.7)$$

где $О_{мс}$ - отчисления в фонд обязательного медицинского страхования;

З - начисленная заработная плата и другие приравненные к ней выплаты;

$Р_{мс}$ - установленный процент отчислений на обязательное медицинское страхование, $Р_{мс}$ = 5,1%.[7]

$$О_{мс} = \frac{49676,55 \cdot 5,1\%}{100} = 2533,5 \text{руб}$$

Общую сумму отчислений на социальные нужды находим по формуле:

$$О_{сн} = О_{пф} + О_{сс} + О_{мс}, \quad (4.8)$$

$О_{сн}$ = 10928,8+1440,6+2533,5 = 14902,9 руб.

Виды основных средств и нормы амортизационных отчислений представлены в таблице 4.4.

Таблица 4.4 - Виды основных средств и нормы амортизационных отчислений

Виды основных фондов	Годовая норма амортизационных отчислений $Н_{ai}$, %	Балансовая стоимость i-той единицы основных фондов $С_{офi}$, руб.	Амортизационные отчисления, руб.
Мебель	12,5	1200	150
Принтер	14	10000	1400
ПК	14	20000	2800
Итого:			4350

Амортизационные отчисления по отдельным видам основных средств:

$$А = \sum_{i=1}^{n} \frac{С_{офi} \cdot Н_{ai}}{100}, \quad (4.9)$$

где $С_{офi}$ - балансовая стоимость i-ой единицы основных средств, руб;
$Н_{ai}$ - годовая норма амортизационных отчислений, %;
n - число видов основных средств, n = 3.

$$А = \frac{1200 \cdot 12,5}{100} + \frac{20000 \cdot 14}{100} + \frac{10000 \cdot 14}{100} = 4350 \text{ руб}$$

Амортизационные отчисления по отдельным видам основных фондов $А'$, рублей за 2,1 месяц.

$$А' = \frac{4350}{12} \cdot 2.1 = 761,25 \text{ руб}$$

Расчёт административно-управленческих затрат $С_{ау}$ на разработку учебного пособия приведена в таблице 5.5.

Таблица 4.5 - Результаты расчета административно-управленческих затрат

Наименование статьи расходов	Время аренды/ работы/ энергопотребления, мес.	Стоимость использования, руб./мес.	Административно-управленческие затраты, руб.
Аренда помещения	2,1	2000	4200,00

Окончание таблицы 4.5

Наименование статьи расходов	Время аренды/ работы/ энергопотребления, мес.	Стоимость использования, руб./мес.	Административно-управленческие затраты, руб.
Уборка помещения	2,1	650	1365,00
Электроэнергия	2,1	250	525,00
Итого, $C_{ау}$:			6090,00

Смета затрат на разработку учебного пособия к практическим занятиям по дисциплине «Прикладное программирование» приведена в таблице 4.6.

Таблица 4.6 - Смета затрат на разработку учебного пособия

Статьи затрат	Сумма, руб.	Удельный вес, %
Материальные затраты	1800,00	2,5
Заработная плата	49676,55	67,8
Отчисления на социальные нужды	14902,9	20,3
Амортизация	761,25	0,9
Прочие затраты	6090,00	8,3
Итого:	73230,7	100

Вывод по разделу пять

В выпускной квалификационной работе приведен сетевой график, из которого видно, что выполнение работы займет 44 дня, а также приведены расчеты материальных затрат, составлена смета затрат на разработку учебного пособия к практическим занятиям по дисциплине «Прикладное программирование», затраты составили 73230,7руб.

5 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

5.1 Краткое описание рассматриваемого объекта

В ходе выпускной квалификационной работы решаются задачи, связанные с разработкой учебного пособия к практическим занятиям по дисциплине «Прикладное программирование».

Компьютерный класс предназначен для обучения студентов различных специальностей средних специальных и высших учебных заведений, изучающих дисциплину «Прикладное программирование» Аудитория может быть использована также для обучения учащихся профессионально-технических училищ и слушателей отраслевых учебных центров повышения квалификации инженерно-технических работников.

Практические работы могут проводиться в специализированной аудитории отдела технических средств обучения ЮУрГУ, в аудитории 1-105.

5.2 Анализ вредных и опасных производственных факторов

В учебной лаборатории прямоугольной формы (8×6) площадью 48 кв.м. имеется 8 лабораторных стендов, за которыми проводятся работы.

К вредным и опасным производственным факторам относятся:

1) физические факторы:

а) метеорологические факторы: пониженная влажность воздуха и повышенная температура в помещении; это может привести к быстрому утомлению человека, снижению внимания, скорости выполняемой работы;

б) светотехнические факторы: недостаточная освещенность рабочего места студента затрудняет длительную работу, вызывает утомление, способствует развитию близорукости; спектральные характеристики используемого света влияют на психологическое и физиологическое состояние человека;

в) электромагнитные факторы: опасность поражения электрическим током, электромагнитное излучение от компьютера, повышенный уровень поля радиочастот. Источником электрических полей промышленной частоты являются токоведущие части, питающие компьютеры электроэнергией. Эти факторы негативно влияют на нервную и сердечнососудистую систему человека. Это выражается в повышенной утомляемости, снижении качества выполнения рабочих операций, болях в области сердца, изменении кровяного давления и пульса.[3]

2) психофизиологические факторы:

а) физические перегрузки (гиподинамия);

б) нервно-эмоциональные нагрузки (умственное перенапряжение, переутомление, перенапряжение зрительных, слуховых анализаторов, монотонность труда, эмоциональные перегрузки).

Все проводимые в лаборатории работы должны быть организованы таким образом, чтобы полностью исключить воздействие опасных и вредных факторов рабочей среды.

Для защиты от приведенных выше вредных и опасных производственных факторов были предприняты действия по их выявлению, а также разработаны мероприятия по защите от негативного влияния этих факторов на организм человека и достижению комфортных условий труда (рассматриваются далее).

5.3 Охрана труда

Ответственными за безопасное ведение работ являются преподаватели, проводящие занятия в аудитории, а также студенты - члены бригад, выполняющие практические работы. К выполнению работ допускаются студенты, прошедшие инструктаж, получившие допуск и ознакомленные с инструкцией по охране труда в аудитории.

Перед началом работы следует внимательно изучить содержание и порядок проведения практической работы. Необходимо подготовить к работе рабочее место, убрать посторонние предметы. В процессе проведения работы следует точно выполнять все указания преподавателя и без его разрешения не проводить самостоятельно никаких действий. Не оставлять без надзора включенное оборудование. По окончанию работы отключить компьютер, привести в порядок рабочее место.

При проведении практических работ за компьютером запрещается:

- оставлять компьютер без присмотра;
- предметы на столе не должны мешать обзору, пользованию мышкой и клавиатурой;
- недопустимо снимать корпус любой из составных частей ПК во время его работы;
- нельзя начинать работу на технике с видимым повреждением;
- в помещении с компьютерами непозволительно курить или употреблять пищу непосредственно на рабочем месте;
- при ощущении даже незначительного запаха гари, нужно как можно быстрее выключить ПК из сети и обратиться к ответственному за обслуживание компьютерной техники;
- на системном блоке не должно находиться никаких предметов, так как в результате вибраций может нарушиться работа устройства.

Для обеспечения в компьютерном классе параметров микроклимата в соответствии с «Санитарными нормами микроклимата производственных помещений» СН № 4088-86 «Отопление, вентиляция и кондиционирование воздуха» для категорий работ 1а - 1б рекомендуется применять системы вентиляции и отопления.

Для нормальной и высокопроизводительной работы в помещении аудитории необходимо, чтобы метеорологические условия (температура, влажность и скорость движения воздуха), т.е. микроклимат в определенных значениях.

В помещении аудитории были обеспечены нормативные значения параметров микроклимата - температуры воздуха, его относительной влажности и скорости движения (соответственно от 21 до 23 °С; от 40 до 60 %; 0,1 м/с - для холодно-

го периода года и от 22 до 24 °С; от 40 до 60 %; 0,2 м/с - для теплого периода года).[3]

Мероприятия по улучшению качества воздушной среды разрабатываются для конкретной аудитории с учетом реально сложившихся причин загрязнения воздуха и климатического дискомфорта (например, скопление пыли в результате присутствия статических полей, пониженная влажность из-за действия систем отопления и нагревающихся частей аппаратуры).

Для помещений высших учебных заведений нормируемая освещенность рабочих поверхностей составляет 400 лк.

5.4 Производственная санитария

5.4.1 Определение категории тяжести труда при работе на рассматриваемом объекте

Вид трудовой деятельности, тяжесть и напряженность работ устанавливаются на основе аттестации рабочих мест по условиям труда.

Выполнение практических работ не связано с систематическим мышечным напряжением. Поэтому данный вид работ относится к категории Iб относятся работы с интенсивностью энерготрат 121-150 ккал/ч (140-174 Вт).[1]

5.4.2 Достижение оптимальных параметров микроклимата для помещений рассматриваемого объекта

Для поддержания заданных значений температуры и влажности в компьютерном классе рекомендуется применять вентиляцию. Вентиляция воздуха в аудитории обеспечивается путем воздухообмена в помещении в результате действия ветрового и теплового напоров, получаемых из-за разности плотности воздуха снаружи и внутри помещения. Организованная естественная вентиляция осуществляется аэрацией. Аэрация предусматривает обмен воздуха в результате его поступления и удаления через окна, форточки и т.п.

5.4.3 Мероприятия по снижению энергетических воздействий

При оценке условий труда учитываются время воздействия электромагнитного поля и характер облучения студентов и обслуживающего персонала. Средства и методы защиты от электромагнитных полей делятся на три группы: организационные, инженерно-технические и лечебно-профилактические. Для обеспечения безопасности работ с источниками электромагнитных волн в аудитории проводится систематический контроль фактических значений нормируемых параметров. В аудитории наиболее рационально применять инженерно-технические меры защиты:

- рациональное размещение оборудования;
- расположение источников электромагнитного поля на безопасном расстоянии от рабочего места;

- использование жидкокристаллических мониторов, которые имеют меньшую интенсивность излучения по сравнению с мониторами, выполненными на основе электроннолучевых трубок.[2]

5.4.4 Мероприятия по снятию психофизиологических перегрузок

Рациональный режим труда и отдыха работников, установленный с учетом психофизиологической напряженности труда, динамики функционального состояния систем организма и работоспособности, предусматривает строгое соблюдение регламентированных перерывов. Для студентов - это перерыв между учебными занятиями, который составляет 10 минут. Для обслуживающего персонала рабочий день составляет 8 часов, основным перерывом является перерыв на обед. В соответствии с особенностями трудовой деятельности и характером функциональных изменений со стороны различных систем организма в режиме труда должны быть введены 2 - 3 регламентированных перерыва длительностью 10 минут каждый.

5.5 Эргономика и производственная эстетика

Вид трудовой деятельности, тяжесть и напряженность работ устанавливаются на основе аттестации рабочих мест по условиям труда. Выполнение практических работ можно отнести к первой категории тяжести, при этом физические усилия составляют до 174 Вт. Такие работы выполняются сидя или стоя, не требуют систематического мышечного напряжения.[4]

Таблица 5.1 - Нормативные значения эргономических параметров рабочего места

Наименование параметра	База отсчета	Нормативное значение
рабочий стол (рабочая поверхность)		
Высота, мм	полы	680 - 800 при регулировке, 725 без регулировки.
Ширина, мм	край стола	800 - 1400
Глубина, мм	передний край стола	600 - 800
рабочий стул		
Высота поверхности сиденья, мм	полы	450
Угол наклона поверхности сиденья, град	горизонтальная плоскость	5
Ширина сиденья, мм	край сиденья	400
Глубина сиденья, мм	передний край сиденья	Более 400

Окончание таблицы 5.1

Наименование параметра	База отсчета	Нормативное значение
Высота спинки стула, мм	поверхность сиденья	350
Радиус кривизны спинки стула, мм	Середина спинки горизонтальная плоскость	Более 400
Угол наклона спинки стула. град		25º

Рассчитаем освещение помещения аудитории № 1 - 105.

Оценка существующей системы искусственного освещения.

Размеры: длина А= 8 м; ширина В= 6 м; высота от светильника до рабочей поверхности Н=2,45 м. Система освещения общая равномерная, количество светильников в помещении N= 12; с количеством ламп в одном светильнике n = 4 шт.; световой поток одной лампы Ф=3000 лм.

Определяем индекс помещения i

$$i = \frac{A \cdot B}{H_П(A+B)}, \quad (5.1)$$

где A, B - длина и ширина рассматриваемого помещения;

$H_П$ - высота подвеса светильника над рабочей поверхностью.

$$i = \frac{8 \cdot 6}{2{,}45(8+6)} = 1.3$$

Следовательно, коэффициент испускания светового потока $\eta = 50\%$

Фактическая освещенность

$$E_ф = \frac{Ф \cdot N \cdot n \cdot \eta}{100 \cdot S \cdot z \cdot K_з}, \quad (5.2)$$

где S - площадь помещения

$$S = A \cdot B \quad (5.3)$$

$$S = 8 \cdot 6 = 48$$

z - коэффициент min освещенности, 1,1;

$K_з$ - коэффициент запаса, 1,5.

$$E_{\text{ф}} = \frac{3000 \cdot 12 \cdot 4 \cdot 50}{100 \cdot 48 \cdot 1{,}1 \cdot 1{,}5} = 909\text{лк}$$

Допустимая освещенность $E_{\text{н}} = 300\text{лк}$.

$$E_{\text{ф}} \geq E_{\text{н}} \tag{5.4}$$

Следовательно, освещенность аудитории достаточная и даже выше нормы. Можно уменьшить количество ламп в светильнике, для уменьшения расходов электроэнергии.

Оценка естественного освещения.

Для гигиенической оценки достаточности естественного освещения помещений определяют геометрические и светотехнические показатели.

К светотехническим показателям относится коэффициент естественного освещения.

Минимальная величина коэффициента естественного освещения

$$e_{min} = \frac{100 S_0 \tau r}{S_n \eta K_{\text{з}}}; \tag{5.5}$$

где S_0, S_n - площадь окна и пола, 5 м^2, 65 м^2 соответственно;

τ - общий коэффициент светопропускания, 0,54;

r - коэффициент, учитывающий повышение коэффициента естественной освещенности отраженного света, 3;

η - световая характеристика окна, 10;

$K_{\text{з}}$ - коэффициент запаса, 1,3.

$$e_{min} = \frac{100 \cdot 5 \cdot 0{,}54 \cdot 3}{48 \cdot 10 \cdot 1{,}3} = 1{,}2\%.$$

Допустимая освещенность $e_{\text{н}} = 1{,}2\%$

$$e_{min} = e_{\text{н}} \tag{5.6}$$

Следовательно, естественное освещение кабинета - достаточное.

5.6 Противопожарная охрана

Пожар может возникнуть при взаимодействии горючих веществ и окислителя при наличии источника зажигания. Горючими компонентами являются строительные материалы отделки помещения, двери, полы, обмотки технических деталей и т.д. Источниками зажигания могут являться электроприборы, питающие

устройства, где в связи различных нарушений могут образоваться перегретые элементы, электрические искры и дуги, способные вызывать возгорание горючих элементов. При этом опасность взрыва отсутствует.

Успех ликвидации пожара зависит главным образом от быстроты оповещения его в начале. Вследствие чего помещение оборудуется пожарной сигнализацией.

В компьютерном классе используются только твердые негорючие вещества и материалы в холодном состоянии. Следовательно, исходя из пожароопасных свойств используемых веществ, помещение аудитории относится к категории Д. При этом целесообразно использовать огнетушитель следующего типа: ОП-4(3)-АВСЕ-01.

Для предупреждения возникновения пожаров необходимо проводить инструктажи со студентами и обязательным оформлением записи в журнале инструктажей под роспись.

Для ликвидации пожаров в начальной стадии используются первичные средства устранения пожаров: огнетушители ручные и передвижные; сухой песок», асбестовые одеяла и другие.[8]

5.7 Обеспечение безопасности при угрозе чрезвычайных ситуаций

При возникновении чрезвычайных ситуаций (ЧС) решается комплекс специальных задач по ликвидации последствий, важнейшим из которых является проведение спасательных и других безотложных дел, нацеленных на спасение жизни и сохранение здоровья людей, на локализацию зон чрезвычайных ситуаций, прекращение действия характерных для нее опасных факторов. Мероприятия по подготовке и проведению спасательных и иных необходимых дел в зоне ЧС плотно связаны с событиями по обеспечиванию стойкости работы объекта. Мероприятия по повышению устойчивости работы объектов будут экономически обоснованы, если они максимально увязаны с задачами, решаемыми в период безаварийной работы объекта, улучшения условий труда, совершенствования производственного процесса.

Аудитория расположена в здании филиала. Наиболее вероятными стихийными бедствиями, которые могут возникнуть в районе расположения филиала, являются подтопление и выброс ядовитых веществ в атмосферу.

В районе расположения филиала находятся несколько предприятий, которые в своем технологическом процессе используют ядовитые вещества, такие как мышьяк и аммиак. При возникновении аварийной ситуации на предприятии возможна разгерметизация емкости для хранения ядовитых веществ; нарушение технологического процесса; террористический акт и другие, появляется опасность выброса вредных веществ в атмосферу. Облако ядовитых веществ может быстро распространиться по району, в котором расположен филиал ЮУрГУ, в результате чего может произойти массовое отравление людей.[3]

Противопожарная профилактика - комплекс организационных и технических мероприятий по предупреждению, локализации и ликвидации пожаров, а

также по обеспечению безопасной эвакуации людей и материальных ценностей в случае пожара.

Противопожарная профилактика в зданиях обеспечивается: правильным выбором огнестойкости аудитории и пределов огнестойкости отделочных элементов и конструкций; ограничением распространения огня в случае возникновения очага пожара; применением систем противодымовой защиты; безопасной эвакуацией людей; применением средств пожарной сигнализации, извещения и пожаротушения; организацией пожарной охраны.

Вывод раздела пять: для обеспечения безопасности и комфорта студентов в ходе выполнения практических работ по дисциплине «Прикладное программирование» созданы все условия, параметры освещённости, шума, рабочих мест соответствуют нормативам.

ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе было произведено сравнение платформ и языков, на основании проведенного анализа был сделан выбор в пользу платформы C Sharp, так как является многофункциональной, представляет мощные, удобные средства для написания компиляции и отладки приложений и позволяет создавать проекты различных типов.

При разработке указаний к практическим занятиям были рассмотрены рабочие программы по предшествующим дисциплинам и с учетом полученных компетенций были выбраны темы занятий: линейные, разветвляющиеся и циклические алгоритмы, работа со строковым типом данных, одномерные и многомерные массивы, графики функций.

В каждом практическом занятии приведен основной теоретический материал, рассмотрены примеры выполнения заданий и разработаны по двадцать индивидуальных заданий с вариантами исходных данных и получаемым результатам для самоконтроля.

В экономической части произведен расчет всех расходов связанных с разработкой учебного пособия. Рассчитана полная себестоимость проекта на разработку учебного пособия к практическим занятиям по дисциплине «Прикладное программирование», итог составил 73230,7руб.

В разделе безопасность жизнедеятельности рассмотрены вопросы по охране труда, расчет искусственного освещения, противопожарная охрана.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. СТО ЮУрГУ 04–2008 Стандарт организации. Курсовое и дипломное проектирование. Общие требования к содержанию и оформлению / составители: Т.И. Парубочая, Н.В. Сырейщикова, В.И. Гузеев, Л.В. Винокурова. – Челябинск: Изд-во ЮУрГУ, 2008. – 56 с.

2. ГОСТ 12.4.113–82. Издания. Система стандартов безопасности труда. Работы учебные лабораторные. Общие требования безопасности. - М.: Изд-во стандартов, 1982. - 14 с

3. Безопасность жизнедеятельности: учебное пособие / С.Н. Трофимова, В.И. Чуманов, В.А. Шишимиров. – Челябинск: Изд. ЮУрГУ, 2003. - 54 с.

4. Безопасность жизнедеятельности: учебное пособие по дипломному проектированию для студентов технических специальностей / С.П. Максимов, Т.Б. Балакина; под ред. С.Н. Трофимовой. - Челябинск: Изд. ЮУрГУ, 2005. - 55 с.

5. ГОСТ 12. 0.003-74. Система стандартов безопасности труда. Опасные и вредные производственные факторы. Классификация. - М.: Издательство стандартов, 1974.

6. Голиков, В.Н. Экономическая часть дипломного проекта: учебное пособие / В.Н. Голиков. - Челябинск: ЮУрГУ, 2000. - 46 с.

7. Основы бизнес-планирования: Методические указания по выполнению экономической части дипломных проектов для студентов технических специальностей / И.А.Баев, В.Н.Голиков, В.Г.Заслонов. - Челябинск: ЧГТУ, 2005. - 15 с.

8. Матушкина, О.Е., Некрасова, Н.В. Экономика предприятия: учебное пособие / О.Е. Матушкина, Н.В. Некрасова. - Челябинск: Изд-во ЮУрГУ, 2001. - 20 с.

9. СНиП 21-01-97 Пожарная безопасность зданий и сооружений.

10. Есипов А.С., Паньгина Н.Н., Громада М.И. Информатика: сборник задач и решений для общеобразовательных учебных заведений. - СПб.: Наука и техника, 2001. - 368 с.

11. Окулов С.М. Программирование в алгоритмах. - М.: Бином; Лаборатория знаний, 2017. - 341 с

12. Троелсен Э.Б., Язык программирования C# 5.0 и платформа .NET 4.5. – М. Вильямс, 2013. - 1312 с.

13. Вихтенко Э.М. Геометрические задачи в олимпиадах по программированию. - Изд-во МИФ-2. - № 4. - 2018

14. Павлоская Т.П., Программирование на языке высокого уровня C# // Интуит [2018]. Дата обновления: 15.09.2018. URL: http://www.intuit.ru/ studies/courses/2247/18/info (дата обращения: 12.06.2019).

15. Липский В.Г., Комбинаторика для программистов. - М.: Мир, 1988. - 200 с.

16. Демин А.Ю., Дорофеев В.А. Программирование на C#: учебное пособие. - Томск: Изд-во Томского политехнического университета, 2013. - 134 с.

17. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. - М.: Изд-во стандартов, 1990, стандартинформ, 2010 - 158 с.

18. ГОСТ 19.003-80. Единая система программной документации. Схемы алгоритмов и программ, Обозначения условные графические. - М.: Изд-во стандартов, 1982. - 14 с.