Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Факультет компьютерных систем и сетей
Кафедра информатики

Курсовой проект по дисциплине:
«Программирование»

Пояснительная записка к курсовой работе

Тема работы:
«Калькулятор квадратных матриц»

Исполнитель
студент гр. 053506 _____ Осадчий О.Э.
(подпись дата)

Руководитель
_____ Клыбик П.М.
(подпись дата)

_____
(оценка)

Минск
2021 год

Оглавление

Введение

Толчком к написанию данной курсовой работы послужило наличие многих предметов в программе ВУЗа, где требуется производить вычисления квадратных матриц различных размеров. А для того, чтобы удостовериться в правильности своих вычислений, требуется оконное или же мобильное приложение для вычисления данных матриц.

Например, в данном семестре присутствует дисциплины «Математика. Геометрия и алгебра», «Математика. Математический анализ», где постоянно требуются вычисления квадратных матриц. А соответственно, требуется проверка вычислений.

Целью данной курсовой работы является разработка оконного приложения, которое будет выполнять основные операции с квадратными матрицами размера 2-10, что будет значительно помогать студентам в учёбе.

В качестве языка для разработки оконного приложения под систему Windows я выбрал C++, а в качестве платформы IDE C++ Builder.

Приложение работает без подключения к сети интернет, что также является большим плюсом, поскольку не всегда студент имеет доступ к интернету.

1. Язык C++

1.1 Описание языка

C++ — высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день C++ является одним из самых популярных и распространенных языков.

C++ поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

В книге «Дизайн и эволюция C++» Бьёрн Страуструп описывает принципы, которых он придерживался при проектировании C++. Эти принципы объясняют, почему C++ именно такой, какой он есть. Некоторые из них:

- Получить универсальный язык со статическими типами данных, эффективностью и переносимостью языка C.
- Непосредственно и всесторонне поддерживать множество стилей программирования, в том числе процедурное программирование, абстракцию данных, объектно-ориентированное программирование и обобщённое программирование.
- Дать программисту свободу выбора, даже если это даст ему возможность выбирать неправильно.

- Максимально сохранить совместимость с C, тем самым делая возможным лёгкий переход от программирования на C.
- Избежать разночтений между C и C++: любая конструкция, допустимая в обоих языках, должна в каждом из них обозначать одно и то же и приводить к одному и тому же поведению программы.
- Избегать особенностей, которые зависят от платформы или не являются универсальными.
- «Не платить за то, что не используется» — никакое языковое средство не должно приводить к снижению производительности программ, не использующих его.
- Не требовать слишком усложнённой среды программирования.

C++ является компилируемым языком, а это значит, что компилятор транслирует исходный код на C++ в исполняемый файл, который содержит набор машинных инструкций. Но разные платформы имеют свои особенности, поэтому скомпилированные программы нельзя просто перенести с одной платформы на другую и там уже запустить. Однако на уровне исходного кода программы на C++ по большей степени обладают переносимостью, если не используются какие-то специфичные для текущей ос функции. А наличие компиляторов, библиотек и инструментов разработки почти под все распространенные платформы позволяет компилировать один и тот же исходный код на C++ в приложения под эти платформы.

Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. Кроме того, в последнее время набирает ход мобильное направление, где C++ тоже нашел свое применение. И даже в веб-разработке также можно использовать C++ для создания веб-приложений или каких-то вспомогательных сервисов, которые обслуживают веб-приложения. В общем C++ - язык широкого пользования, на котором можно создавать практически любые виды программ.

В отличие от Си язык C++ позволяет писать приложения в объектно-ориентированном стиле, представляя программу как совокупность взаимодействующих между собой классов и объектов. Что упрощает создание крупных приложений.

2.Матрицы

Ма́трица — математический объект, записываемый в виде прямоугольной таблицы элементов кольца или поля (например, целых, действительных или комплексных чисел), который представляет собой совокупность строк и столбцов, на пересечении которых находятся его элементы. Количество строк и столбцов задает размер матрицы. Хотя исторически рассматривались, например, треугольные матрицы, в настоящее время говорят исключительно о матрицах прямоугольной формы, так как они являются наиболее удобными и общими.

Матрицы широко применяются в математике для компактной записи систем линейных алгебраических или дифференциальных уравнений. В этом случае количество строк матрицы соответствует числу уравнений, а количество столбцов — количеству неизвестных. В результате решение систем линейных уравнений сводится к операциям над матрицами.

Квадра́тная ма́трица — это матрица, у которой число строк совпадает с числом столбцов, и это число называется порядком матрицы. Любые две квадратные матрицы одинакового порядка можно складывать и умножать.

Квадратные матрицы часто используются для представления простых линейных отображений — таких, как деформация или поворот. Например, если **R** — квадратная матрица, представляющая вращение (матрица поворота) и **v** — вектор-столбец, определяющий положение точки в пространстве, произведение **Rv** даёт другой вектор, который определяет положение точки после вращения. Если **v** — вектор-строка, такое же преобразование можно получить, используя $\mathbf{vR}^{\mathrm{T}}$, где $\mathbf{R}^{\mathrm{T}}$ — транспонированная к **R** матрица. Также с помощью квадратных матриц мы имеем возможность находить якобиан.

$$\begin{bmatrix} 9 & 13 & 5 & 2 \\ 1 & 11 & 7 & 6 \\ 3 & 7 & 4 & 1 \\ 6 & 0 & 7 & 10 \end{bmatrix}$$

Рисунок 1. Квадратная матрица 4*4

3.Логика программы

3.1. ООП

При создании данной программы я придерживался ООП. Объектно-ориентированное программирование - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Класс матрицы хранится в отдельном файле, имеет все нужные поля и методы, что является реализацией концепций ООП.

3.2. Функционал класса матрицы

1. Класс `Calc` имеет два поля:

```
int rows;
float **arr;
```

Поле **rows** является размером матрицы, а **arr** является двумерным массивом, который хранит в себе значения матрицы.

Оба поля заполняются в конструкторе класса:

```
Calc( int rows, TStringGrid *Grid )
      {
            this->rows = rows;
            arr = new float*[rows];
               for (int i = 0; i < rows; i++)
               {
                  arr[i] = new float[rows];
               }

               for (int i = 0; i < rows; i++)
               {
                       for (int j = 0; j < rows; j++)
                       {
                         arr[i][j] = StrToFloat(Grid->Cells[i][j]);
                       }
               }
      }
```

2. Одной из самых часто используемых функций является нахождение определителя матрицы:

```
int determinant(float** arr, const int n)
{
      if (n == 1)
            return arr[0][0];
      else if (n == 2)
            return arr[0][0] * arr[1][1] - arr[0][1] * arr[1][0];
      else {
             int determ = 0;
             for (int k = 0; k < n; k++)
```

```
            {
               float** m = new float*[n-1];

               for (int i = 0; i < n - 1; i++)
                  {
                     m[i] = new float[n - 1];
                  }

               for (int i = 1; i < n; i++)
                  {
                     int t = 0;
                     for (int j = 0; j < n; j++)
                        {
                           if (j == k)
                           continue;
                           m[i-1][t] = arr[i][j];
                           t++;
                        }
                  }
               determ += pow(-1, k + 2) * arr[0][k] * determinant(m, n - 1);
               deleting(m, n - 1);
            }
         return determ;
      }
}
```

Для данной функции написана вспомогательная функция, которая будет очищать память:

```
void deleting (float** arr, const int k)
      {
            for (int i = 0; i < k; i++)
               {
                 delete[] arr[i];
               }
            delete [] arr;
      }
```

3. Функция транспонирования матрицы:

```
void transp(float** arr, int n, TStringGrid *Grid)
   {
         for (int i = 0; i < n; i++)
          {
               for (int j = 0; j < n; j++)
                  {
                     Grid->Cells[i][j]= arr[j][i];
                  }
          }
   }
```

4. Функция нахождения обратной матрицы, при условии, что определитель матрицы не равен нулю:

```
void obrmatrix(TStringGrid *Grid)
   {
         int determ = this->determinant(this->arr,this->rows);
         if (determ == 0)
           {
              ShowMessage("Обратной матрицы не существует.");
           }
           else
           {
              for (int i = 0; i < rows; i++)
                 {
                   for (int j = 0; j < rows; j++)
                      {
                        Grid->Cells[i][j] =
((arr[(j+1)%rows][(i+1)%rows] * arr[(j+2)%rows][(i+2)%rows]) -
(arr[(j+1)%rows][(i+2)%rows] * arr[(j+2)%rows][(i+1)%rows]))/ determ;
                      }
                 }
           }
     }
```

5. Функция сложения двух матриц:

```
void plus (float** arr,float** arr2 , int rows,TStringGrid *Grid)
   {
      for (int i = 0; i < rows; i++)
         {
             for (int j = 0; j < rows; j++)
                {
                   arr[i][j] += arr2[i][j];
                }
         }

       for (int i = 0; i < rows; i++)
         {
             for (int j = 0; j < rows; j++)
                {
                   Grid->Cells[i][j] = arr[i][j];
                }
         }
   }
```

6. Функция вычитания двух матриц:

```
void minus (float** arr, float** arr2, int rows, TStringGrid *Grid)
   {
     for (int i = 0; i < rows; i++)
        {
          for (int j = 0; j < rows; j++)
             {
               arr[i][j] -= arr2[i][j];
             }
```

```
        }
      for (int i = 0; i < rows; i++)
         {
            for (int j = 0; j < rows; j++)
               {
                 Grid->Cells[i][j]= arr[i][j];
               }
         }
   }
```

7. Функция вычисления произведения матриц:

```
void myltyply(float** arr, float** arr2 , int rows,TStringGrid *Grid)
   {
         float **arr3 = new float*[rows];
         for (int i = 0; i < rows; i++)
            {
               arr3[i] = new float[rows];
            }

         for (int i = 0; i < rows; i++)
            {
              for (int j = 0; j < rows; j++)
                 {
                    arr3[i][j] = 0;
                    for (int z = 0; z < rows; z++)
                        {
                          arr3[i][j]  =  arr3[i][j]  +  arr[i][z]  *
arr2[z][j];
                         }
                 }

         for (int i = 0; i < rows; i++)
            {
               for (int j = 0; j < rows; j++)
                  {
                    Grid->Cells[i][j]= arr3[i][j];
                  }
            }

         for (int i = 0; i < rows; i++)
            {
              delete[] arr3[i];
            }
              delete[] arr3;
         }
```

8.Последней функций класса Calc является его деструктор, который очищает память выделенную под динамически созданный массив **arr**:

```
~Calc()
{
   for  (int i = 0; i < rows; i++)
        {
        delete[] arr[i];
         }
        delete[] arr;
 }
```

4. Интерфейс

Интерфейс моей программы представлен одним рабочим окном:

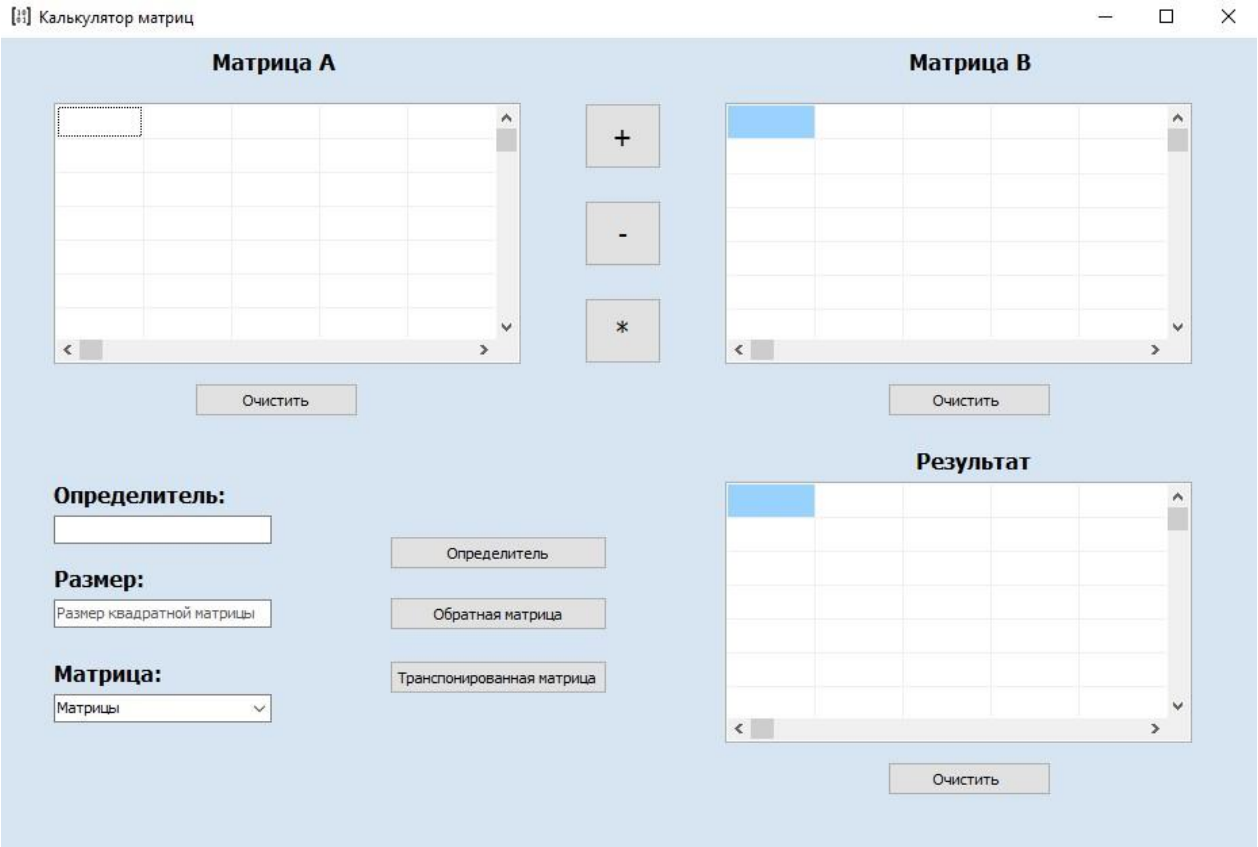

Рисунок 2. Окно калькулятора матриц

Оно условно разделено на 5 блоков:

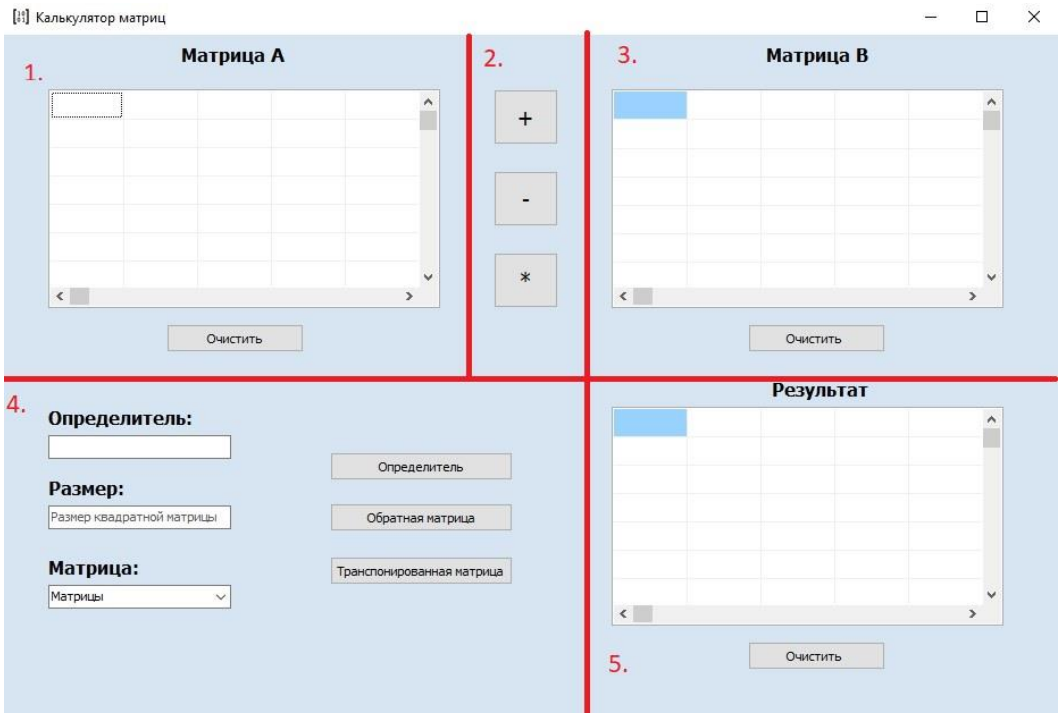

Рисунок 3. Разделение программы на блоки

- Первый блок:

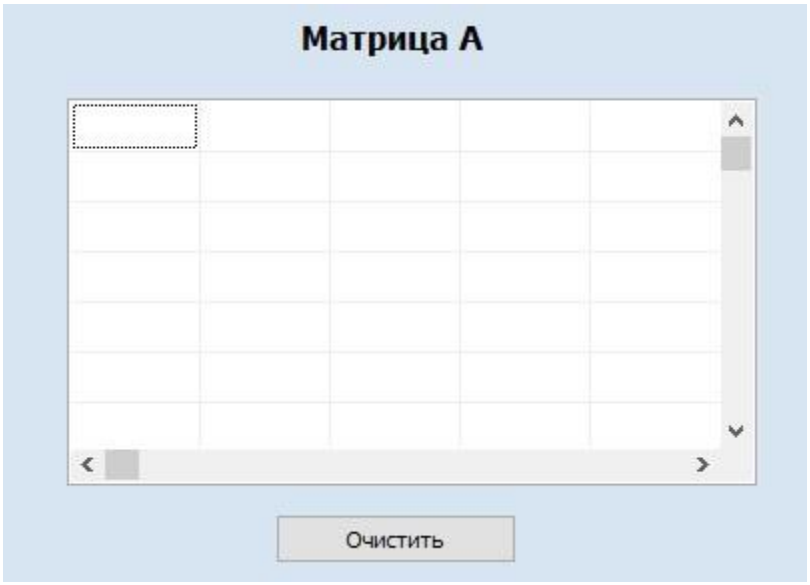

Рисунок 4. Блок 1

В первом блоке находится элемент TStringGrid, который отвечает за внесение в программу элементов первой матрицы – «матрицы А».
Так же в данном блоке находится элемент TButton. Данная кнопка отвечает за полное очищение матрицы А.

- Второй блок:

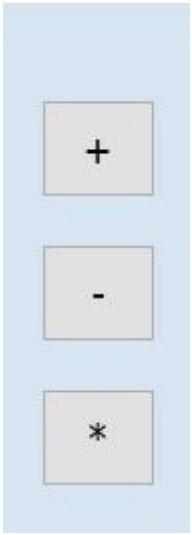

Рисунок 5. Блок 2

Во втором блоке находятся только элементы TButton, каждая из данных кнопок отвечает за определённую операцию с матрицами.
Каждая кнопка работает в паре с элементом TComboBox, который находится в блоке номер 4. После выбора матрицы в ComboBox, мы выбираем кнопку.
Первая кнопка отвечает за операцию сложения.

Вторая кнопка отвечает за операцию вычитания и работает аналогично операции сложения.

Третья кнопка отвечает за операцию умножения. За основу берётся матрица, выбранная в элементе ComboBox в блоке 4, после чего происходят операции умножения матриц. В данном случае, в отличии от операций сложения и вычитания, имеет большое значение то, какую матрицу мы выбрали первой, поскольку умножение матриц в общем случае некоммутативно:

A*B != B*A

Также для каждой из кнопок написана подсказка, которая будет появляться при наведении курсора мыши на саму кнопку и удержании там курсора 1-2 секунды.

- Третий блок:

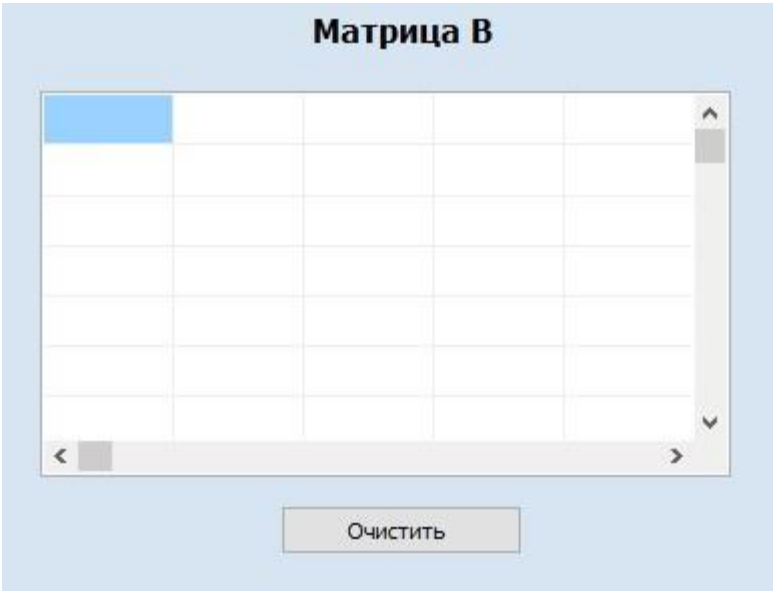

Рисунок 6. Блок 3

Третий блок аналогичен первому и так же предназначен для заполнения матрицы – «матрицы B». Так же, как и в первом блоке, присутствует элемент TButton, для очистки матрицы B.

- Четвёртый блок:

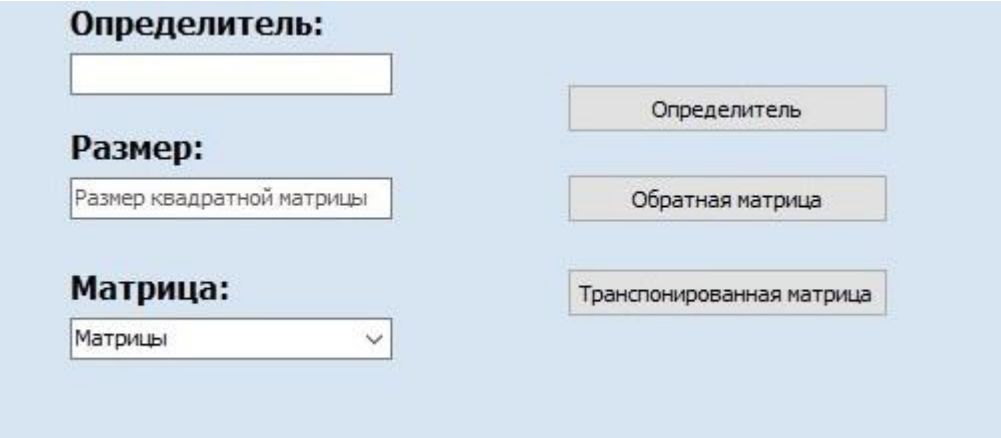

Рисунок 7. Блок 4

В четвёртом блоке находятся элементы управления матрицами:

1. TComboBox. Данный элемент отвечает за выбор «ведущей» матрицы. Как было описано мной выше, данный элемент очень важен при выполнении операции умножения матриц. Если при работе с матрицами вы не выберете ничего, то по умолчанию будет выбрана матрица B.
2. TEdit. Здесь находятся элементы типа Tedit и TLabeledEdit, второй служит для ввода в программу размера матрицы, с которым мы будем работать. Это является обязательным условием работы с программой. При игнорировании введения размера матрицы будет всплывать следующее окно:

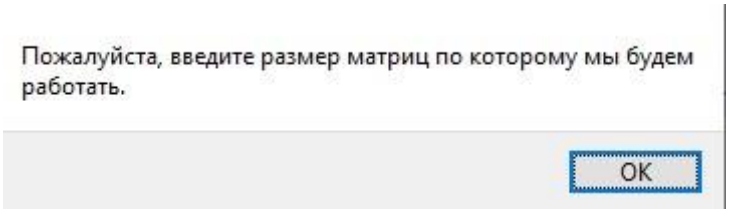

Рисунок 8. Сообщение об ошибке

Также в случае, если пользователь ввёл размер матрицы n, а заполнил её на размер n-1, то при выполнении операций с матрицей, программа автоматически заполнит оставшиеся пустые поля нулями:

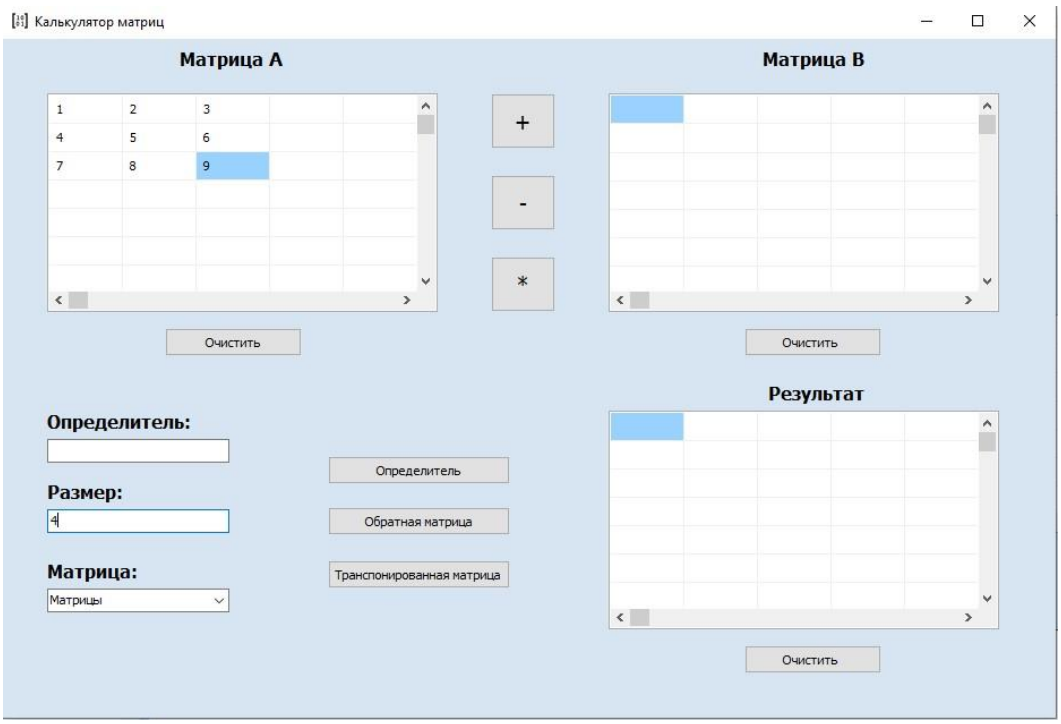

Рисунок 9 а. Неверный размер

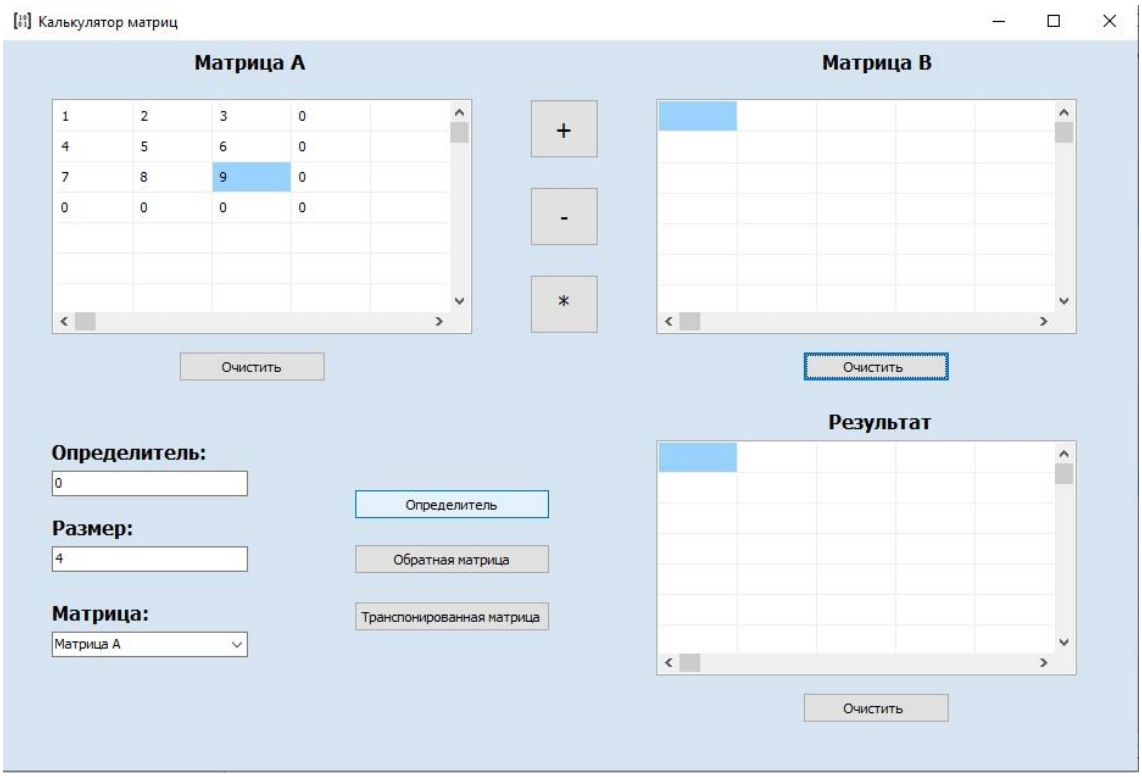

Рисунок 9 б. Неверный размер

Первый же элемент TEdit служит для вывода определителя матрицы. Элемент нельзя изменять или редактировать.

3. TButton. В этом блоке находятся 3 элемента типа TButton – 3 кнопки:

а) «Определитель» - кнопка отвечает за вычисления определителя матрицы размера, введённого в TLabeledEdit.

b) «Обратная матрица» - кнопка отвечает за построение и вывод обратной матрицы в матрицу «результат» в пятом блоке, если это возможно, т.е. определитель матрицы не равен 0.
c) «Транспонированная матрица» - кнопка так же отвечает за построение и вывод транспонированной матрицы в матрицу «результат» в пятом блоке.

- Пятый блок:

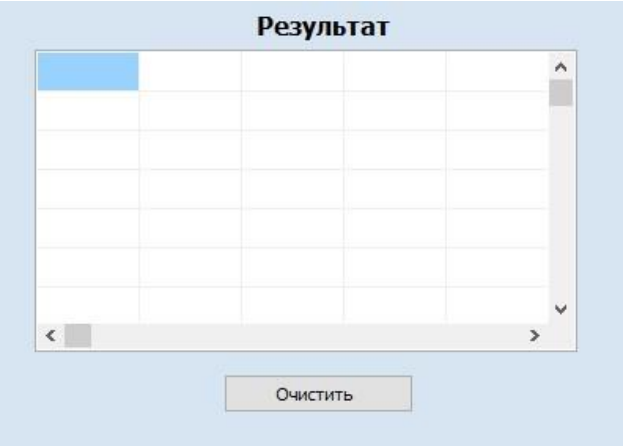

Рисунок 10. Блок 5

Элементы пятого блока являются идентичными элементам первого и второго блоков, однако же используются они не для внесения элементов матрицы в программу, а наоборот – для вывода результата операций с матрицами. Элемент является неизменяемым.
Аналогично первому и второму блокам, присутствует кнопка очистки матрицы результата.

5. Заключение

В данной курсовой работе было разработано десктопное приложение под систему Windows, которое способно выполнять операции над квадратными матрицами.

В ходе курсового проектирования были рассмотрены основные характеристики используемого языка программирования C++, а также основные возможности, предоставляемые интегрированной средой разработки C++ Builder.

Программа была протестирована мной и моими однокурсниками на таких программных предметах как «математика. Математический анализ», «математика. Геометрия и алгебра». Ошибок выявлено не было.

В перспективе планируется расширение рамок использования данного приложение: осуществление переноса на мобильные устройства , а так же распространение данного приложения через площадки «AppStore» и «PlayMarket» .

6. Список использованных источников

1. https://www.embarcadero.com/ru/products/cbuilder
2. https://metanit.com/cpp/tutorial/1.1.php
3. Корн Г., Корн *Т.* Алгебра матриц и матричное исчисление // Справочник по математике. — 4-е издание. — М: Наука, 1978. — С. 392—394.
4. Жевняк Р.М., Карпук А.А. Высшая математика. Основы аналитической геометрии и линейной алгребры. Часть 1. С 52, 72, 81.
5. https://cubook.pro/object/tablitsa-strok-stringgrid
6. Герберт Шилдт, C++ руководство для начинающих. Модуль 4.
7. Роберт Лафоре , объектно-ориентированное программирование в C++ , 4-е издание.
8. https://ru.onlinemschool.com/math/assistance/matrix/
9. https://math.semestr.ru/kramer/opred.php
10. https://matrixcalc.org/