МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КРЫМ
ГБОУВО РК «КРЫМСКИЙ ИНЖЕНЕРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет экономики, менеджмента и информационных технологий

Кафедра прикладной информатики

КУРСОВОЙ ПРОЕКТ

по учебной дисциплине

«ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»

на тему: Операции над матрицами

Студентки III курса

группы И-2-15

очной формы обучения

_____ Авдиль С.Л.
(подпись)

Руководитель:

_____ ст. преп. Сейдаметов Г.С.
(подпись)

Симферополь – 2018

Аннотация

Авдиль С.Л. Приложение «Операции над матрицами».

Данный курсовой проект содержит описание основных принципов ООП, различных объектно-ориентированных языков программирования, более подробно рассмотрен язык C#. Приложение спроектировано с использованием UML-диаграмм.

Приложение «Операции над матрицами» помогает пользователям проводить различные вычисления над матрицами без потери времени.

Ключевые слова: ООП, C#, UML-диаграмма, язык программирования, Visual Studio.

Анотація

Авділь С.Л. Додаток «Операції над матрицями».

Даний курсовий проект містить опис основних принципів ООП, різних об'єктно-орієнтованих мов програмування, більш докладно розглянута мова C#. Додаток спроектовано з використанням UML-діаграм.

Додаток «Операції над матрицями» допомагає користувачам проводити різні обчислення над матрицями без втрати часу.
Ключові слова: ООП, C #, UML-діаграма, мова програмування, Visual Studio.

Abstract

Avdil S.L. Application "Operations over matrices".

This course project contains a description of the basic principles of OOP, various object-oriented programming languages, the C # language is considered in more detail. The application is designed using UML diagrams.

The "Matrix Operations" application helps users to perform various calculations on matrices without losing time.

Keywords: OOP, C #, UML-diagram, programming language, Visual Studio.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность. В настоящее время, в связи со стремительным развитием информационных технологий, компьютер занимает огромную роль в жизни человека, он помогает ему упрощать некоторые задачи, которые ему необходимо выполнить. К примеру, вычислить какие-либо сложные арифметические операции. Таким образом, возникает необходимость и в таком приложении, которое бы вычисляло различные операции над матрицами, как наиболее востребованные, так и дополнительные. Впервые в математике такое понятие как матрица появилось в середине 19 века в работах Гамильтона, Сильвестра. В настоящее время матричное исчисление встречается в различных областях математики, информатики, механики, теоретической физики и т.д.

Цель курсового проекта – разработка приложения «Операции над матрицами».

Для достижения цели были поставлены следующие **задачи**:

1. рассмотреть основные принципы ООП;
2. рассмотреть соответствующую предметную область;
3. проанализировать современные объектно-ориентированные языки программирования;
4. выбрать наиболее подходящий язык программирования для создания приложения «Операции над матрицами»;
5. разработать UML диаграммы;
6. разработать приложение;
7. создать справку;
8. создать инсталляционный пакет.

Объектом курсового проекта является объектно-ориентированное программирование.

Предметом курсового проекта является разработка приложения «Операции над матрицами».

Курсовой проект состоит из введения, двух глав, выводов, заключения и списка использованных источников. Общий объем курсового проекта: составляет 27 страниц печатного текста, включает 11 рисунков (схем и графиков), 5 приложений.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

1.1. Основные принципы ООП

В 80-х годах XX века появился такой стиль программирования как ООП (Объектно-ориентированное программирование). Как известно, в процедурных языках программирования данные и инструкции по их обработке существуют отдельно, а в ООП эта информация объединяется в единую сущность.

Объектно-программное программирование имеет свои постулаты. Принципами ООП являются его основные идеи. Наследование, полиморфизм и инкапсуляция – главные из них. Основы программирования на языках ООП заключаются в использовании объектов и классов.

Инкапсуляция — это способ скрытия внутренних деталей при предоставлении открытого интерфейса к определяемому пользователем типу [1]. Иными словами используется объединение данных и инструкций по их обработке в единую сущность, а именно класс. При написании программ на одном из языков ООП выполняется разделение между информацией внутри сущности и снаружи. Это обеспечивает безопасность данных и методов их реализации от внешних воздействий. Внешними воздействиями могут являться воздействия со стороны других классов, которые не относятся к этому объекту. Данные надежно защищены от несанкционированного доступа извне при том, что внутри сущности они успешно взаимодействуют друг с другом (Рис. 1).

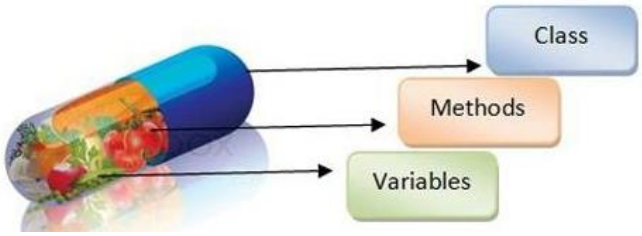

Рис. 1. Пример инкапсуляции

То есть пользователь имеет доступ только к интерфейсу объекта. Взаимодействие может происходить непосредственно через этот интерфейс, реализуемый ключевым словом: public. Закрытые данные и методы, реализующиеся посредством ключевых слов: private, protected, internal, пользователю использовать запрещается.

Зачастую сокрытие реализации применяют при [2]:

– предельной локализации изменений (в случае необходимости);

– прогнозируемости изменений (какие изменения необходимо сделать в коде, для заданного изменения функциональности) и прогнозируемости последствий изменений.

Вторым принципом ООП является *наследование*. Наследование - это способ использования одним классом методов другого класса без повторения их фактической реализации. Иными словами, использование наследования даёт возможность для описания нового класса на основе уже существующего (базового), при том, что свойства и функциональность базового (родительского) класса заимствуются новым. Это помогает избавиться от избыточности исходного кода [3]. Некоторые языки программирования различают два вида наследования: простое и множественное. Простое наследование – это наследование, в котором производный класс имеет одного предка. Множественным называют наследование, в котором производный класс может иметь более одного родителя. При множественном наследовании класс наследует методы всех предков. Достоинством данного подхода является большая гибкость.

Еще один принцип ООП – *полиморфизм*. Полиморфизмом называют возможность объектов с одинаковой спецификацией иметь различную

реализацию [4]. То есть, можно создать один интерфейс для манипуляции с объектами различной степени сложности. Этот интерфейс будет по-разному реагировать на события, и в тоже время будет происходить правильная реализация поставленных задач.

Полиморфизм поддерживается языком программирования в том случае, если классы с одинаковой спецификацией могут иметь различную реализацию. К примеру, реализация класса была изменена в процессе наследования[5].

1.2. Описание предметной области

В данном курсовом проекте предполагается разработать приложение, выполняющее различные операции над матрицами, следовательно, предметной областью будет являться теория матриц. Поскольку теория матриц нашла огромное применение в решении систем линейных уравнений и не только, целесообразно было бы создать такое приложение.

Матрицей называют прямоугольную таблицу чисел. Для удобства элементы матрицы можно пронумеровать, тогда элемент матрицы a_{ij} будет находиться на пересечении *i*-той строки и *j*-того столбца. Матрица, состоящая из *m* строк и *n* столбцов, имеет размерность *m x n* и записывается следующим образом:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

Способы обозначения матриц

Существует несколько вариантов обозначения матриц:

- круглыми скобками или двойными вертикальными линиями

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 4 & 2 \\ -1 & 3 & 1 \end{pmatrix}$$

или

$$\begin{Vmatrix} 1 & 2 & 1 \\ 0 & 4 & 2 \\ -1 & 3 & 1 \end{Vmatrix}$$

- большими буквами *A, B, C.*
- сокращенно $A= ||a_{ij}||$, *(i* = 1,2,…,*m; j* = 1,2,…,*n)* или $A= ||a_{ij}||_{m\,x\,n}$.

Виды матриц

Различают несколько видов матриц, в зависимости от размера матрицы, вида и размещения в ней элементов [6]:

1. ***Квадратная***. Матрицу называют квадратной порядка *n*, если число строк и столбцов в ней совпадают и они равны *n:*

$$A = \begin{pmatrix} 2 & -1 \\ 0 & 3 \end{pmatrix}$$

2. ***Прямоугольная.*** Матрицу называют прямоугольной, если число строк не равно числу столбцов:

$$B = \begin{pmatrix} 1 & 4 & 3 \\ -1 & 0 & -2 \end{pmatrix}$$

3. ***Вектор-строка.*** Матрица является вектор-строкой, если она состоит из одной строки:

$$C = \begin{pmatrix} -1 & 0 & 2 & 1 \end{pmatrix}$$

4. ***Вектор-столбец.*** Матрица является вектор-столбцом, если она состоит из одного столбца:

$$D = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

5. ***Скаляр*** – матрица размером 1 x 1:

K = (5)

6. ***Диагональная.*** Матрицу называют диагональной, если все элементы квадратной матрицы, кроме элементов a_{ij}, которые стоят на главной диагонали, равны нулю:

$$M = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

7. ***Единичная.*** Единичной является диагональная матрица, у которой все элементы главной диагонали – единицы. Такая матрица обозначается буквой *E:*

$$E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

8. ***Нулевая.*** Матрицу называют нулевой, если все её элементы нули.

9. ***Треугольная.*** Квадратная матрица называется треугольной, если все элементы ниже или выше главной диагонали равны нулю. Существует два вида треугольных матриц: ***верхнетреугольная*** (нули расположены ниже главной диагонали) и ***нижнетреугольная*** нули расположены выше главной диагонали).

Две матрицы *A* и *B* считаются равными, если

1. Они обе имеют одинаковый размер;
2. Их соответствующие элементы равны.

Над матрицами можно выполнять различные действия: *сложение, вычитание* и *умножение* (по аналогии с числами), а также операции, которые применяются только для матриц: *транспонирование* и *нахождение* обратной матрицы к данной.

Сложение и вычитание матриц

Суммой двух матриц $A = (a_{ij})$ и $B = (b_{ij})$ одинакового порядка называют матрицу $C = (c_{ij})$ такого же порядка, элементы которой равны сумме соответствующих элементов матриц *A* и *B*, то есть $c_{ij} = a_{ij} + b_{ij}$.

Аналогично, разностью двух матриц $A = (a_{ij})$ и $B = (b_{ij})$ одинакового порядка называют матрицу $C = (c_{ij})$ такого же порядка, элементы которой равны разности соответствующих элементов матриц *A* и *B:* $c_{ij} = a_{ij} - b_{ij}$.

Свойства операций сложения и вычитания матриц

$$A + (B + C) = (A + B) + C$$

$$A + B = B + A$$

$$A + \theta = A$$

$$A - \theta = A$$

$$A - A = \theta$$

Умножение матриц

Если количество столбцов матрицы *A (k* x *n)*равно числу строк матрицы *B (n* x *p)*, то для них определена матрица *C* размерности *(k* x *p)*, которую называют её произведением. Элементы матрицы *C* находятся по правилу: элемент c_{ij} равен сумме попарных произведений элементов *i*-той строки матрицы *A* и *j*-того столбца матрицы *B*:

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{in}b_{nj} = \sum_{s=1}^{n} a_{sn}b_{sj}$$

Свойства операции умножения матриц

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

$$k \cdot (A \cdot B) = (k \cdot A) \cdot B$$

$$(A + B) \cdot C = A \cdot C + B \cdot C$$

$$C \cdot (A + B) = C \cdot A + C \cdot B$$

Примечание: В общем случае, для произвольных матриц *A* и *B*, *A*B*≠ *B*A*. Если же это равенство выполняется (*A*B = B*A*), то матрицы *A* и *B* называются коммутативными. Единичная матрица *E* коммутативна с любой другой, то есть *E*A = A*E = A* и играет роль единицы при умножении.

Транспонирование матриц

Если в матрице *A =*(a_{ij}) размерности *m* x *n* заменить строки столбцами с соответствующими номерами, то получим транспонированную матрицу A^T размерности *n* x *m*.

Свойства транспонированной матрицы

1. Дважды транспонированная матрица равна исходной матрице

$$A^{TT} = \left(A^T\right)^T = A;$$

2. транспонированная матрица суммы равна сумме транспонированных матриц

$$(A + B)^T = A^T + B^T;$$

3. Транспонированная матрица произведения равна произведению транспонированных матриц сомножителей, взятых в обратном порядке

$$(A \cdot B)^T = B^T \cdot A^T.$$

1.3. Основные объектно-ориентированные языки программирования

В последнее время, большой популярностью среди программистов пользуются объектно-ориентированные языки программирования, поскольку с их помощью можно использовать преимущества объектно-ориентированного подхода не только на этапах проектирования и конструирования программных систем, но также и на этапах их реализации, тестирования и сопровождения.

Обзор некоторых современных объектно-ориентированных языков приводится ниже.

Microsoft Visual C++. Язык программирования C++ был разработан сотрудником AT&T Bell Laboratories, Бьерном Страустрапом. Бьерн Страуструп немногим ранее, в 1980 году, создал язык C with Classes, чьим непосредственным предшественником является C++. В свою очередь язык C with Classes, создался под сильным влиянием C и Simula. В значиельной степени C++ является надстройкой над C. В какой-то степени C++ – это улучшенный язык C, обеспечивающий контроль типов, перегрузку функций и другие полезные свойства. Однако главным критерием остаётся то, что C++ дополняет C объектной ориентированностью.

Достоинства.

- не ограниченные возможности в рамках Windows;
- довольно хороший компилятор C++, несмотря на то, что немного медленный;
- мощная библиотека MFC, отличный отладчик;
- предусмотрена правка кода в режиме отладки и последующее его выполнение без полной перекомпиляции и прерывания отладочной сессии;
- быстрее работает технология подсказок;
- полноценный браузер классов;
- исчерпывающая справочная система.

Недостатки.

- необходимо запоминать методы работы с каждым объектом;
- достаточно длинные и абсолютно непроизносимые идентификаторы;
- прежде чем заработает приложение, необходимо предварительно изучить техническую литературу.

Java. Примерно в 1995 году широкое распространение получил новый объектно-ориентированный язык программирования Java, который ориентирован на сети компьютеров и, в первую очередь, на Internet. Синтаксис данного языка похож на синтаксис языка C++, однако, несмотря на это, в остальном языки очень сильно отличаются. Java является интерпретируемым на конечной стадии языком: для него определены внутреннее представление, то есть прекомпиляция в байткод (bytecode) и постинтерпретатор этого представления на целевой машине, реализованные уже сегодня на большинстве платформ. Положительные свойства технологии Java разрешают использовать ее для программ, которые распространены по сетям (в частности, по сети Internet).

Достоинства:

- высокая безопасность;

- огромный выбор бесплатных программных библиотек к языку, которые написаны программистами всего мира;
- компиляция приложений в специальный байт-код и их выполнение в специальной виртуальной машине на любом оборудовании и в любой операционной системе.

Недостатки:

- огромная нагрузка на оперативную память оборудования;
- у продуктов данного языка время выполнения одних и тех же задач в несколько раз медленнее, чем например на C.

Visual Basic .NET (VB.NET) — это объектно-ориентированный язык программирования, который реализован на платформе .NET, рассматривающийся как очередной виток эволюции Visual Basic (VB). Основным отличием от «классического» VB является то, что VB.NET — полностью объектно-ориентированный язык, который также поддерживает полиморфизм, наследование и другие основные принципы ООП.

Достоинства:

- отличается высокой скоростью разработки графических программ для Windows;
- имеет простой синтаксис;
- дает возможность отлаживать и редактировать код без перекомпиляции при приостановке программы;
- защищает от ошибок, которые связаны доступом к памяти;
- дает возможность использовать WinAPI.

Недостатки:

- поддерживает операционные системы Windows и Mac OS;
- невысокая скорость работы.

1.4. Язык программирования C# (Sharp)

C# — это один из самых распространённых на сегодняшний день объектно-ориентированных языков программирования. Был разработан группой инженеров под руководством Андерса Хейлсберга в 1998—2001 годах, в компании Microsoft как основной язык разработки приложений для платформы Microsoft .NET. Сторонники C# считают, что он наиболее мультипарадигменный, универсальный, продвинутый и удобный в использовании язык программирования, чем другие. Поскольку за данным языком стоит платформа Microsoft .NET, число сторонников быстро увеличивается. Потому как компилятор с C# входит в стандартную установку самой .NET, то имеется возможность создавать и компилировать программы на нём даже без инструментальных средств, таких как Visual Studio. C# имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства и т.д. Поскольку C# имеет таких предшественников, как язык C++, Java, Delphi, Модула и Smalltalk, опираясь на практику их использования, он исключает некоторые модели, которые плохо зарекомендовали себя при разработке программных систем. Например, C# не поддерживает множественное наследование классов, в отличие от C++.

C# поддерживает все основные принципы объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм. Кроме этого, в данном языке реализована автоматическая «сборка мусора», обработка исключений, а также динамическое связывание.

Выделяют несколько особенностей языка программирования C#. Рассмотрим некоторые из них. Данный язык разрабатывался как язык программирования прикладного уровня для Common Language Runtime (CLR) и, как таковой, зависит, прежде всего, от возможностей самой CLR. Прежде всего, это касается системы типов C#, отражающей Base Class Library (BCL). В зависимости от того, может ли быть транслирована в соответствующие конструкции CLR конкретная языковая особенность,

зависит то, будут ли присутствовать или отсутствовать те или иные выразительные особенности языка.

Сегодня, язык программирования C# считается одним из самых мощных, быстро развивающихся и востребованных языков в IT-сфере. В настоящий момент на нем пишутся самые разные приложения: начиная от небольших программ для персонального компьютера и заканчивая разработкой крупных веб-порталов и веб-сервисов, которые ежедневно обслуживают миллионы пользователей.

По сравнению с другими языками программирования, C# был создан не так давно, однако, несмотря на это, он пользуется большой популярностью и является одним из наиболее мощных языков на сегодняшний день. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 7.0, которая вышла 7 марта 2017 года вместе с Visual Studio 2017.

Роль платформы .NET

Фреймворк .NET представляет собой мощную платформу для создания приложений. Выделяют следующие основные черты этой платформы:

1. Поддержка нескольких языков. Основой данной платформы является то, что .NET поддерживает несколько языков благодаря общеязыковой среде исполнения Common Language Runtime (CLR). Поддерживаемые языки: вместе с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. Во время компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language), который считается своего рода ассемблером платформы .NET. Отсюда следует, что существует возможность сделать отдельные модули одного приложения на отдельных языках.

2. Кроссплатформенность. .NET переносимая платформа с некоторыми ограничениями. К примеру, последняя версия платформы, на сегодняшний день .NET Framework, поддерживается на большинстве современных ОС Windows (Windows 10/8.1/8/7/Vista). А проект Mono позволяет создавать

приложения, работающих и на других ОС семейства Linux, а также на мобильных платформах, таких как Android и iOS.

3. Мощная библиотека классов. В .NET представляется единая для всех поддерживаемых языков библиотека классов. И независимо от того, какое приложение нам необходимо написать на C#(чат, текстовый редактор, сложный веб-сайт), мы в любом случае задействуем библиотеку классов .NET.

4. Разнообразие технологий. Основой для целого стека технологий, задействованные разработчиками при построении различных приложений, является базовая библиотека классов и общеязыковая среда исполнения CLR. Так, работая с базами данных в этом стеке технологий, используют технологию ADO.NET, для построения графических приложений с богатым насыщенным интерфейсом – технологию WPF, а для создания веб-сайтов – ASP.NET и т.д.

Немаловажной особенностью языка C# и фреймворка .NET является автоматическая сборка мусора. То есть, нет необходимости в освобождении памяти как в C++. Вышеупомянутая общеязыковая среда CLR автоматически вызывает сборщик мусора и очищает память.

Итак, C# является одним из ведущих языков программирования на сегодняшний день, поскольку он довольно универсальный, продвинутый и удобный в использовании. Немаловажную роль в востребованности языка играет платформа .NET, которая представляет собой мощную платформу для создания приложений.

Вывод к первой главе

Наиболее востребованными на сегодняшний день языками программирования являются объектно-ориентированные. Это обусловлено тем, что в них используются такие основополагающие концепции как: наследование, инкапсуляция, полиморфизм и абстракция.

Практически все объектно-ориентированные языки программирования являются развивающимися, их стандарты регулярно обновляются и

расширяются. Вследствие этого развития во входных языках компиляторов различных систем программирования происходят неизбежные различия. Каждый язык имеет свои достоинства и недостатки, а отсюда и различные области эффективного применения. Это объясняет необходимость в изучении и освоении нескольких различных языков программирования.

Для разработки приложения «Операции над матрицами» был выбран язык программирования C#, поскольку он является универсальным, удобным в использовании, а также одним из ведущих языков программирования на сегодняшний день.

ГЛАВА 2. РАЗРАБОТКА ПРИЛОЖЕНИЯ «ОПЕРАЦИИ НАД МАТРИЦАМИ»

2.1. Модель приложения

При запуске программы пользователь может наблюдать две кнопки выбора. Перейдя по первой кнопке, пользователь может ввести размерность матриц, после чего появятся ячейки в них, заполнить эти матрицы значениями и провести основные операции над ними, такие как: сложение, вычитание и умножение, также является возможным нажав на кнопку в нижнем правом углу выйти в главное меню. Выбрав вторую кнопку, пользователь переходит на форму, в которой тем же способом может заполнить матрицы, а затем провести различные операции с каждой из них, получив необходимое значение, которое будет выводиться либо в дополнительной матрице, либо в текстовой ячейке, в зависимости от поставленной задачи.

Интерфейс программы будет состоять из трех форм. Первая форма будет содержать три кнопки: «Выполнить основные операции над матрицами», «Выполнить дополнительные операции над матрицами», а также кнопку «Выход». На второй форме, которая появится при нажатии первой кнопки на главной форме, имеются: две ячейки для ввода размерности матриц, сами матрицы, кнопки «A+B» (сложение двух матриц), «A-B» (вычитание двух матриц), «A*B» (перемножение двух матриц) и еще одна кнопка «Главное меню», которая позволяет вернуться на главную форму. Третья форма имеет выход на главную форму, с помощью кнопки «Главное меню», такую же структуру ввода матриц, как и вторая форма, а также имеет дополнительные кнопки, такие как: кнопка «Умножить на » (умножение матрицы на число), кнопка «Возвести в степень» (возведение матрицы в указанную степень), кнопка «Транспонировать матрицу», кнопка «Найти определитель», кнопка «Найти ранг матрицы», а также кнопка «Проверка на единичность».

2.2. Проектирование приложения «Операции над матрицами» с использованием UML-диаграмм

UML – это графический язык моделирования общего назначения, предназначенный для спецификации, визуализации, проектирования и документирования всех артефактов, создаваемых при разработке программных систем [7].

Главные цели в разработке UML:

- предоставить пользователям готовый к использованию выразительный язык визуального моделирования, позволяющий им разрабатывать осмысленные модели и обмениваться ими;
- предусмотреть механизмы расширяемости и специализации для расширения базовых концепций;
- обеспечить формальную основу для понимания этого языка моделирования (язык должен быть одновременно точным и доступным для понимания, без лишнего формализма);
- обеспечить независимость от конкретных языков программирования и процессов разработки.
- стимулировать рост рынка объектно-ориентированных инструментальных средств;
- интегрировать лучший практический опыт.

В данном курсовом проекте с помощью UML диаграмм было описано взаимодействие и поведение пользователя с системой приложения.

2.2.1. Диаграмма вариантов использования

Диаграммы вариантов использования описывают взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующими в процессе. Они предназначены для упрощения взаимодействия с будущими пользователями системы, с клиентами, и особенно пригодятся для определения необходимых характеристик системы.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов которые система предоставляет актеру.

Вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его описание, обозначающее выполнение какой-либо операции или действия.

В данном курсовом проекте пользователю доступны следующие действия: «Войти в приложение», «Выполнить основные операции над матрицами», «Выполнить дополнительные операции над матрицами», «Выйти из приложения» (Рис. 2.1).

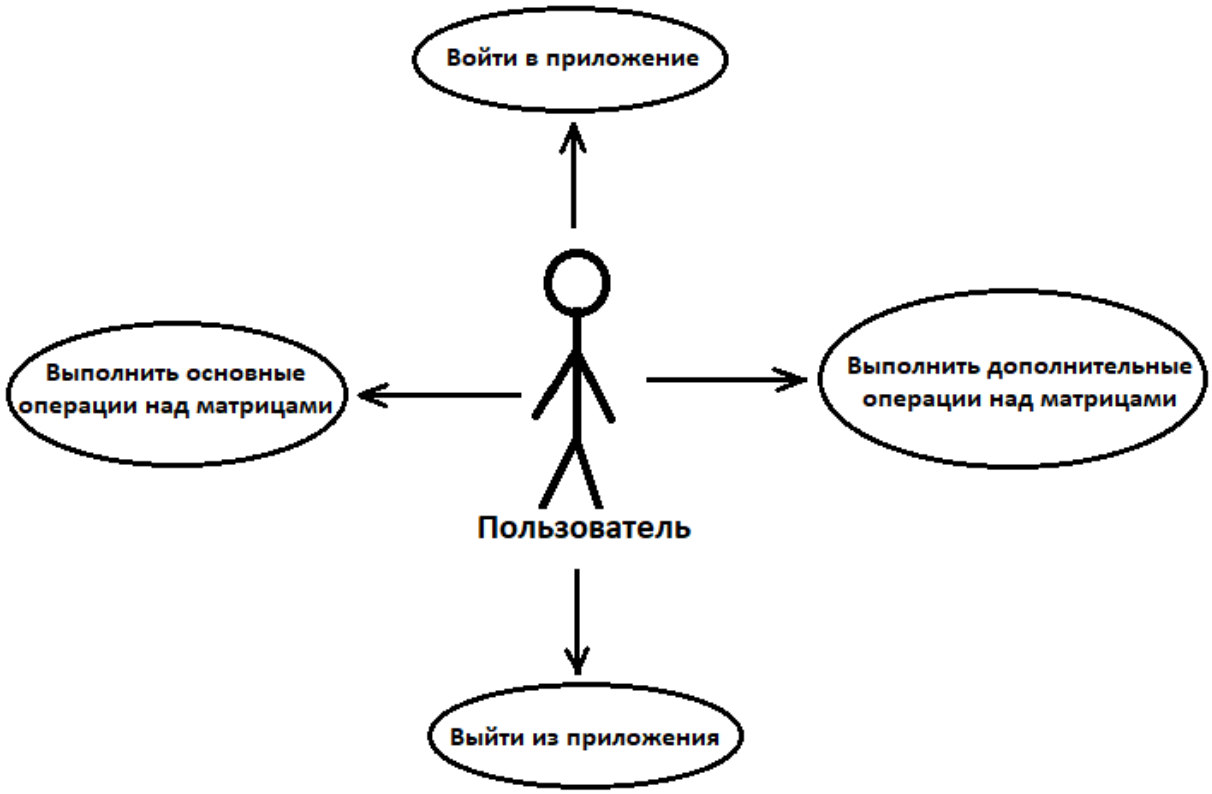

Рис.2.1. Диаграмма вариантов использования

2.2.2. Диаграмма деятельности

Диаграммы деятельности используют для моделирования динамических аспектов поведения системы. Они в общем случае состоят из: состояний деятельности и состояний действия; переходов; элементов ветвлений. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами – переходы от одного состояния действия к другому. Графически состояние действия изображается фигурой, напоминающей прямоугольник, боковые стороны которого заменены выпуклыми дугами. Внутри этой фигуры записывается выражение действия, которое должно быть уникальным в пределах одной диаграммы деятельности. Действие может быть записано на естественном языке, некотором псевдокоде или языке программирования. Ветвление на диаграмме деятельности обозначается небольшим ромбом, внутри которого нет никакого текста. В языке UML для распараллеливания вычислений используется специальный символ для разделения и слияния параллельных вычислений или потоков управления. Разработанная диаграмма деятельности для приложения «Операции над матрицами» изображена на рисунке 2.2.

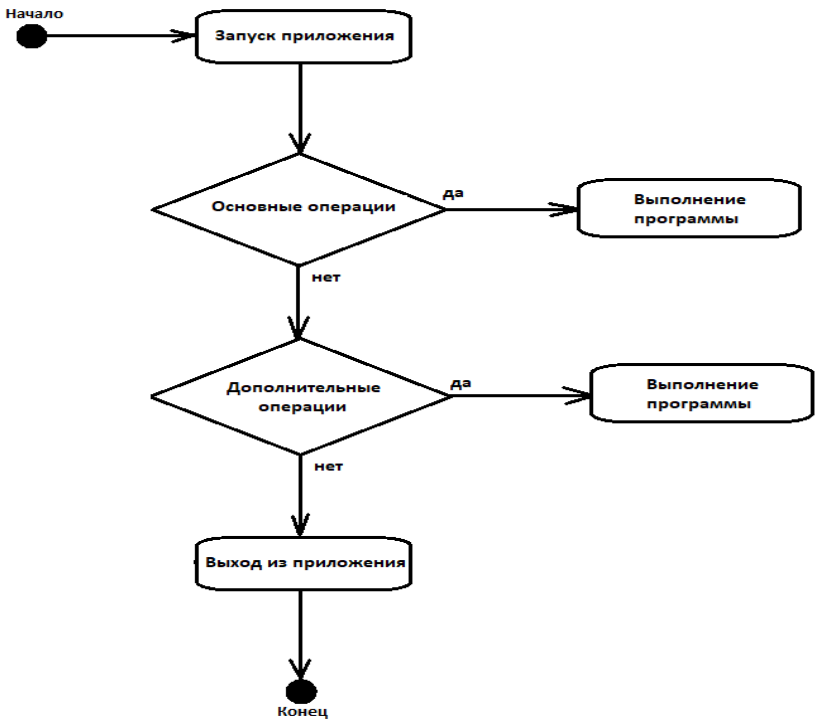

Рис.2.2. Диаграмма деятельности

2.2.3. Диаграмма классов

Диаграмма классов демонстрирует классы системы, их атрибуты, методы и взаимосвязи между ними. Диаграмма классов является ключевым элементом в объектно-ориентированном моделировании. На диаграмме классы представлены в рамках, содержащих три компонента:

- В верхней части написано имя класса. Имя класса выравнивается по центру и пишется полужирным шрифтом. Имена классов начинаются с заглавной буквы. Если класс абстрактный – то его имя пишется полужирным курсивом.
- Посередине располагаются поля (атрибуты) класса. Они выровнены по левому краю и начинаются с маленькой буквы.
- Нижняя часть содержит методы класса. Они также выровнены по левому краю и пишутся с маленькой буквы.

Разработанная диаграмма классов изображена на рисунке 2.3.

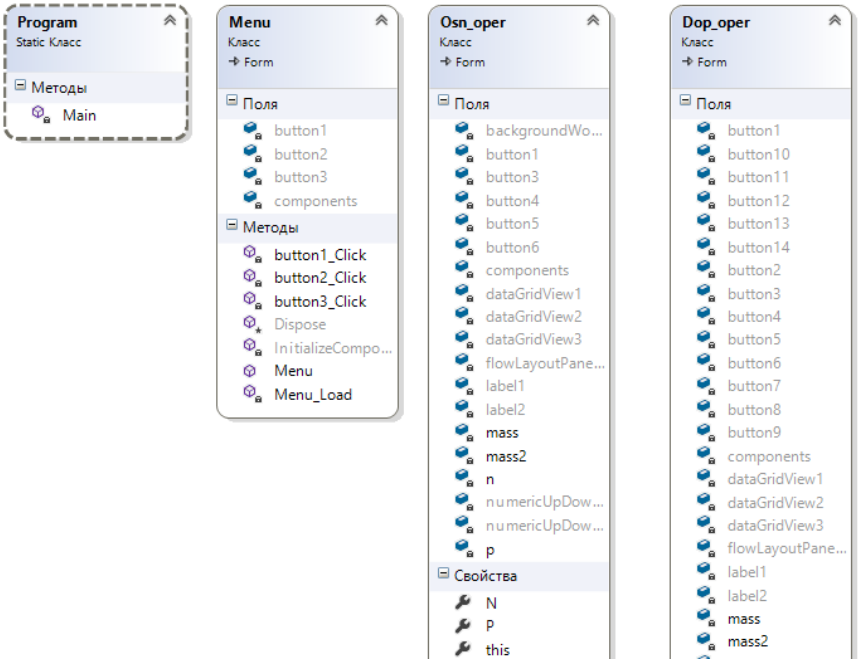

Рис.2.3. Диаграмма классов

2.3. Программирование приложения

Для программной реализации необходимо установить IDE Visual Studio. После установки запустить и создать новый проект (Приложение 5. Рис. 1).

Чтобы создать новый класс, необходимо в обозревателе решений добавить новый элемент (Приложение 5. Рис. 2).

Таким же способом можно создать Форму Windows Forms (Приложение 5. Рис 3).

После добавления класса необходимо написать программный код (Приложение 5. Рис. 4).

После написания всех классов можно скомпилировать и запустить проект.

В курсовом проекте реализовано 4 класса: Program, Menu, Osn_oper, Dop_oper. Рассмотрим подробное описание данных классов.

1. Program – главный класс программы, с помощью которого выполняется запуск приложения. Программный код описан в ПРИЛОЖЕНИИ 1. Данный класс содержит единственный метод Main() – главный метод программы.

2. Menu – класс, который предназначен для выбора необходимых операций над матрицами. Программный код описан в ПРИЛОЖЕНИИ 2. Класс Menu содержит следующие методы:

- Menu_Load() – который устанавливает основные свойства главной формы;
- InitializeComponent() – метод, который используется для
- инициализации объектов;
- Close() – метод, который закрывает приложение;
- Show() – метод, отображающий необходимую форму;
- Hide() – метод, закрывающий указанную форму;
- button1_Click, button2_Click, button3_Click – обработчики различных событий (входа во вторую форму, входа в третью форму, закрытие основной формы).

3. Osn_oper – класс, который предназначен для вычисления основных операций над матрицами. Программный код описан в ПРИЛОЖЕНИИ 3. Класс Osn_oper содержит следующие методы:

- InitializeComponent();
- Osn_oper_Load() – установление основных свойств данной формы;
- button1_Click(), button3_Click_1(), button4_Click(), button5_Click(), button6_Click() – обработчики таких событий как: установление размерности матриц, открытие основной формы, сложение матриц, вычитание и умножение.

4. Dop_oper – класс, который предназначен для вычисления дополнительных операций над матрицами. Программный код описан в ПРИЛОЖЕНИИ 4. Класс содержит следующие методы:

- InitializeComponent();
- Dop_oper_Load() – установление основных свойств данной формы;
- button1_Click() – обработчик события, для установления размерности матрицы;
- button6_Click(), button14_Click() – умножение различных матриц на введенное число;
- button5_Click(), button13_Click() – возведение матриц ы указанную степень;
- button4_Click(), button12_Click() – транспонирование различных матриц;
- button8_Click(), button11_Click() – нахождение определителя;
- button7_Click(), button10_Click() – проверка на единичность каждой матрицы;
- button3_Click(), button9_Click() – нахождение ранга введенных матриц.

2.4. Создание инсталляционного пакета проекта

Инсталляционный пакет для текущего проекта был создан в программе Smart Install Maker. Поскольку данная программа имеет достаточно понятный и удобный интерфейс.

Принцип работы с инсталляционным пакетом:

1) Запуск Smart Install Maker (Приложение 5. Рис. 5)

Выбираем название пакета, указываем папку проекта с игрой, запускаем инсталляцию (Приложение 5. Рис. 6).

После завершения компиляции, выходит окно (Приложение 5. Рис. 7):

Вывод ко второй главе

Во второй главе была описана модель приложения, осуществлено проектирование с помощью UML диаграмм. Также были рассмотрены и созданы различные виды UML-диаграмм, а именно:

- Диаграмма вариантов использования;
- Диаграмма деятельности;
- Диаграмма классов.

Спроектировав необходимые диаграммы, было написано приложение «Операции над матрицами», содержащее 4 класса: Program, Menu, Osn_oper, Dop_oper.

Последним этапом проектирования являлось создание инсталляционного пакета проекта, необходимого для того, чтобы пользователь мог установить приложение «Операции над матрицами» на свой компьютер.

ЗАКЛЮЧЕНИЕ

В ходе написание курсового проекта «Операции над матрицами» были решены следующие задачи:

Были рассмотрены основные принципы ООП и необходимость в их использовании на сегодняшний день. Также была рассмотрена предметная область Матрицы, выявлены основные ее свойства и правила использования операций над ними.

Были проанализированы основные современные объектно-ориентированные языки программирования, в следствии чего, для создания приложения «Операции над матрицами» был выбран язык C#, поскольку на сегодняшний день он является одним из ведущих языков программирования, его считают довольно универсальным, продвинутым и удобным в использовании. Немаловажную роль в востребованности языка играет платформа .NET, которая представляет собой мощную платформу для создания приложений.

Для удобства использования и визуального отображения работы приложения «Операции над матрицами» были рассмотрены и созданы три UML-диаграммы: диаграмма вариантов использования, диаграмма деятельности и диаграмма классов.

Один из важнейших этапов написания данного курсового проекта – разработка приложения, в котором описывается вся программная реализация.

Последним этапом написания курсового проекта являлось создание инсталляционного пакета, который необходим для установки приложения «Операции над матрицами».

Список использованных источников

1. Пол А. Объектно-ориентированное программирование на C++. – А.Пол. – 2001. – 476с.
2. Основные принципы ООП: инкапсуляция, наследование, полиморфизм. – Электронный ресурс. URL: http://mylektsii.ru/14-6957.html. (дата обращения: 20.04.2018)
3. Основные принципы ООП и их использование. – Эелектронный ресурс. URL: http://fb.ru/article/91926/osnovnyie-printsipyi-oop-i-ih-ispolzovanie. (дата обращения: 20.04.2018).
4. Использование полиморфизма в объектно-ориентированном программировании. – Электронный ресурс. URL: http://mykonspekts.ru/2-298.html. (дата обращения: 20.04.2018)
5. Инкапсуляция, наследование, полиморфизм. – Электронный ресурс. URL: http://codrob.ru/lesson/26. (дата обращения: 20.04.2018)
6. Матрицы. Виды матриц. – Электронный ресурс. URL: http://matworld.ru/matrix/matrix.php. (дата обращения: 26.04.2018).
7. UML2 и ER-диаграммы. – Электронный ресурс. URL: https://proft.me/2013/05/26/uml-2-tipy-diagramm/. (дата обращения: 27.04.2018)

ПРИЛОЖЕНИЕ 1

Класс «Program»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MatrixCS
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Menu());
        }
    }
}
```

ПРИЛОЖЕНИЕ 2

Класс «Menu»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MatrixCS
{
    public partial class Menu : Form
    {
        public Menu()
        {
            InitializeComponent();
        }

        private void Menu_Load(object sender, EventArgs e)
        {
            this.Text = "Главное меню";
            this.BackgroundImage = Image.FromFile("F:\\11.jpg");
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Osn_oper a = new Osn_oper();
            a.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Dop_oper a = new Dop_oper();
            a.Show();
            this.Hide();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

ПРИЛОЖЕНИЕ 3

Класс «Osn_oper»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MatrixCS
{
    public partial class Osn_oper : Form
    {
        private int n;
        private int p;
        private int[,] mass;
        private int[,] mass2;

        public Osn_oper()
        {
            InitializeComponent();
        }

        public int N
        {
            get { return n; }
            set { if (value > 0) n = value; }
        }
        public int P
        {
            get { return p; }
            set { if (value > 0) p = value; }
        }

        public Osn_oper(int n, int p)
        {
            this.n = n;
            this.p = p;
            mass = new int[this.n, this.p];
        }
        public int this[int n, int p]
        {
            get
            {
                return mass[n, p];
            }
            set
            {
                mass[n, p] = value;
            }
        }



        private void button1_Click(object sender, EventArgs e)
        {
            n = Convert.ToInt32(numericUpDown1.Text);
            p = Convert.ToInt32(numericUpDown2.Text);
            dataGridView1.RowCount = n;
            dataGridView1.ColumnCount = p;
```

```
        dataGridView2.RowCount = n;
        dataGridView2.ColumnCount = p;
        dataGridView3.RowCount = n;
        dataGridView3.ColumnCount = p;
        dataGridView1.Visible = true;
        dataGridView2.Visible = true;
        dataGridView3.Visible = true;

    }

    private void Osn_oper_Load(object sender, EventArgs e)
    {
        this.Text = "Выполнение основных операций";
        this.BackgroundImage = Image.FromFile("F:\\11.jpg");
    }

    private void button3_Click_1(object sender, EventArgs e)
    {
        Menu a = new Menu();
        this.Hide();
        a.Show();
    }

    private void button6_Click(object sender, EventArgs e)
    {
        mass = new int[n, p];
        mass2 = new int[n, p];
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < p; j++)
            {
                mass[i, j] = Convert.ToInt32(dataGridView1[i, j].Value);
                mass2[i, j] = Convert.ToInt32(dataGridView2[i, j].Value);
            }
        }

        int[,] resMass = new int[n, p];

        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < p; j++)
            {
                resMass[i, j] = mass[i, j] + mass2[i, j];
            }
        }
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < p; j++)
            {
                dataGridView3[i, j].Value = resMass[i, j];
            }
        }
    }
    private void button5_Click(object sender, EventArgs e)
    {
        mass = new int[n, p];
        mass2 = new int[n, p];
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < p; j++)
            {
                mass[i, j] = Convert.ToInt32(dataGridView1[i, j].Value);
                mass2[i, j] = Convert.ToInt32(dataGridView2[i, j].Value);
            }
        }
```

```
            int[,] resMass = new int[n, p];

            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    resMass[i, j] = mass[i, j] - mass2[i, j];
                }
            }
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    dataGridView3[i, j].Value = resMass[i, j];
                }
            }
        }
        private void button4_Click(object sender, EventArgs e)
        {
            int[,] resMass = new int[n, p];
            mass = new int[n, p];
            mass2 = new int[n, p];
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    mass[i, j] = Convert.ToInt32(dataGridView1[i, j].Value);
                    mass2[i, j] = Convert.ToInt32(dataGridView2[i, j].Value);
                }
            }
            for (int i = 0; i < n; i++)
                for (int j = 0; j < p; j++)
                    for (int k = 0; k < p; k++)
                        resMass[i, j] += mass[i, j] * mass2[i, j];
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    dataGridView3[i, j].Value = resMass[i, j];
                }
            }
        }
    }
}
```

ПРИЛОЖЕНИЕ 4

Класс «Dop_oper»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MatrixCS
{
    public partial class Dop_oper : Form
    {
        private int n;
        private int p;
        private int[,] mass;
        private int[,] mass2;

        public Dop_oper()
        {
            InitializeComponent();
        }

        public int N
        {
            get { return n; }
            set { if (value > 0) n = value; }
        }
        public int P
        {
            get { return p; }
            set { if (value > 0) p = value; }
        }

        public Dop_oper(int n, int p)
        {
            this.n = n;
            this.p = p;
            mass = new int[this.n, this.p];
        }
        public int this[int n, int p]
        {
            get
            {
                return mass[n, p];
            }
            set
            {
                mass[n, p] = value;
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Menu a = new Menu();
            this.Hide();
            a.Show();
        }



        private void button1_Click(object sender, EventArgs e)
```

```
{
    n = Convert.ToInt32(numericUpDown1.Text);
    p = Convert.ToInt32(numericUpDown2.Text);
    dataGridView1.RowCount = n;
    dataGridView1.ColumnCount = p;
    dataGridView2.RowCount = n;
    dataGridView2.ColumnCount = p;
    dataGridView3.RowCount = n;
    dataGridView3.ColumnCount = p;
    dataGridView1.Visible = true;
    dataGridView2.Visible = true;
    dataGridView3.Visible = true;

}

private void Dop_oper_Load(object sender, EventArgs e)
{
    this.Text = "Выполнение дополнительных операций";
    this.BackgroundImage = Image.FromFile("F:\\11.jpg");
}

private void button6_Click(object sender, EventArgs e)
{
    int a = Convert.ToInt32(textBox1.Text);
    int[,] resMass = new int[n, p];
    mass = new int[n, p];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            mass[i, j] = Convert.ToInt32(dataGridView1[i, j].Value);
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            resMass[i, j] = mass[i, j] * a;
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            dataGridView3[i, j].Value = resMass[i, j];
        }
    }

}

private void button14_Click(object sender, EventArgs e)
{
    int a = Convert.ToInt32(textBox4.Text);
    int[,] resMass = new int[n, p];
    mass2 = new int[n, p];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            mass2[i, j] = Convert.ToInt32(dataGridView2[i, j].Value);
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            resMass[i, j] = mass2[i, j] * a;
        }
    }
```

```
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            dataGridView3[i, j].Value = resMass[i, j];
        }
    }
}

private void button5_Click(object sender, EventHandler e)
{
    int a = Convert.ToInt32(textBox2.Text);
    int[,] resMass = new int[n, p];
    mass = new int[n, p];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            mass[i, j] = Convert.ToInt32(dataGridView1[i, j].Value);
        }
    }

    if (n == p)
    {
        if (a == 0)
        {
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    if (i == j)
                    {
                        resMass[i, j] = 1;
                    }
                    else
                    {
                        resMass[i, j] = 0;
                    }

                }
            }
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    dataGridView3[i, j].Value = resMass[i, j];
                }
            }
        }
    }

    if (a % 2 == 1)
    {
        for (int y = 0; y < a; y++)
        {
            for (int i = 0; i < n; i++)
                for (int j = 0; j < p; j++)
                    for (int k = 0; k < p; k++)
                        resMass[i, j] += mass[i, k] * mass[k, j];
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            dataGridView3[i, j].Value = resMass[i, j];
        }
    }
}
```

```
private void button13_Click(object sender, EventArgs e)
{
    int a = Convert.ToInt32(textBox3.Text);
    int[,] resMass = new int[n, p];
    mass2 = new int[n, p];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            mass2[i, j] = Convert.ToInt32(dataGridView2[i, j].Value);
        }
    }

    if (n == p)
    {
        if (a == 0)
        {
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    if (i == j)
                    {
                        resMass[i, j] = 1;
                    }
                    else
                    {
                        resMass[i, j] = 0;
                    }

                }
            }
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    dataGridView3[i, j].Value = resMass[i, j];
                }
            }
        }
    }

    if (a % 2 == 1)
    {
        for (int y = 0; y < a; y++)
        {
            for (int i = 0; i < n; i++)
                for (int j = 0; j < p; j++)
                    for (int k = 0; k < p; k++)
                        resMass[i, j] += mass2[i, k] * mass2[k, j];
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            dataGridView3[i, j].Value = resMass[i, j];
        }
    }
}

private void button4_Click(object sender, EventArgs e)
{
    int[,] resMass = new int[n, p];
    mass = new int[n, p];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
```

```
                {
                    mass[i, j] = Convert.ToInt32(dataGridView1[i, j].Value);
                }
            }
            for (int i = 0; i < p; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    resMass[i, j] = mass[j, i];
                }
            }
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    dataGridView3[i, j].Value = resMass[i, j];
                }
            }
        }

        private void button12_Click(object sender, EventArgs e)
        {
            int[,] resMass = new int[n, p];
            mass2 = new int[n, p];
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    mass2[i, j] = Convert.ToInt32(dataGridView2[i, j].Value);
                }
            }
            for (int i = 0; i < p; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    resMass[i, j] = mass2[j, i];
                }
            }
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    dataGridView3[i, j].Value = resMass[i, j];
                }
            }
        }

        private void button8_Click(object sender, EventArgs e)
        {
            mass = new int[n, p];
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    mass[i, j] = Convert.ToInt32(dataGridView1[i, j].Value);
                }
            }
            if (n != p) throw new Exception(" Число строк в матрице не совпадает с числом
столбцов");
            int[,] buf = new int[n - 1, p - 1];
            for (int i = 0; i < n; i++)
                for (int j = 0; j < p; j++)
                {
                    if ((i != dataGridView1.RowCount) || (j != dataGridView1.ColumnCount))
                    {
                        if (i > dataGridView1.RowCount && j < dataGridView1.ColumnCount) buf[i
- 1, j] = mass[i, j];
                        if (i < dataGridView1.RowCount && j > dataGridView1.ColumnCount)
buf[i, j - 1] = mass[i, j];
```

```
                        if (i > dataGridView1.RowCount && j > dataGridView1.ColumnCount) buf[i
- 1, j - 1] = mass[i, j];
                        if (i < dataGridView1.RowCount && j < dataGridView1.ColumnCount)
buf[i, j] = mass[i, j];
                    }
                }
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    dataGridView3[i, j].Value = buf[i, j];
                }
            }

            if (n != p) throw new Exception(" Число строк в матрице не совпадает с числом
столбцов");
            double det = 0;
            int Rank = n;
            if (Rank == 1) det = mass[0, 0];
            if (Rank == 2) det = mass[0, 0] * mass[1, 1] - mass[0, 1] * mass[1, 0];
            if (Rank > 2)
            {
                for (int j = 0; j < p; j++)
                {
                    det += Math.Pow(-1, 0 + j) * mass[0, j];
                }
            }
            richTextBox1.Text = Convert.ToString(det);
        }

        private void button11_Click(object sender, EventArgs e)
        {
            mass2 = new int[n, p];
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    mass2[i, j] = Convert.ToInt32(dataGridView2[i, j].Value);
                }
            }
            if (n != p) throw new Exception(" Число строк в матрице не совпадает с числом
столбцов");
            int[,] buf = new int[n - 1, p - 1];
            for (int i = 0; i < n; i++)
                for (int j = 0; j < p; j++)
                {
                    if ((i != dataGridView2.RowCount) || (j != dataGridView2.ColumnCount))
                    {
                        if (i > dataGridView2.RowCount && j < dataGridView2.ColumnCount) buf[i
- 1, j] = mass2[i, j];
                        if (i < dataGridView2.RowCount && j > dataGridView2.ColumnCount)
buf[i, j - 1] = mass2[i, j];
                        if (i > dataGridView2.RowCount && j > dataGridView2.ColumnCount) buf[i
- 1, j - 1] = mass2[i, j];
                        if (i < dataGridView2.RowCount && j < dataGridView2.ColumnCount)
buf[i, j] = mass2[i, j];
                    }
                }
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < p; j++)
                {
                    dataGridView3[i, j].Value = buf[i, j];
                }
            }

            if (n != p) throw new Exception(" Число строк в матрице не совпадает с числом
столбцов");
            double det = 0;
```

```
    int Rank = n;
    if (Rank == 1) det = mass2[0, 0];
    if (Rank == 2) det = mass2[0, 0] * mass2[1, 1] - mass2[0, 1] * mass2[1, 0];
    if (Rank > 2)
    {
        for (int j = 0; j < p; j++)
        {
            det += Math.Pow(-1, 0 + j) * mass2[0, j];
        }
    }
    richTextBox1.Text = Convert.ToString(det);
}

private void button7_Click(object sender, EventArgs e)
{
    int count = 0;
    mass = new int[n, p];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            mass[i, j] = Convert.ToInt32(dataGridView1[i, j].Value);
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            if (mass[i, j] == 1 && i == j)
            {
                count++;
            }
        }
    }
    if (count == n)
    {
        richTextBox1.Text = Convert.ToString("Единичная!");
    }
    else richTextBox1.Text = Convert.ToString("Не единичная!");
}

private void button10_Click(object sender, EventArgs e)
{
    int count = 0;
    mass2 = new int[n, p];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            mass2[i, j] = Convert.ToInt32(dataGridView2[i, j].Value);
        }
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < p; j++)
        {
            if (mass2[i, j] == 1 && i == j)
            {
                count++;
            }
        }
    }
    if (count == n)
    {
        richTextBox1.Text = Convert.ToString("Единичная!");
    }
    else richTextBox1.Text = Convert.ToString("Не единичная!");
}
```

```
    private void button3_Click(object sender, EventArgs e)
    {
        int rang = 0;
        int q = 1;

        while (q <= MinValue(n, p))
        {
            int[,] resMass = new int[n, p];
            for (int i = 0; i < (n - (q - 1)); i++)
            {
                for (int j = 0; j < (p - (q - 1)); j++)
                {
                    for (int k = 0; k < q; k++)
                    {
                        for (int c = 0; c < q; c++)
                        {
                            resMass[k, c] = mass[i + k, j + c];
                        }
                    }
                    rang = q;
                }
            }
            q++;
        }

        richTextBox1.Text = Convert.ToString(rang);
    }

    private static int MinValue(int a, int b)
    {
        if (a >= b)
            return b;
        else
            return a;
    }

    private void button9_Click(object sender, EventArgs e)
    {
        int rang = 0;
        int q = 1;

        while (q <= MinValue(n, p))
        {
            int[,] resMass = new int[n, p];
            for (int i = 0; i < (n - (q - 1)); i++)
            {
                for (int j = 0; j < (p - (q - 1)); j++)
                {
                    for (int k = 0; k < q; k++)
                    {
                        for (int c = 0; c < q; c++)
                        {
                            resMass[k, c] = mass2[i + k, j + c];
                        }
                    }
                    rang = q;
                }
            }
            q++;
        }

        richTextBox1.Text = Convert.ToString(rang);
    }

    private void Dop_oper_Load_1(object sender, EventArgs e)
    {

    }
} }
```

ПРИЛОЖЕНИЕ 5

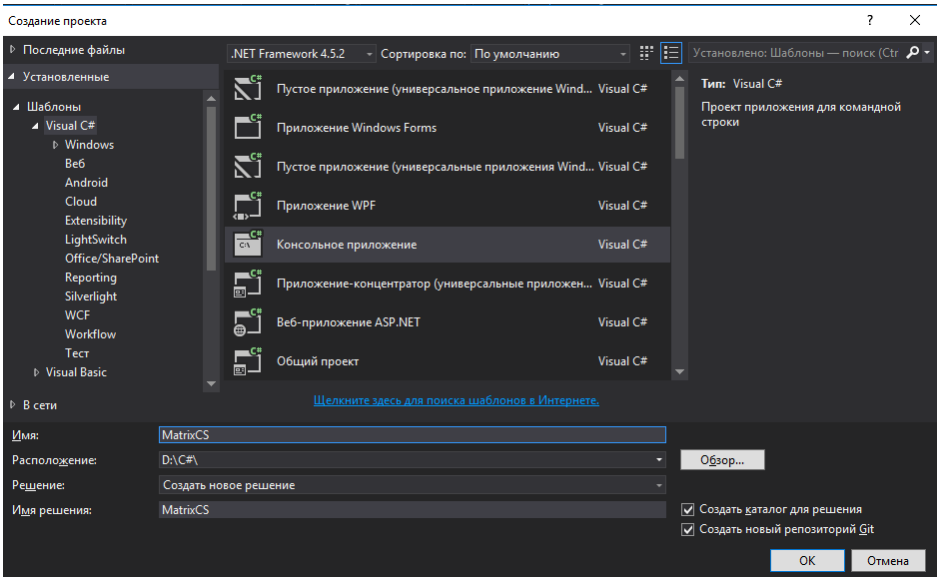

Рис. 1. Создание нового проекта в IDE Visual Studio

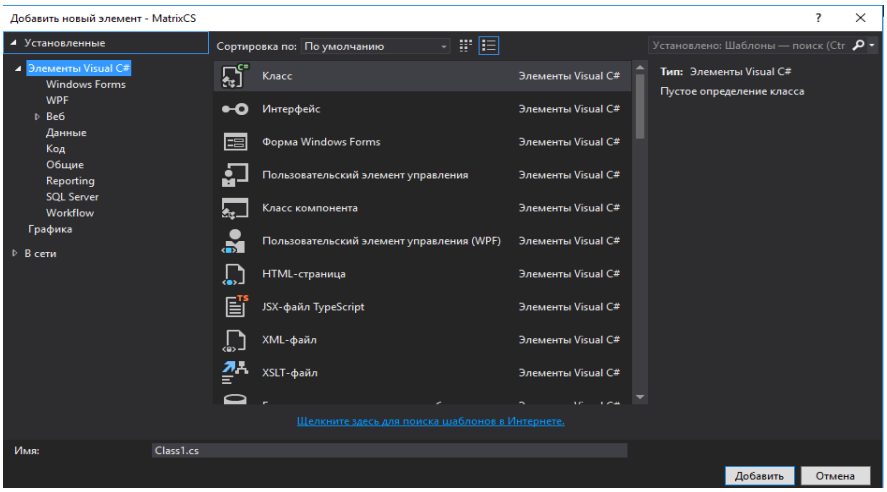

Рис. 2. Добавление класса проекта

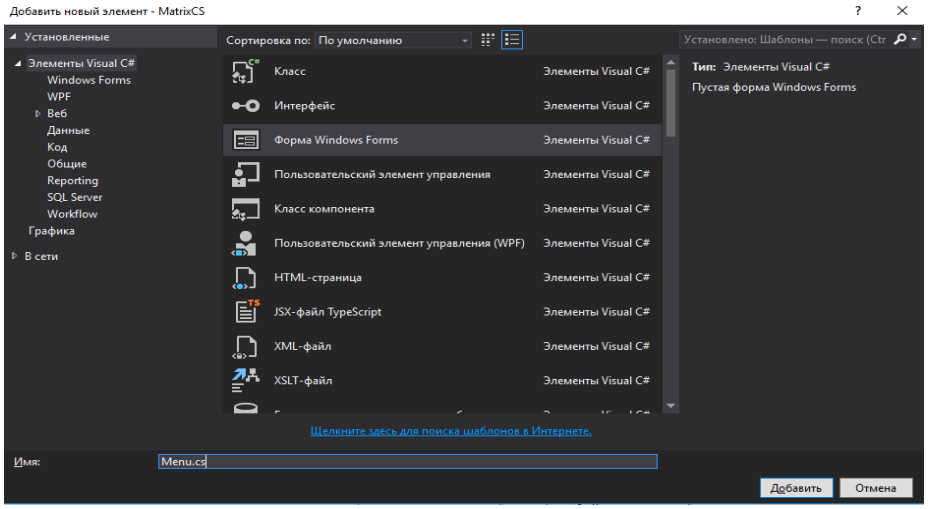

Рис. 3. Добавление формы Windows Forms

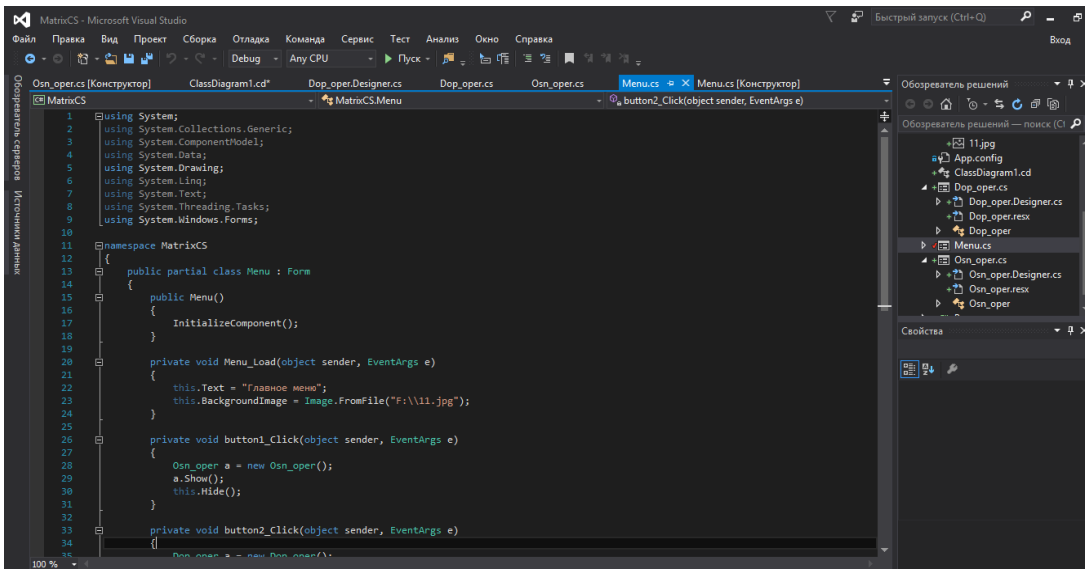

Рис. 4. Написание программного кода в Visual Studio

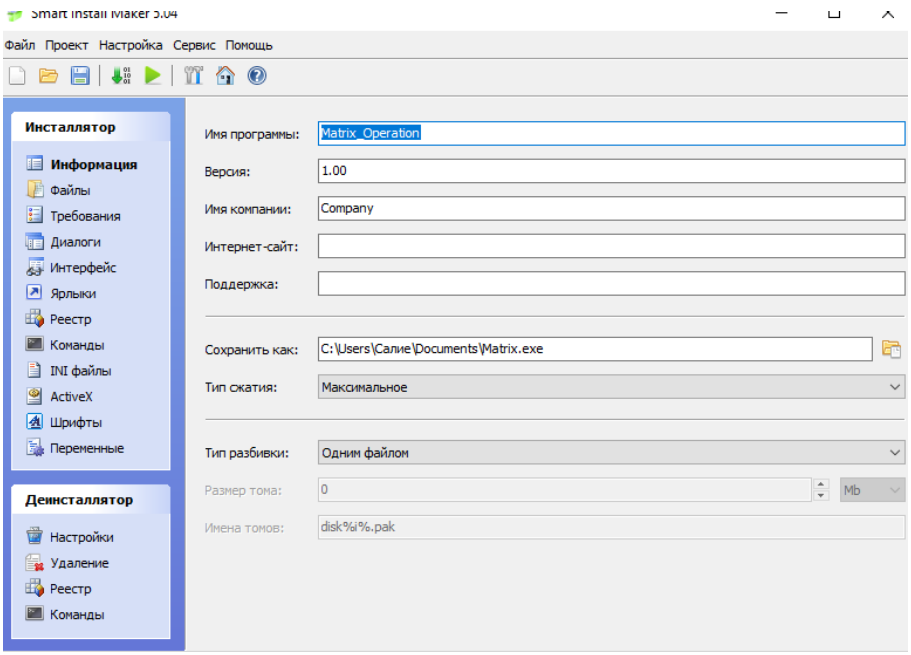

Рис. 5. Внешний вид Smart Install Maker

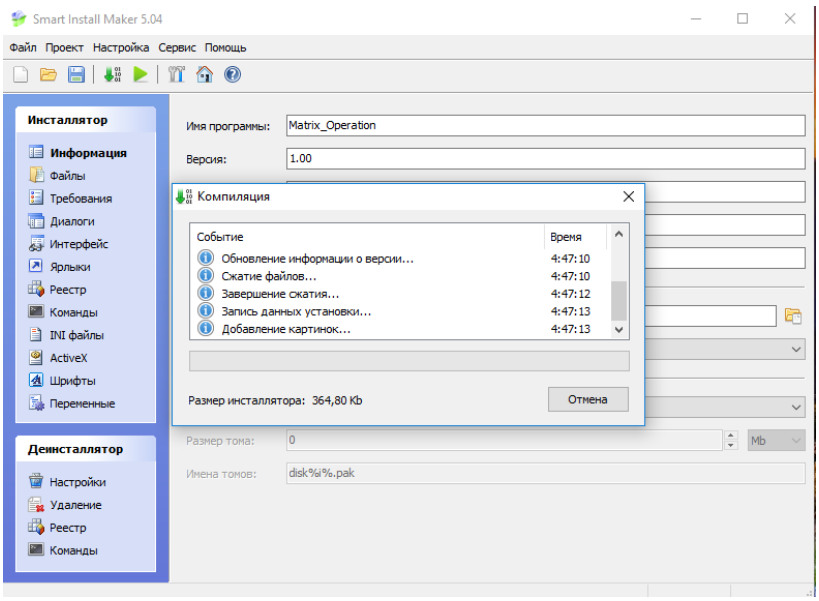

Рис. 6. Компиляция пакета

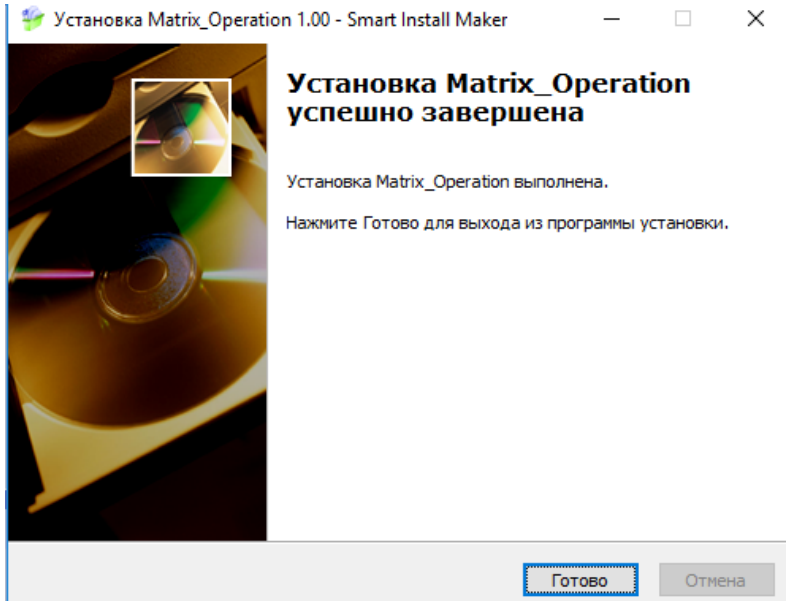

Рис. 7. Мастер установки