Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему:

«Программное средство «Польский калькулятор» с визуализацией заданной функции»

БГУИР КП 1-40 01 01  85100097  ПЗ

Студент: гр. 851005 Марченко Д. А.

Руководитель: асс. Болтак С.В.

Минск 2019

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc10760544)

[1 АНАЛИЗ ПРОТОТИПОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПС 4](#_Toc10760545)

[1.1 Анализ прототипов 4](#_Toc10760546)

[1.2 Выводы 5](#_Toc10760547)

[1.3 Требования к разрабатываемому ПС. 6](#_Toc10760548)

[2 РАЗРАБОТКА АЛГОРИТМОВ ПРОГРАММНОГО СРЕДСТВА 6](#_Toc10760549)

[2.1 Основные алгоритмы программного средства 6](#_Toc10760550)

[2.2 Схемы алгоритмов решения задачи по ГОСТ 19.701-9 7](#_Toc10760551)

[3 РАЗРАБОТКА СТРУКТУР ДАННЫХ 13](#_Toc10760552)

[3.1 Разработка алгоритма перевода инфиксной записи в обратную польскую 13](#_Toc10760553)

[4 ОБОСНОВАНИЕ ВЫБОРА СРЕДЫ РАЗРАБОТКИ И ТЕХНИЧЕСКИХ ПРИЕМОВ ПРОГРАММИРОВАНИЯ 14](#_Toc10760554)

[4.1 Обоснование выбора среды разработки Borland Delphi 7.0 14](#_Toc10760555)

[4.2 Обоснование выбора линейного однонаправленного списка 15](#_Toc10760556)

[4.3 Обоснование технических приемов программирования 16](#_Toc10760557)

[5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 16](#_Toc10760558)

[5.1 Тестирование функционала кнопок на главной форме. 16](#_Toc10760559)

[5.2 Тестирование функционала кнопок построить и очистить. 17](#_Toc10760560)

[5.2 Тестирование функционала кнопок на форме: Загрузить и Сохранить. 19](#_Toc10760561)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЮ 32](#_Toc10760562)

[6.1 Начало работы 32](#_Toc10760563)

[6.2 Завершение работы программы 34](#_Toc10760564)

[ЗАКЛЮЧЕНИЕ 34](#_Toc10760565)

[СПИСОК ЛИТЕРАТУРЫ 35](#_Toc10760566)

[ПРИЛОЖЕНИЕ 1 36](#_Toc10760567)

# ВВЕДЕНИЕ

В наше время калькуляторы играют значимую роль во всех сферах жизни: начиная профессиональной, заканчивая бытовой. Эти вычислительные приборы заменили человечеству абаки и счеты, пользующиеся популярностью в свое время. История счетной техники – это процесс приобретения опыта и знаний человечеством, в результате чего счетные механизмы смогли гармонично вписаться в жизнь человека.

Первый электронный калькулятор появился в конце двадцатого века. В дальнейшем он совершенствовался, претерпевал изменения внутренний механизм машины, менялась форма калькулятора. Совсем скоро, после выхода в свет первого калькулятора, усовершенствованные его собратья стали удобными, компактными и маленькими.

Первый электронный калькулятор появился в конце двадцатого века. В дальнейшем он совершенствовался, претерпевал изменения внутренний механизм машины, менялась форма калькулятора. Совсем скоро, после выхода в свет первого калькулятора, усовершенствованные его собратья стали удобными, компактными и маленькими.

Быт современных людей уже сложно представить без карманного калькулятора. Помимо прибора для вычисления, в наше время есть и калькулятор онлайн в интернет, позволяющий решать задачи, не отходя от компьютера.

Данная курсовая работа посвящена разработке программному средству «Польский калькулятор» с визуализацией заданной функции. Программа разрабатывается на языке Delphi.

Пояснительная записка состоит из шести разделов. Первый содержит в себе анализ аналогов и формирование требований к разрабатываемой программе. Второй раздел включает разработку основных алгоритмов программного средства. В третьем разделе описывается разработка структур данных, использованных в программе. В четвертом разделе проводиться обоснование выбора среды разработки и некоторых технических приемов программирования. Пятый раздел построен полностью в виде таблицы, описывающей результаты тестирования программного средства. Шестой раздел представляет собой руководство пользователя, содержащее последовательность действий, необходимую для успешного выполнения работы программы.

# 1 АНАЛИЗ ПРОТОТИПОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПС

## Анализ прототипов

Калькулятор Windows – одна из наиболее распространенных программ-калькуляторов. Разрабатывается и распространяется компанией Windows.

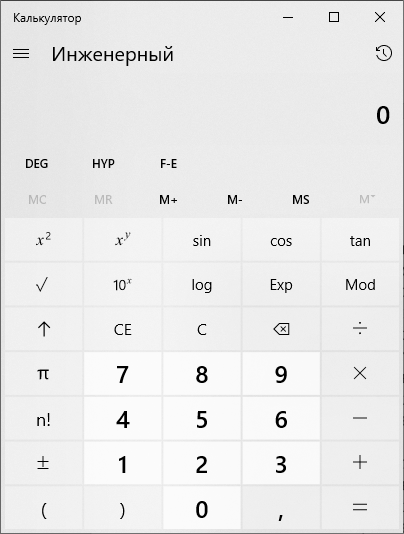


Рисунок 1.1 − Калькулятор на Windows 10

Преимущества:

- Стабильная работа программы

- Бесплатный

- В комплекте с ОС;

Недостатки:

- отсутствие визуализации графика функции;

- нет возможности сохранения и загрузки из текстового файла.

Следующий рассматриваемый ресурс – Google calculator.

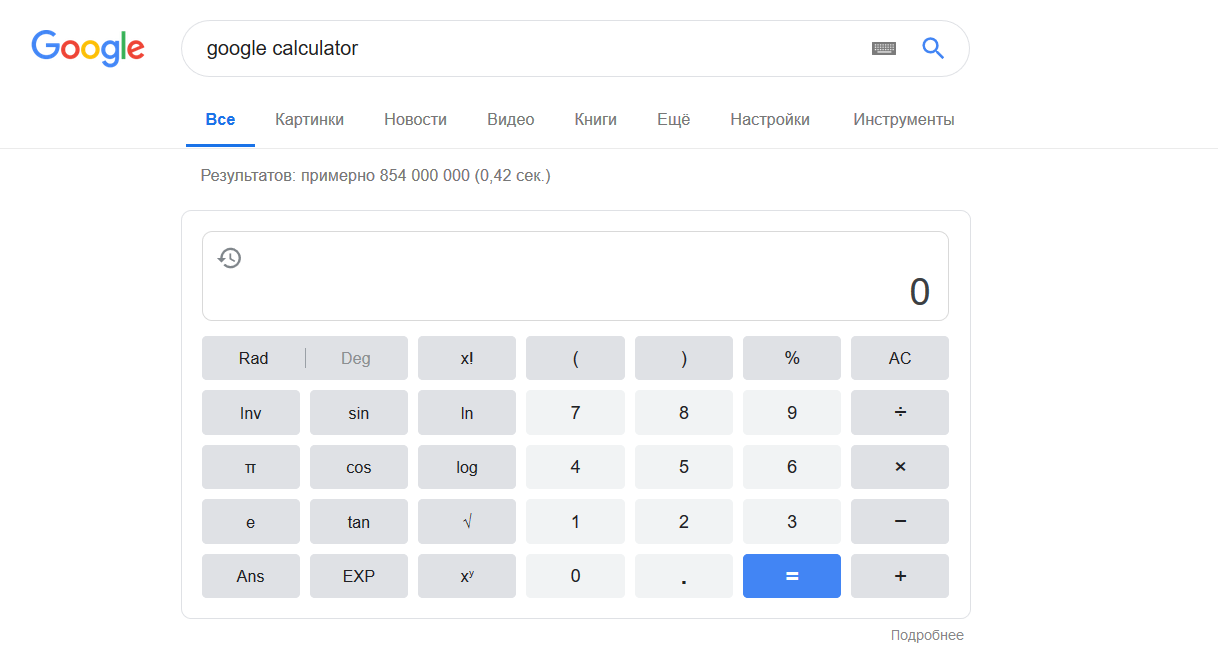


Рисунок 1.2 – Калькулятор Google

Преимущества:

- удобный пользовательский интерфейс;

- Бесплатный

Недостатки:

- отсутствие визуализации графика функции;

- нет возможности сохранения и загрузки из текстового файла;

- требуется интернет соединение

- долгое время загрузки сайта и работы калькулятора при плохом интернет соединении

## 1.2 Выводы

Сравнительный анализ прототипов показывает, что наиболее важными характеристиками: удобство, интуитивность, продуктивность.

Целью данного курсового проекта является разработка desktop-приложения.

Преимуществами данного курсового проекта являются:

- интуитивно понятный пользовательский интерфейс;

- высокая скорость работы калькулятора;

- построение графиков введенной функции;

- наличие всех важнейших инструментов на одной панели программы.

## 1.3 Требования к разрабатываемому ПС.

Основной задачей калькулятора является математические вычисления над числами. Можно производить умножение и деление, сложение и вычитание, возведение в степень, нахождения факториала, вычисления синуса и косинуса. А также программа должна быть способна визуализировать заданную функцию.

**1.3.1 Описание предметной области**

Калькулятор – электронное вычислительное устройство, предоставляющее пользователю выполнять сложные операции над числами. Пользователям, которым не так важны вычисления, как алгоритм решения определенной задачи, лучше будет не тратить время на подсчёты, а обратиться к помощнику, калькулятор дает эту возможность

Главной задачей предметной области стоит быстрое и точное вычисление выражений и их визуализация, повышая кпд пользователя. Из этого можно сделать вывод, что главными участниками данной предметной области являются потенциальные пользователи.

С данной информационной системой может работать любые группы пользователей.

# 2 РАЗРАБОТКА АЛГОРИТМОВ ПРОГРАММНОГО СРЕДСТВА

## 2.1 Основные алгоритмы программного средства

В первую очередь на вход программы поступает выражение, состоящее из односимвольных идентификаторов и знаков арифметических действий, необходимо создать алгоритм, способный распознавать заданную пользователем функцию и подсчитывать ее или же сообщить об ошибке.

Когда вы записываете арифметическое выражение вроде B \* C, то его форма предоставляет вам достаточно информации для корректной интерпретации. В данном случае мы знаем, что переменная B умножается на переменную C, поскольку оператор умножения \* находится в выражении между ними. Такой тип записи называется инфиксной, поскольку оператор расположен между двух операндов, с которыми он работает.

Рассмотрим другой инфиксный пример: B + C \* D. Операторы + и \* по-прежнему располагаются между операндами, но тут уже есть проблема. D какими именно операндами они будут работать? + работает с B и C или \* принимает C и D? Выражение выглядит неоднозначно.

Фактически, вы можете читать и писать выражения такого типа долгое время, и они не будут вызывать у вас вопросов. Причина в том, что вы кое-что знаете о + и \*. Каждый оператор имеет свой приоритет. Операторы с высоким приоритетом используются прежде операторов с низким. Единственной вещью, которая может изменить порядок приоритетов, являются скобки. Для арифметических операций умножение и деление стоят выше сложения и вычитания. Если появляются два оператора одинакового приоритета, то используются порядок слева направо, или их ассоциативность.

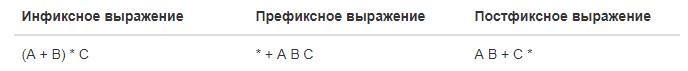
Но для компьютера инфиксная запись не является оптимальной, зачастую они используют либо префиксную запись, либо постфикную запись. В данных записях не существует скобочек и других элементов, способных поменять порядок выполнения операций рисунок 3.1.

Рисунок 2.1 – инфиксное, префиксное и постфиксное выражение

Единственная разница между префиксным и постфисным методам состоит в направлении, в котором следует читать запись (слева направо или справа налево), поэтому достаточно подробно рассмотреть только один из них. В обратной польской записи оператор записывается после его операндов: A + B – инфексная форма, A B + - постфиксная или обратная польская запись. Преимуществом обратной польской записи в том, что приоритет операторов может быть представлен порядком их появления и наоборот для префиксной записи.

Для оптимальной работы программы необходимо реализовать следующие алгоритмы:

а) чтение выражения и проверка его на корректность;

б) преобразование X в строке, если имеется;

в) преобразование выражения в форму обратной польской записи;

г) подсчет строки;

д) вывод результата.

Рассмотрим каждый из алгоритмов детально.

## 2.2 Схемы алгоритмов решения задачи по ГОСТ 19.701-9

**2.2.1 Схема алгоритма чтение выражения и проверка его на корректность**

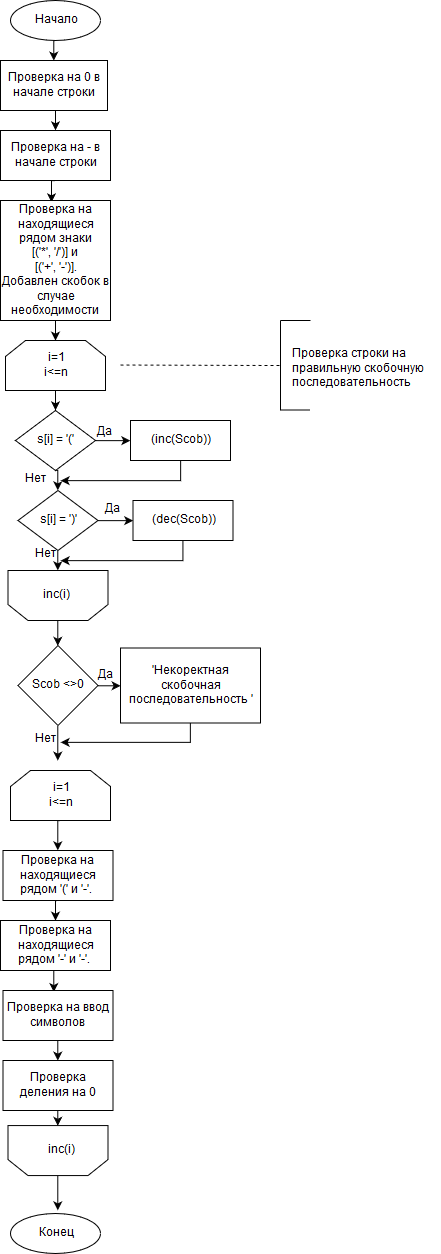


Рисунок 2.2 – Алгоритм чтения выражения и проверки его на корректность

**2.2.2 Схема алгоритма преобразование X в строке**

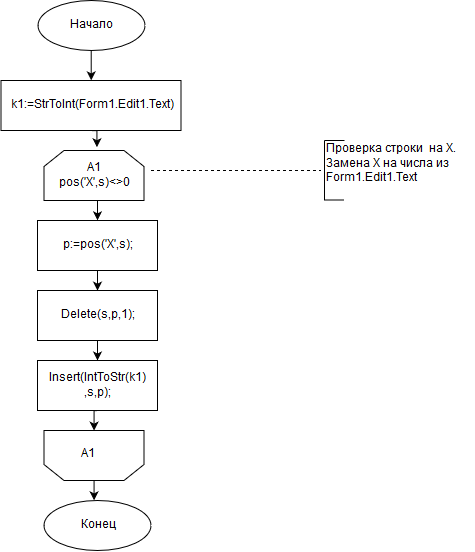


Рисунок 2.3 – Алгоритм преобразования X в строке

**2.2.3 Схема алгоритма преобразование выражения в форму обратной польской записи**

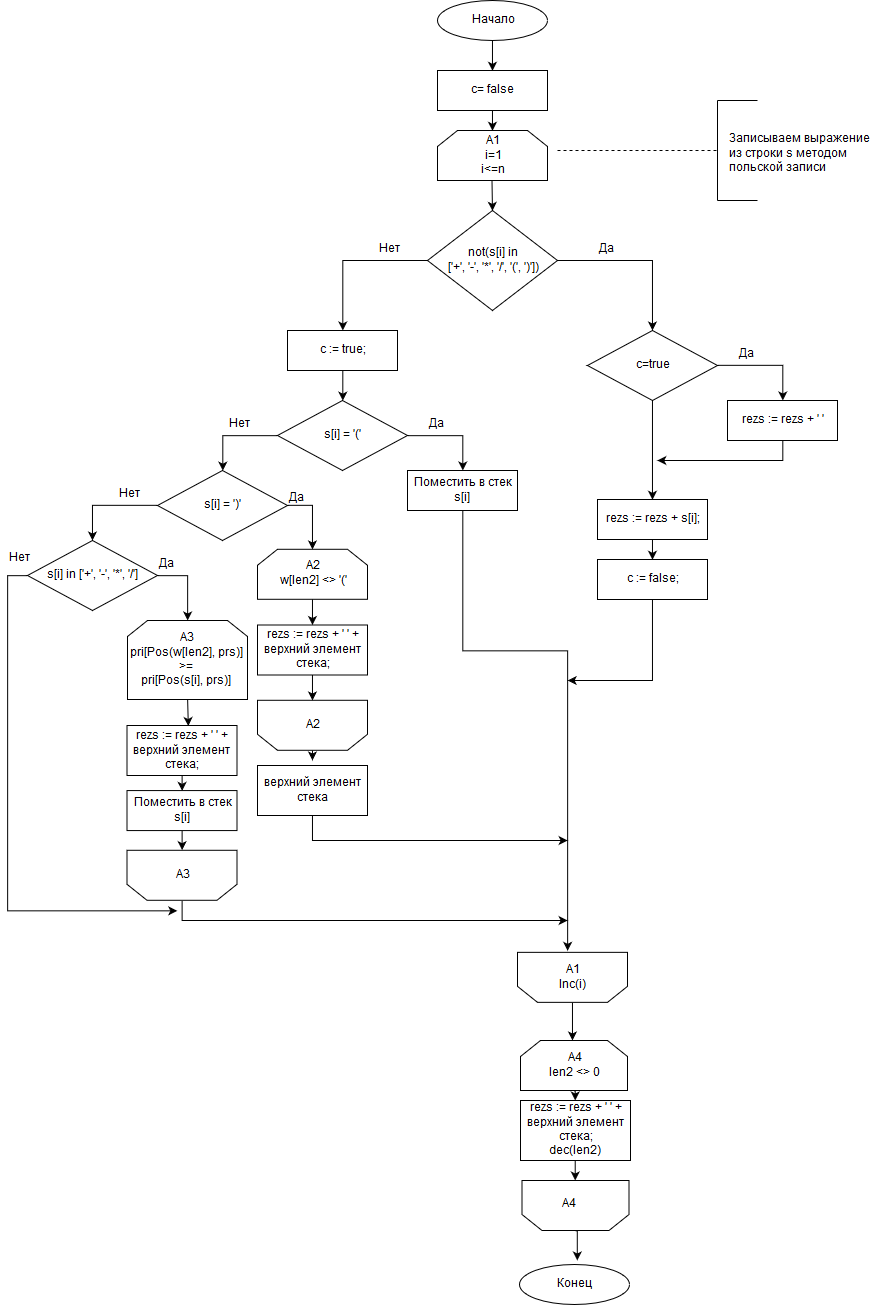


Рисунок 2.4 – Алгоритм преобразование выражения в форму обратной польской записи

**2.2.4 Схема алгоритма подсчета строки**

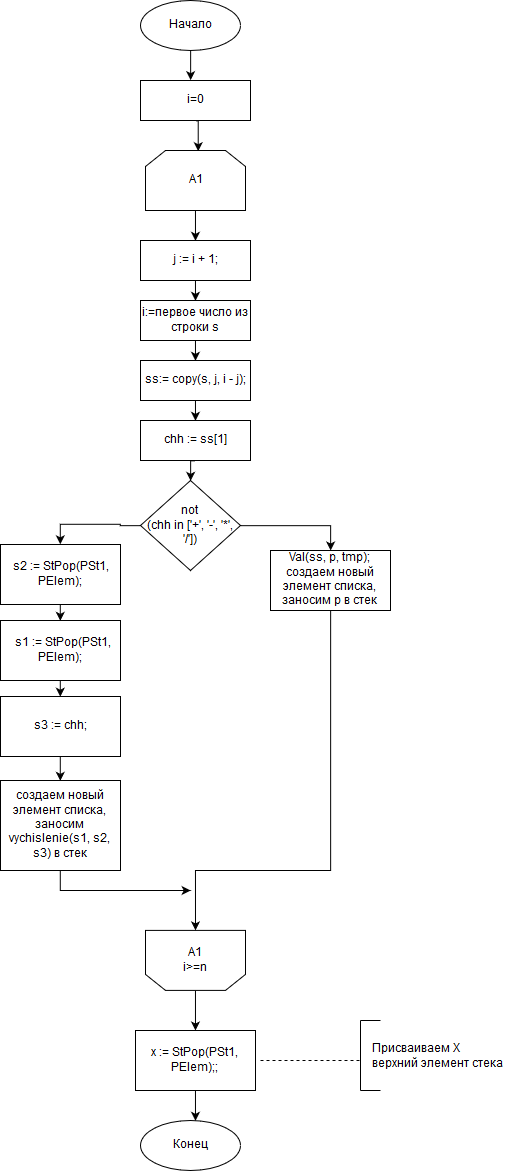


Рисунок 2.5 – Алгоритм подсчета строки

**2.2.5 Схема алгоритма вывода результата**

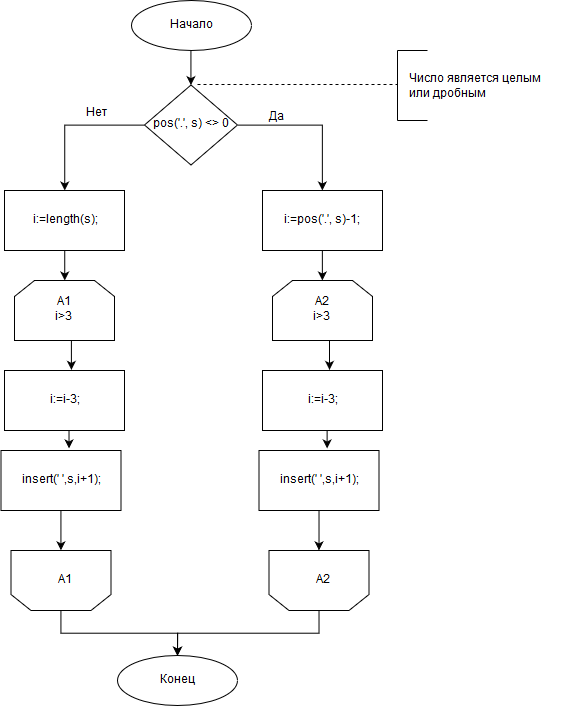


Рисунок 2.6– Алгоритм вывода результата

# 3 РАЗРАБОТКА СТРУКТУР ДАННЫХ

## 3.1 Разработка алгоритма перевода инфиксной записи в обратную польскую

Чтобы подсчитать значение функции заданной пользователем, нужно перевести ее из инфксной формы в обратную польскую запись. Для этого посмотрим ближе на сам процесс конвертирования.

Рассмотрим выражение A + B \* C. Его постфиксным эквивалентом является A B C \* +. Операнды A, B и C остаются на своих местах, а местоположение меняют только операторы. Ещё раз взглянем на знаки в инфиксном выражении. Первым при проходе слева направо нам попадётся +. Однако, в постфиксном выражении + находится в конце, так как следующий оператор \*, имеет приоритет над сложением. Порядок операторов в первоначальном выражении обратен результирующему постфиксному выражению.

В процессе обработки выражения операторы должны где-то храниться, пока не найден их соответствующий правый операнд. Также порядок этих сохраняемых операторов может быть обратным (из-за их приоритета), как в данном примере со сложением и умножением. Поскольку оператор сложения, появляющийся перед оператором умножения, имеет более низкий приоритет, то он должен появиться после использования последнего. Из-за такого обратного порядка имеет смысл рассмотреть использование стека для хранения операторов до тех пор, пока они не понадобятся.

Что насчёт (A + B) \* C? Его постфиксный эквивалент: A B + C \*. Обрабатывая это инфиксное выражение слева направо, первым мы встретим +. В этом случае, когда мы увидим \*, + уже будет помещён в результирующее выражение, поскольку имеет преимущество над \* в силу использования скобок. Теперь можно приступить к рассмотрению работы алгоритма преобразования. Когда мы видим левую скобку, то сохраняем её как знак, что должен будет появиться другой оператор с высоким приоритетом. Он будет ожидать, пока не появится соответствующая правая скобка, чтобы отметить его местоположение. После появления правой скобки оператор выталкивается из стека.

Поскольку мы сканируем инфиксное выражение слева направо, то для хранения операторов будем использовать стек. Это предоставит нам обратный порядок. На вершине стека всегда будет последний сохранённый оператор. Когда бы мы не прочитали новый оператор, мы должны сравнить его по приоритету с операторами в стеке. Если текущий оператор будет приоритетом выше следующего, то выталкиваем его из стека и записываем следующий оператор в стек на место вытолкнутого. Схема алгоритма изображена на рисунке 2.4

# 4 ОБОСНОВАНИЕ ВЫБОРА СРЕДЫ РАЗРАБОТКИ И ТЕХНИЧЕСКИХ ПРИЕМОВ ПРОГРАММИРОВАНИЯ

## 4.1 Обоснование выбора среды разработки Borland Delphi 7.0

Среда визуального программирования Delphi 7.0 является мощным средством для быстрой и качественной разработки приложений. Имеющаяся библиотека визуальных компонентов позволяет быстро создать пользовательский интерфейс. Объектно-ориентированный язык Object Pascal, положенный в основу Delphi, является расширением языков Turbo Pascal и Borland Pascal фирмы Borland и нашел в себе отражение новых веяний в программировании. Компонентный принцип, используемый в Delphi, позволяет создавать полноценные Windows-приложения, написав минимальное количество строк кода. Delphi представляет собой открытую систему, позволяя добавлять свои компоненты в систему, модифицировать уже имеющиеся стандартные компоненты благодаря тому, что предоставлены их исходные тексты. Благодаря всему этому разработка программ в среде Delphi становится простой и быстрой.

Основные преимущества среды Delphi 7.0:

- высокая скорость разработки приложения (RAD);

- высокая производительность разработанного приложения:

- низкие требования разработанного приложения к ресурсам компьютера;

- наращиваемость за счет встраивания новых компонентов в среду Delphi;

- возможность разработки новых компонентов и инструментов собственными средствами Delphi (существующие компоненты и инструменты доступны в исходных кодах);

- удачная проработка иерархии объектов.

Система программирования Delphi рассчитана на программирование различных приложений и предоставляет большое количество компонентов для этого. К тому же крайне важны скорость и качество создания программ, а эти характеристики может обеспечить только среда визуального проектирования, способная взять на себя значительные объемы рутинной работы по подготовке приложений, а также согласовать деятельность группы постановщиков, кодировщиков, тестеров и технических писателей. Возможности Delphi полностью отвечают подобным требованиям и подходят для создания систем любой сложности.

## 4.2 Обоснование выбора линейного однонаправленного списка

Преимуществом использования динамической памяти служит то, что одна и та же память мо­жет быть использована для хранения различной информации в процессе исполнения программы. Поскольку память выделяется для определенной цели и освобождается, когда ее использование завершилось, то можно использовать ту же самую память в другой момент времени для других целей в другой части программы. Другим преимуществом динамического выделения памяти явля­ется возможность создания с ее помощью связанных списков, двоичных деревьев и других дина­мических структур данных.

Таким образом динамическое выделение памяти способствует экономии места, занимаемого программой на жестком диске, оптимизирует работу программы с памятью и снижает затраты ресурсов системы.

В качестве структуры для хранения списка видеокарт был выбран линейный однонаправленный список. Информационным элементом списка, может являться значение любого типа – числа, массивы записи и т.п.

В рамках задачи курсового проектирования нет необходимости в просмотре списка в обе стороны. В связи с этим выбран именно однонаправленный список.

При подсчете выражения при каждом новом числе создается дополнительный элемент однонаправленного динамического списка и туда заносится число. При дальнейшем подсчете выражения число достается из списка, а динамическая память, занимаемая им, освобождается и может быть использована другими процессами.

## 4.3 Обоснование технических приемов программирования

Главная форма программного средства «Польский калькулятор» проектировалась с целью создать удобную рабочую область для пользователя с интуитивно-понятным интерфейсом. Это было достигнуто при помощи следующих приемов программирования:

1. Все кнопки для работы с программой были помещены на главный экран;
2. Процедуры для работы с программой разделены по разделам на главном экране;
3. Разработана возможность как вводить выражение с клавиатуры, так и пользоваться специальными кнопками на главном экране приложения, что позволяет эффективнее и быстрее вводить выражение.

# 5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## 5.1 Тестирование функционала кнопок на главной форме.

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Полученный результат |
| 1.Ввести функцию 2+2\*2 в поле ввода  2.Кликнуть правой кнопкой мышки на кнопку ‘=’ | 1. В поле ввода выведется результат:  ‘6’ | Тест пройден (рис. 5.21). |
| 1.Ввести функцию 2^3+5! в поле ввода  2.Кликнуть правой кнопкой мышки на кнопку ‘=’ | 1. В поле ввода выведется результат:  ‘128’ | Тест пройден (рис. 5.22). |
| 1.Ввести функцию 1000! в поле ввода  2.Кликнуть правой кнопкой мышки на кнопку ‘=’ | 1. В поле ввода выведется результат:  ‘4,023872600E2657’ | Тест пройден (рис. 5.3). |
| 1.Ввести функцию 100^100 в поле ввода  2.Кликнуть правой кнопкой мышки на кнопку ‘=’ | 1. В поле ввода выведется результат:  ‘1E200’ | Тест пройден (рис. 5.4). |
| 1.Ввести функцию (2+2)\*2 в поле ввода  2.Кликнуть правой кнопкой мышки на кнопку ‘=’ | 1. В поле ввода выведется результат:  ‘8’ | Тест пройден (рис. 5.5). |
| 1.Ввести функцию 3+4-5\*6/7+8^9 в поле ввода  2.Кликнуть правой кнопкой мышки на кнопку ‘=’ | 1. В поле ввода выведется результат:  ‘134217730,714285714285714’ | Тест пройден (рис. 5.6). |
| 1.Ввести функцию Sin(30) в поле ввода  2.Кликнуть правой кнопкой мышки на кнопку ‘=’ | 1. В поле ввода выведется результат:  ‘-0,98803162’ | Тест пройден (рис. 5.7). |
| 1.Ввести функцию Cos(30) в поле ввода  2.Кликнуть правой кнопкой мышки на кнопку ‘=’ | 1. В поле ввода выведется результат:  ‘0,1542514498’ | Тест пройден (рис. 5.8). |

## 5.2 Тестирование функционала кнопок построить и очистить.

|  |  |  |
| --- | --- | --- |
| 1. Ввести функцию Cos(Sin(X)) в поле ввода  2.Выставить значения коэффициента растяжения, X от/до, шаг X  3.Кликнуть левой кнопкой мышки на кнопку построить | 1. В графическом поле отрисуется график введенной функции | Тест пройден (рис. 5.9). |
| 1. Ввести функцию 1/(X) в поле ввода  2.Выставить значения коэффициента растяжения, X от/до, шаг X  3.Кликнуть левой кнопкой мышки на кнопку построить | 1. В графическом поле отрисуется график введенной функции | Тест пройден (рис. 5.10). |
| 1. Ввести функцию -2\*(X)^2+2\*(X)+200 в поле ввода  2.Выставить значения коэффициента растяжения, X от/до, шаг X  3.Кликнуть левой кнопкой мышки на кнопку построить | 1. В графическом поле отрисуется график введенной функции | Тест пройден (рис. 5.11). |
| 1. Ввести функцию Sin(X) в поле ввода  2.Выставить значения коэффициента растяжения, X от/до, шаг X  3.Кликнуть левой кнопкой мышки на кнопку построить | 1. В графическом поле отрисуется график введенной функции | Тест пройден (рис. 5.12). |
| 1. Ввести функцию Cos(X) в поле ввода  2.Выставить значения коэффициента растяжения, X от/до, шаг X  3.Кликнуть левой кнопкой мышки на кнопку построить | 1. В графическом поле отрисуется график введенной функции | Тест пройден (рис. 5.13). |
| 1. Ввести функцию (X)^2 в поле ввода  2.Выставить значения коэффициента растяжения, X от/до, шаг X  3.Кликнуть левой кнопкой мышки на кнопку построить | 1. В графическом поле отрисуется график введенной функции | Тест пройден (рис. 5.14). |
| 1. Ввести функцию (X)^3 в поле ввода  2.Выставить значения коэффициента растяжения, X от/до, шаг X  3.Кликнуть левой кнопкой мышки на кнопку построить | 1. В графическом поле отрисуется график введенной функции | Тест пройден (рис. 5.15). |

## 5.2 Тестирование функционала кнопок на форме: Загрузить и Сохранить.

|  |  |  |
| --- | --- | --- |
| Тест | Ожидаемый результат | Полученный результат |
| Тест-кейс для кнопки “Загрузить”:  1.Записать в текстовом файле выражение  2.Нажать левой кнопкой мыши на кнопку загрузить  3.выбрать нужный текстовый файл | 1. В поле ввода выведется выражение из загруженного текстового файла | Тест пройден (рис. 5.16), (рис. 5.16.1). |
| Тест-кейс для кнопки “Сохранить”:  1.Нажать левой кнопкой мыши на кнопку сохранить  3.выбрать нужный текстовый файл | 1.В текстовый файл запишется выражение из поля ввода | Тест пройден (рис. 5.17). |

**5.6 Тестирование на крайних значениях и взлом программы**

|  |  |  |
| --- | --- | --- |
| Последовательность действий | Ожидаемый результат | Полученный результат |
| Попытка деления на 0:  1.Ввод функции  2.Кликнуть правой кнопкой мышки на кнопку ‘=’ | Сообщение об ошибке:  ‘Деление на 0’ | Тест пройден. (рис. 5.18). |
| Попытка ввести некорректные символы:  1.Ввод функции  2.Кликнуть правой кнопкой мышки на кнопку ‘=’ | Сообщение об ошибке.  ‘Ошибка ввода(Error1)’ | Тест пройден. (рис. 5.19). |
| Попытка ввести выражение с неправильной скобочной последовательностью:  1.Ввод функции  2.Кликнуть правой кнопкой мышки на кнопку ‘=’ | Сообщение об ошибке.  ‘Неправильная скобочная последовательность’ | Тест пройден. (рис. 5.20). |
| Попытка ввести выражение без чисел:  1.Ввод функции  2.Кликнуть правой кнопкой мышки на кнопку ‘= | Сообщение об ошибке.  ‘Выражение без чисел | Тест пройден. (рис. 5.21). |

Программа успешно прошла все тесты, что показывает корректность работы программы и соответствие функциональным требованиям.

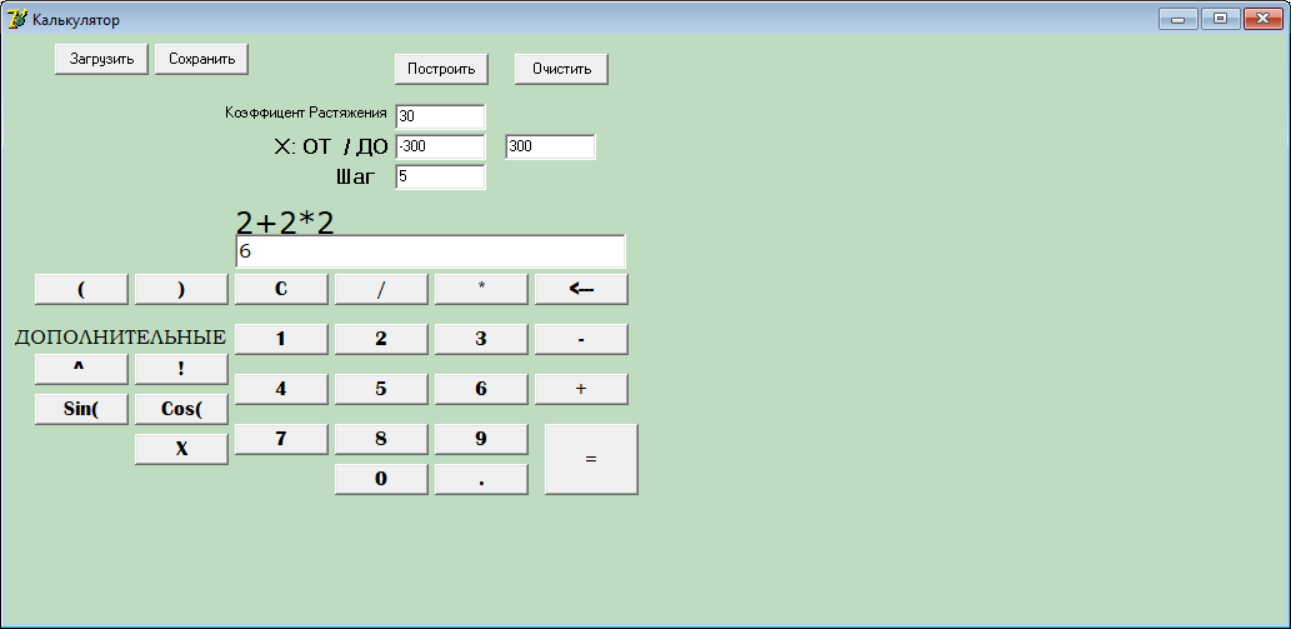


Рисунок 5.1

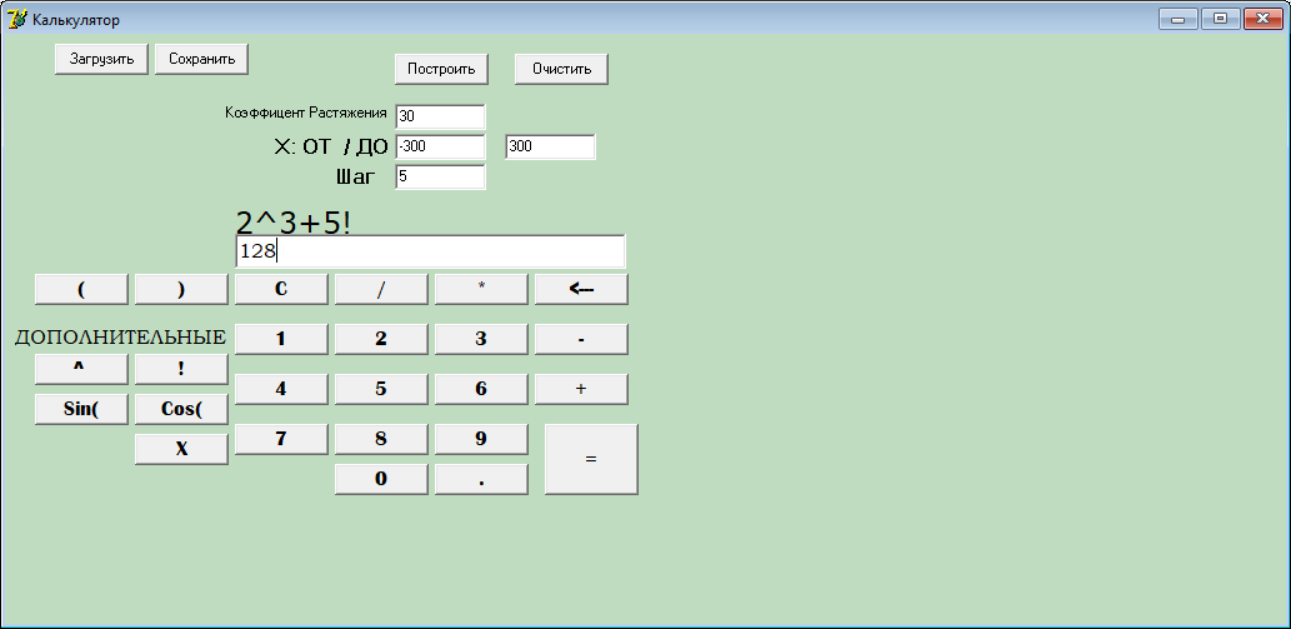


Рисунок 5.2

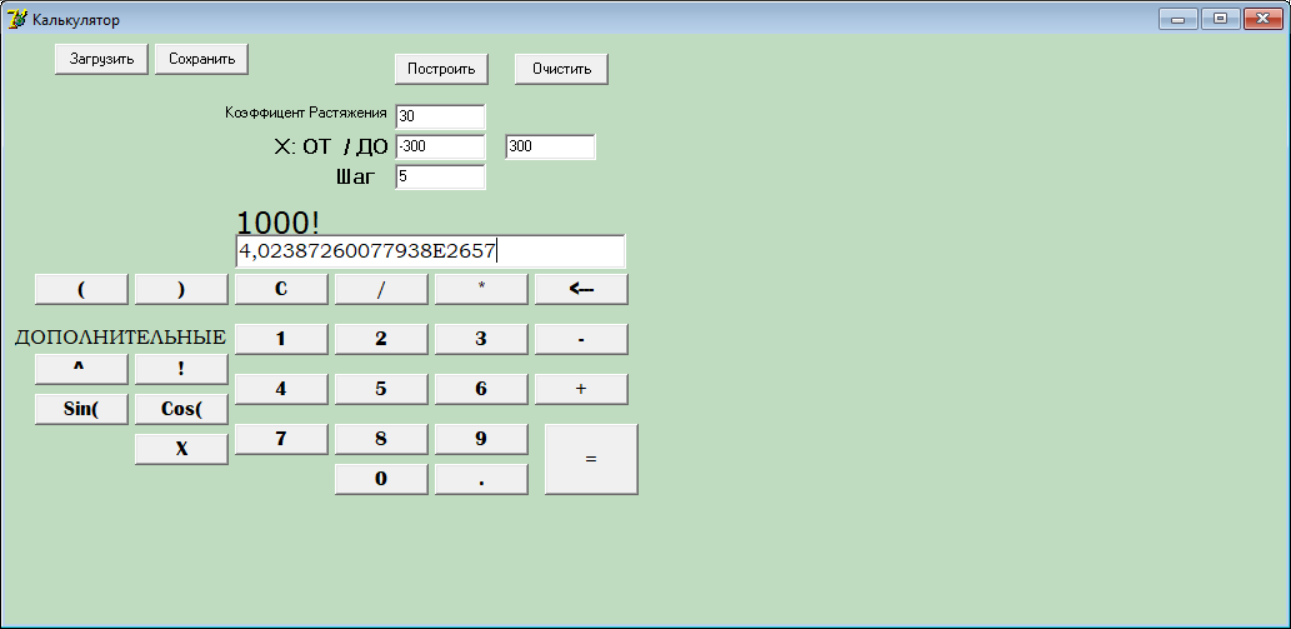


Рисунок 5.3

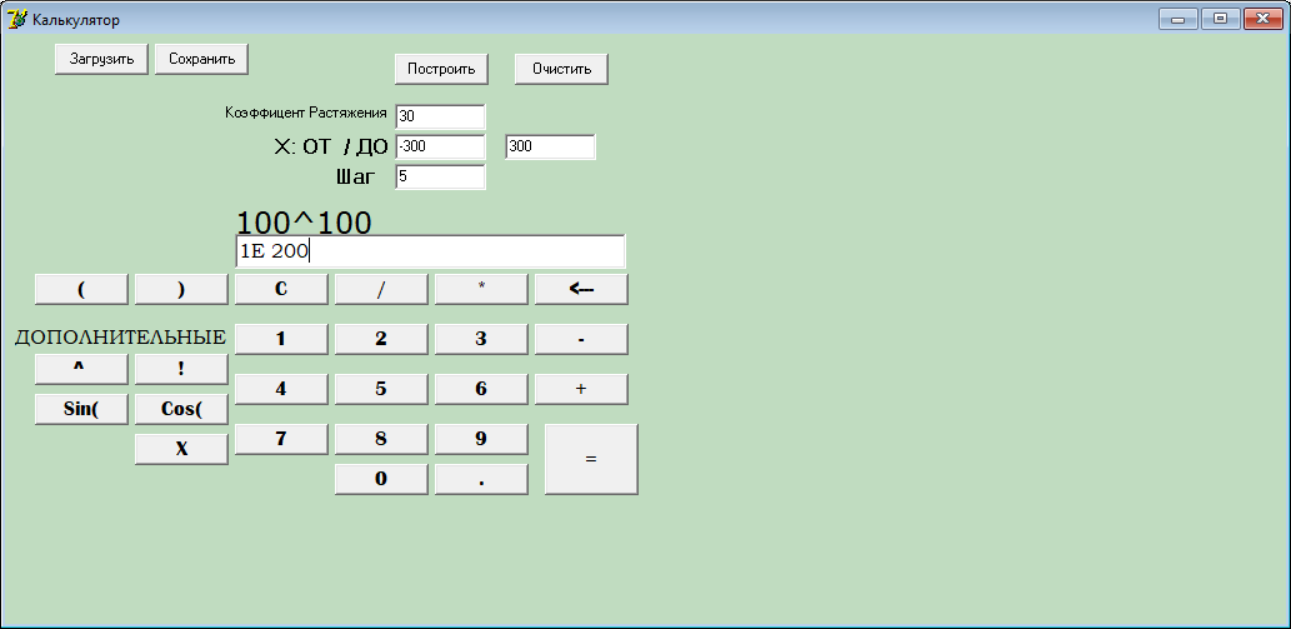


Рисунок 5.4

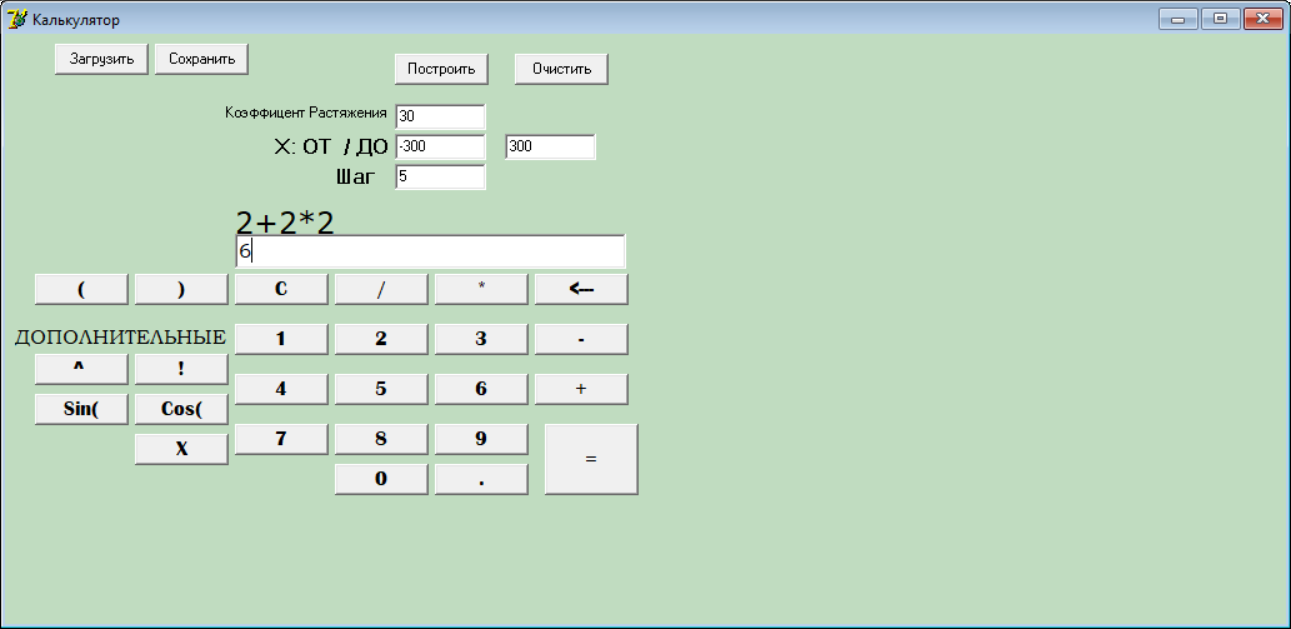


Рисунок 5.5

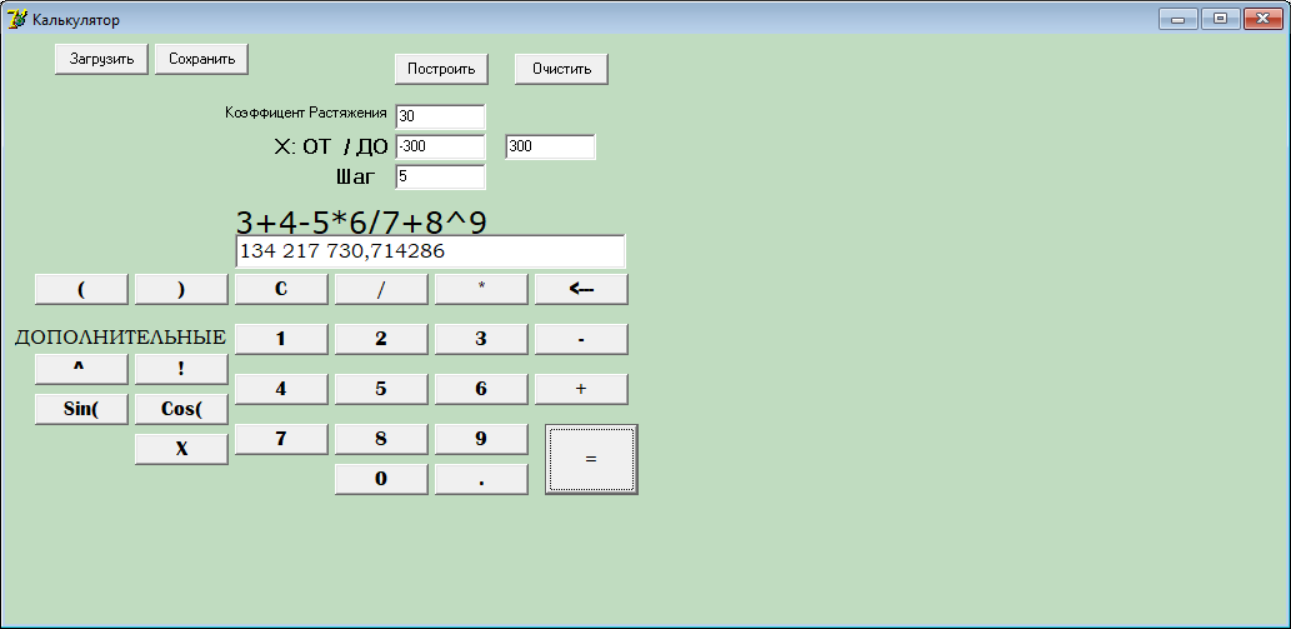


Рисунок 5.6

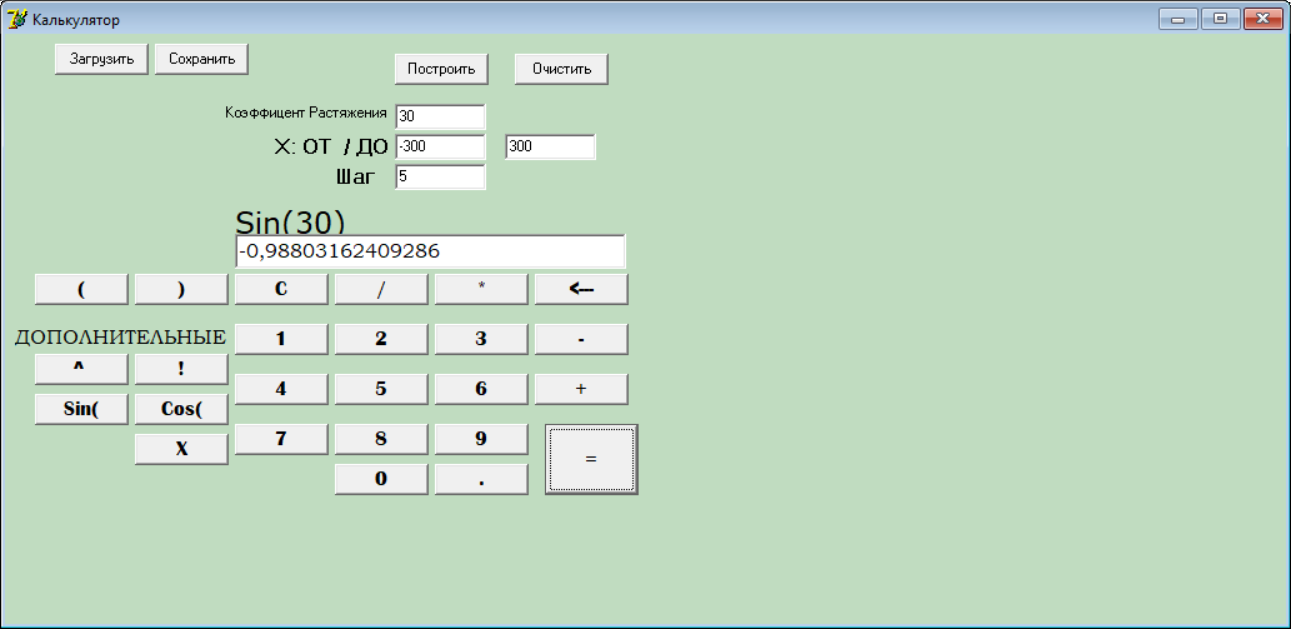


Рисунок 5.7

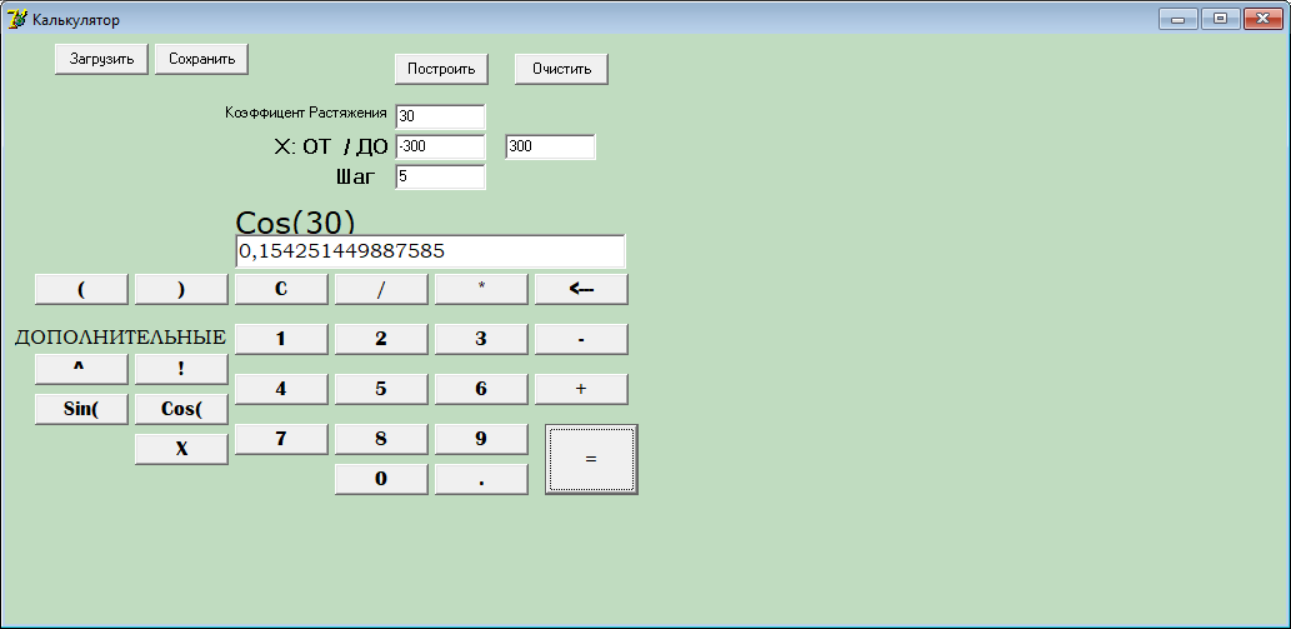


Рисунок 5.8

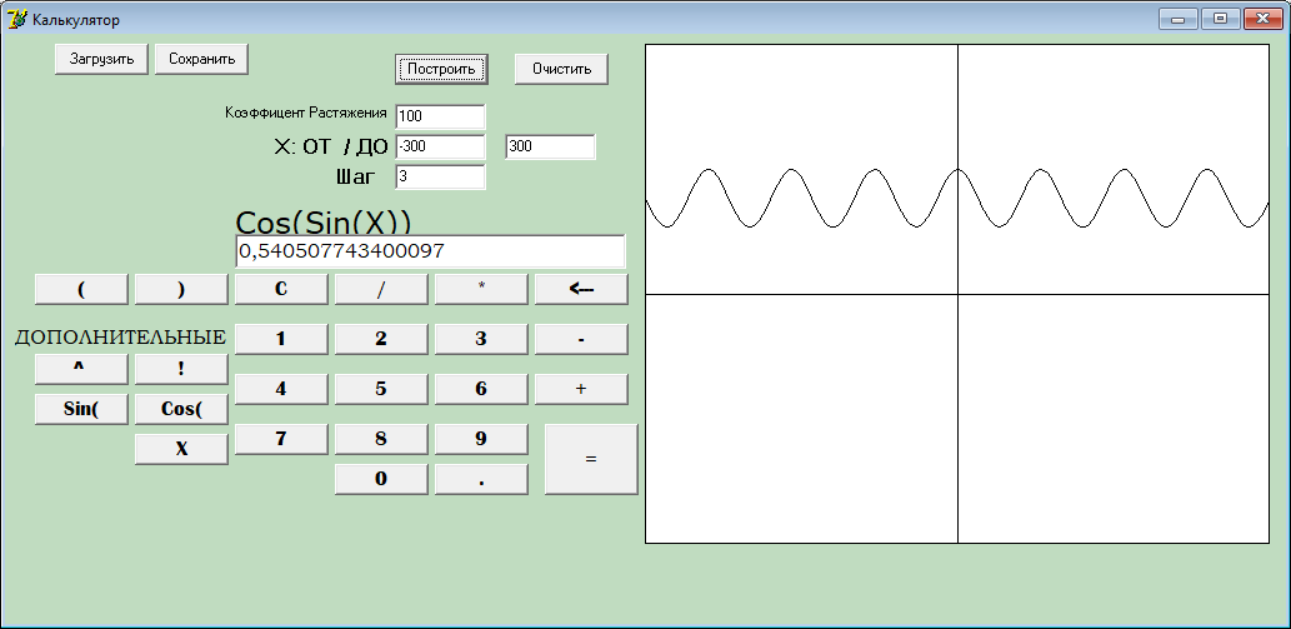


Рисунок 5.9

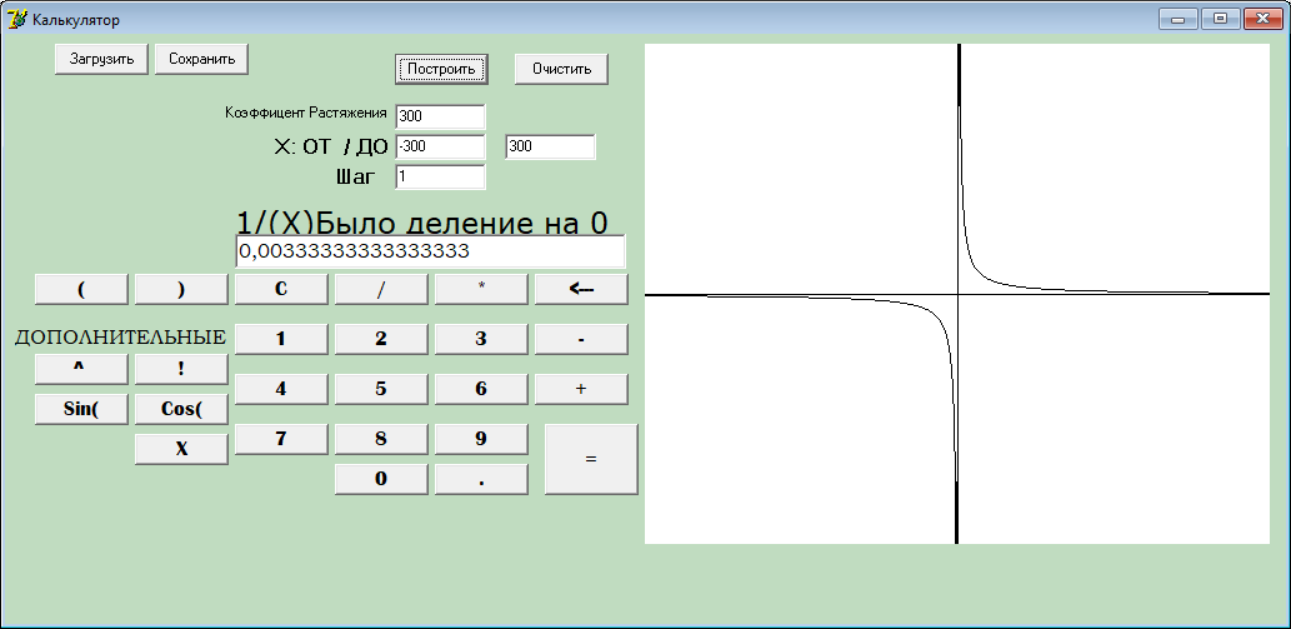


Рисунок 5.10

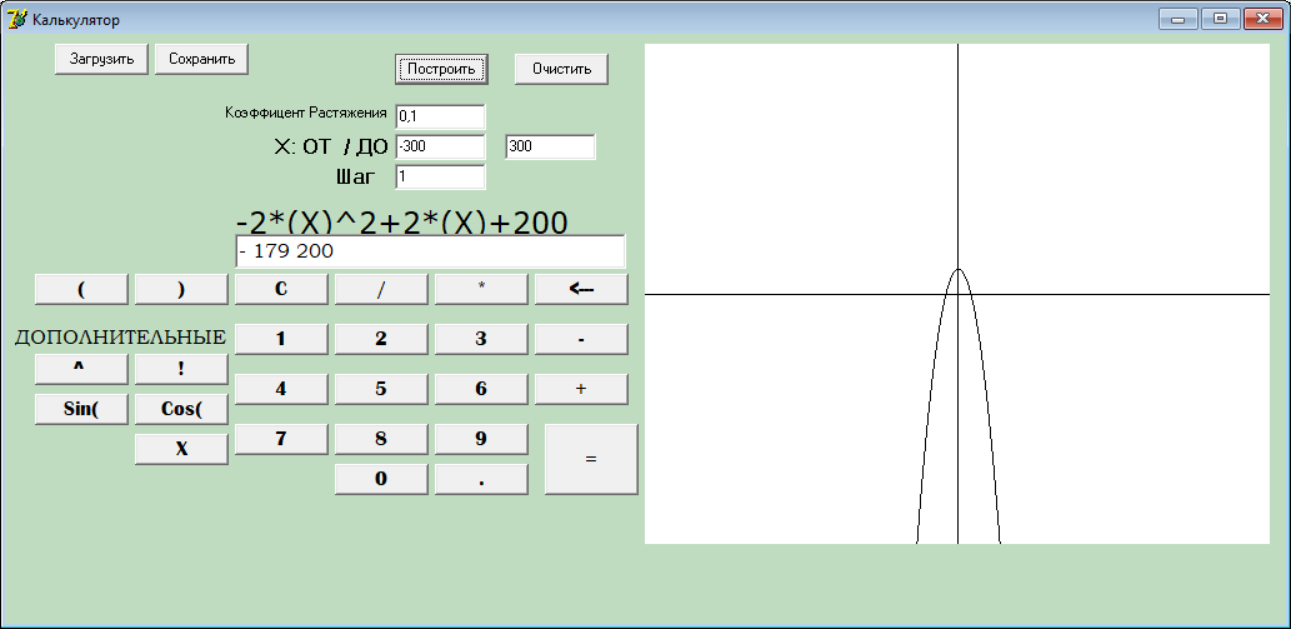


Рисунок 5.11

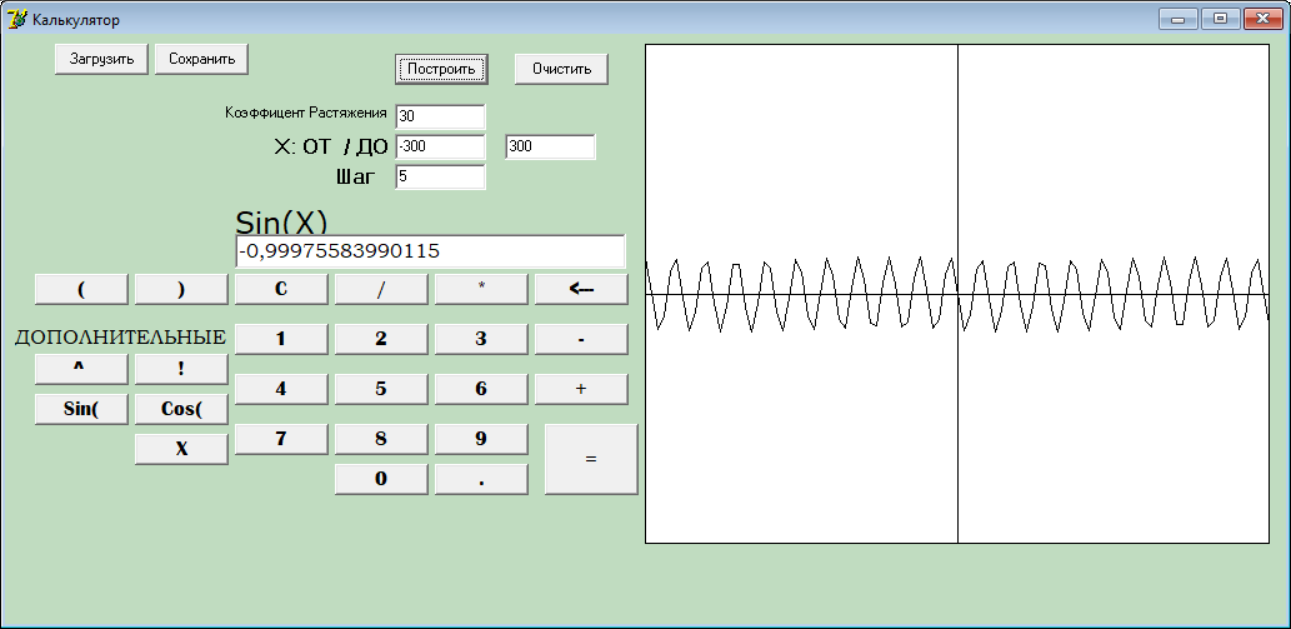


Рисунок 5.12

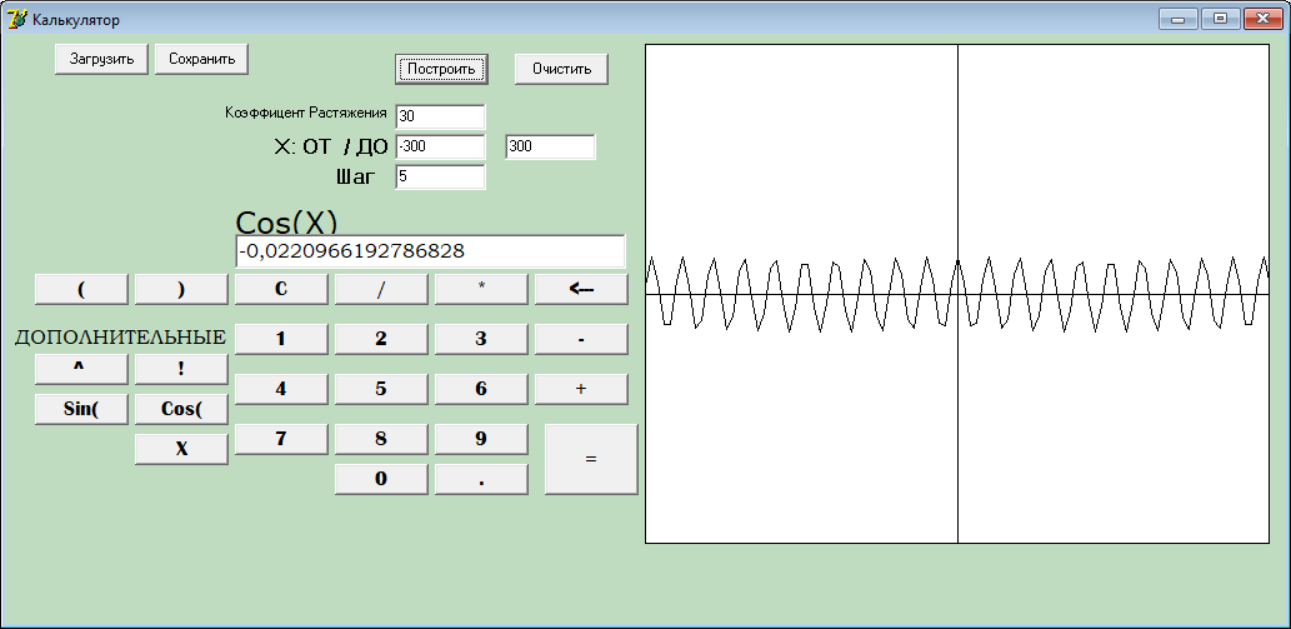


Рисунок 5.13

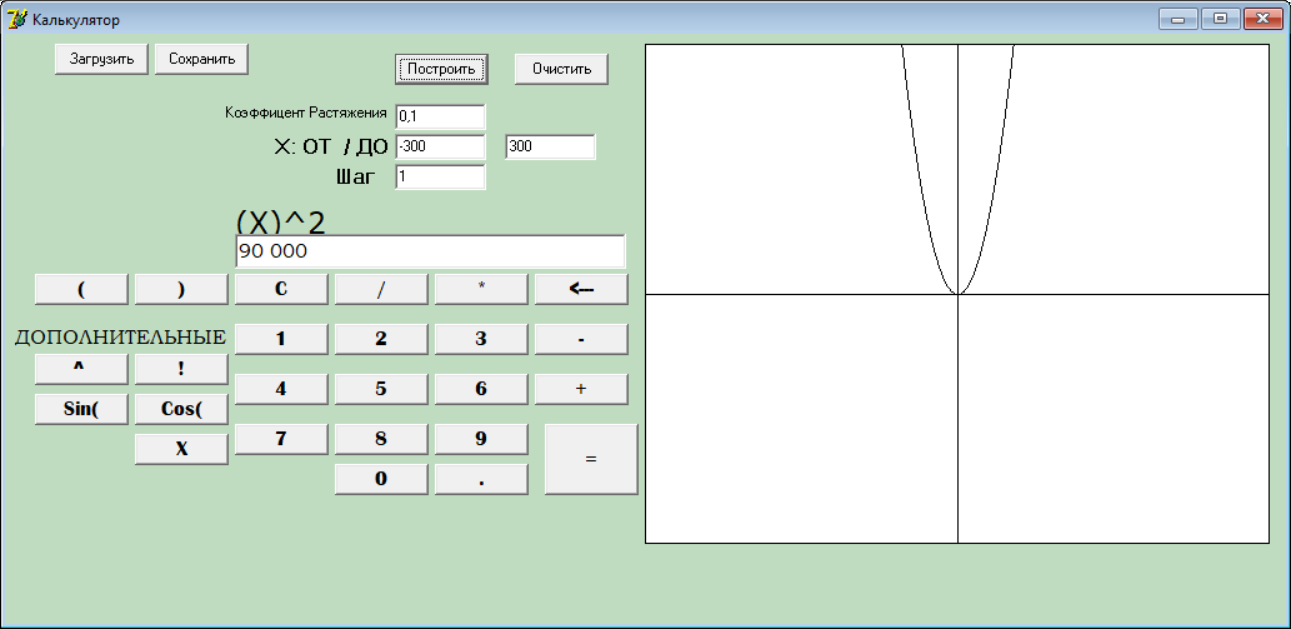


Рисунок 5.14

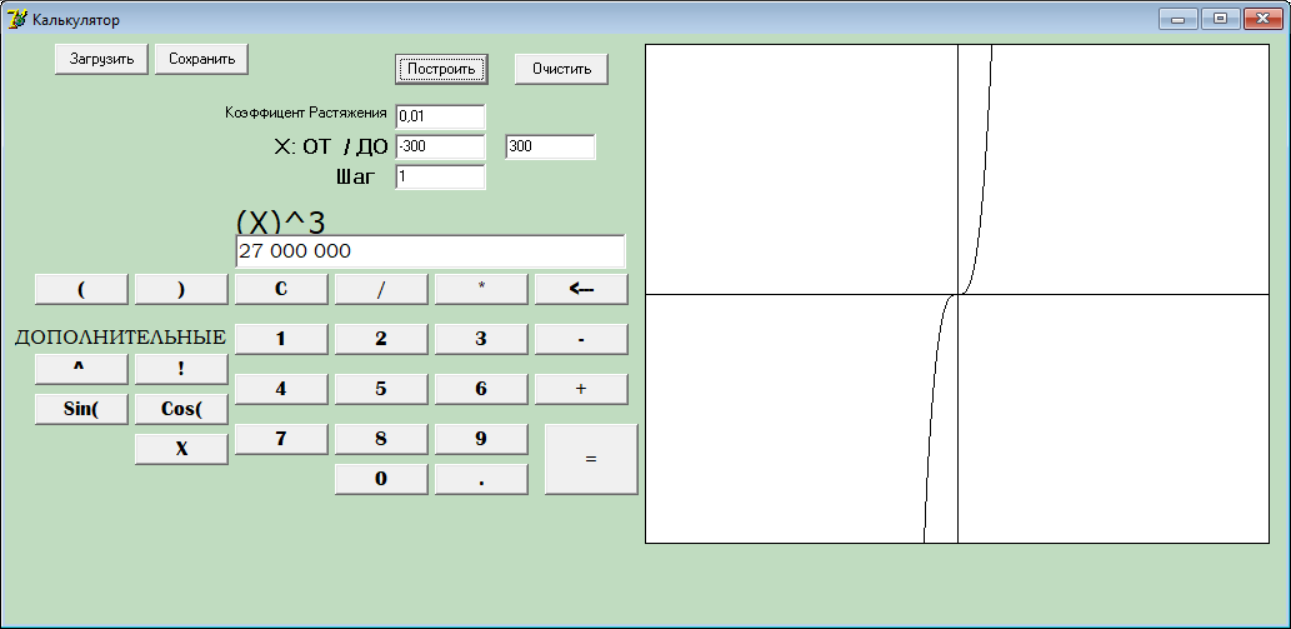


Рисунок 5.15

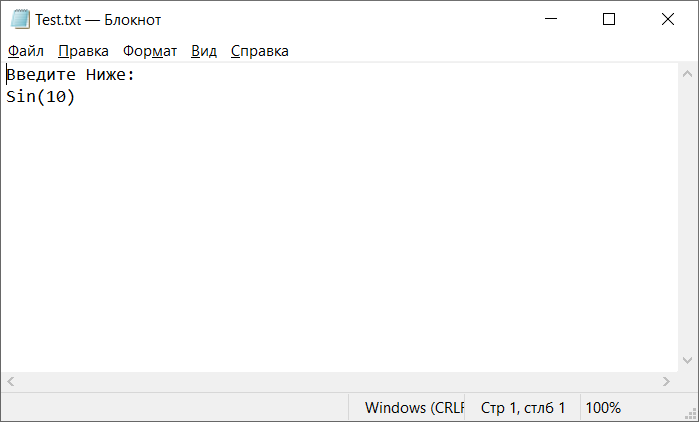


Рисунок 5.16

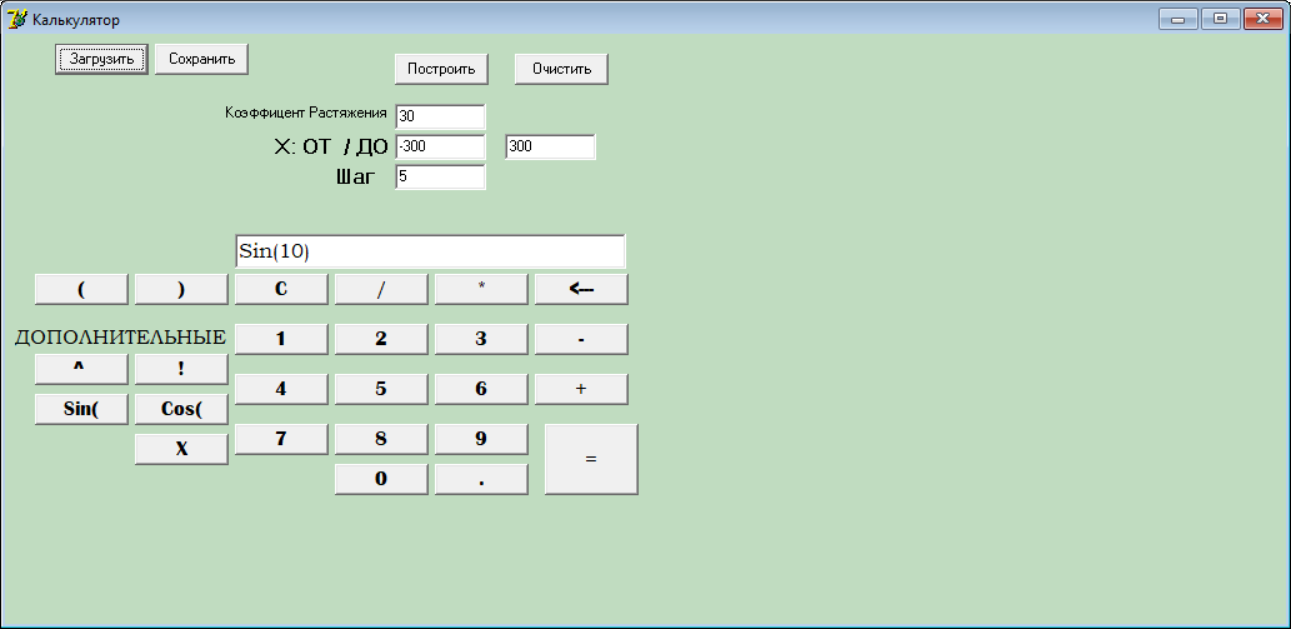


Рисунок 5.16.1

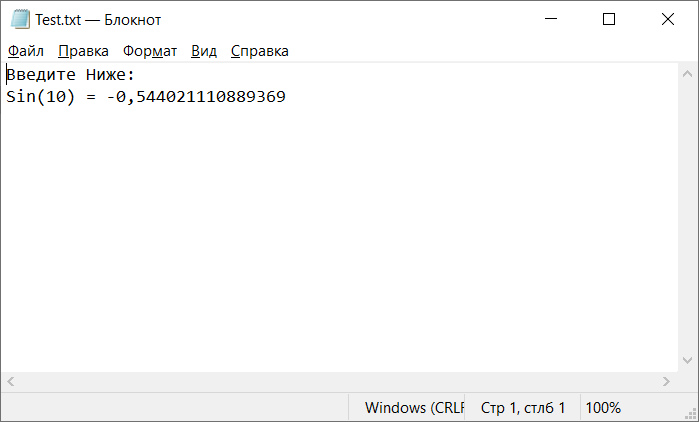


Рисунок 5.17

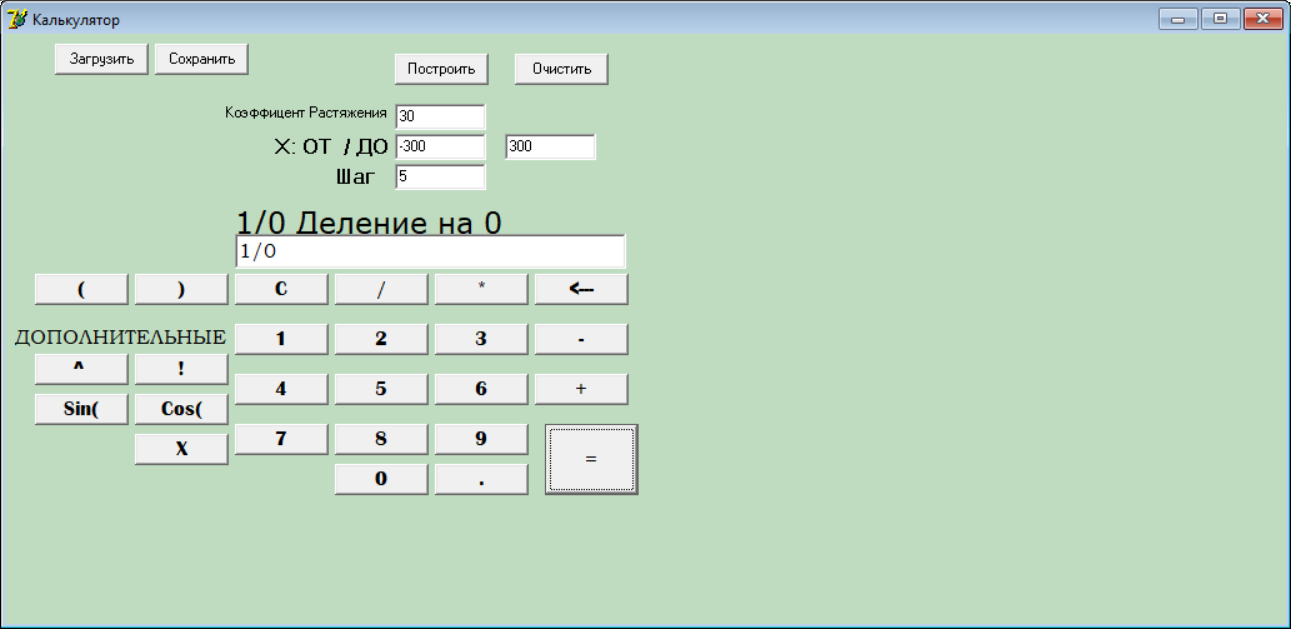


Рисунок 5.18

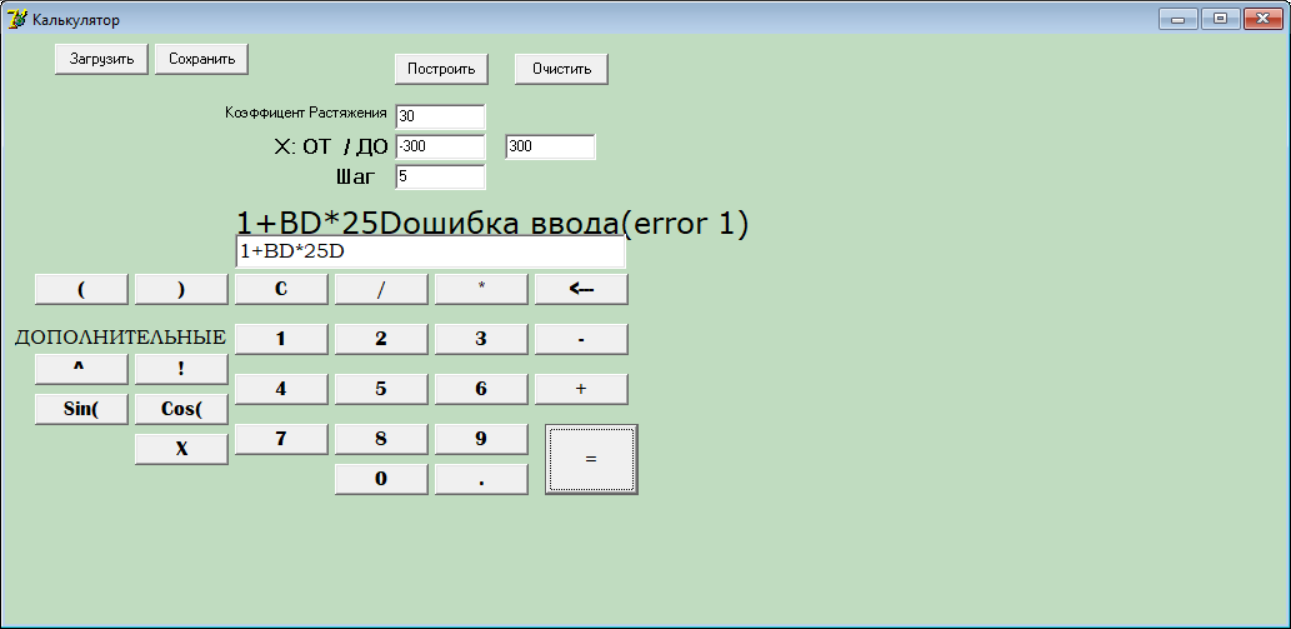


Рисунок 5.19

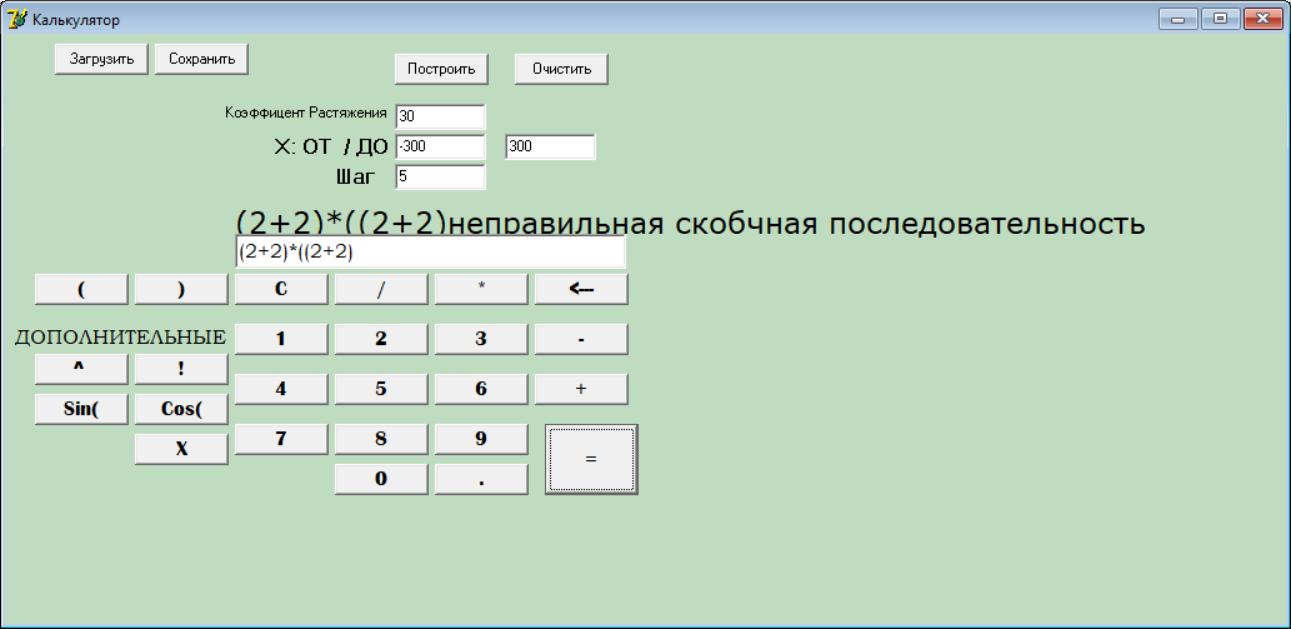


Рисунок 5.20

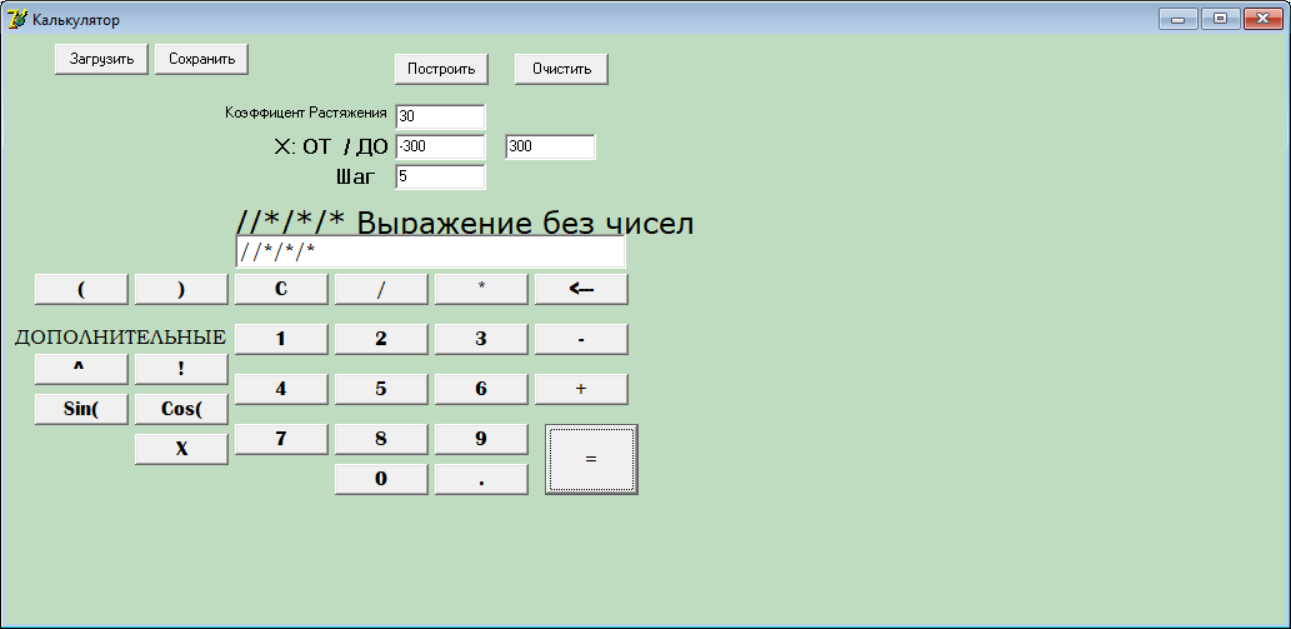


Рисунок 5.21

# 6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЮ

## 6.1 Начало работы

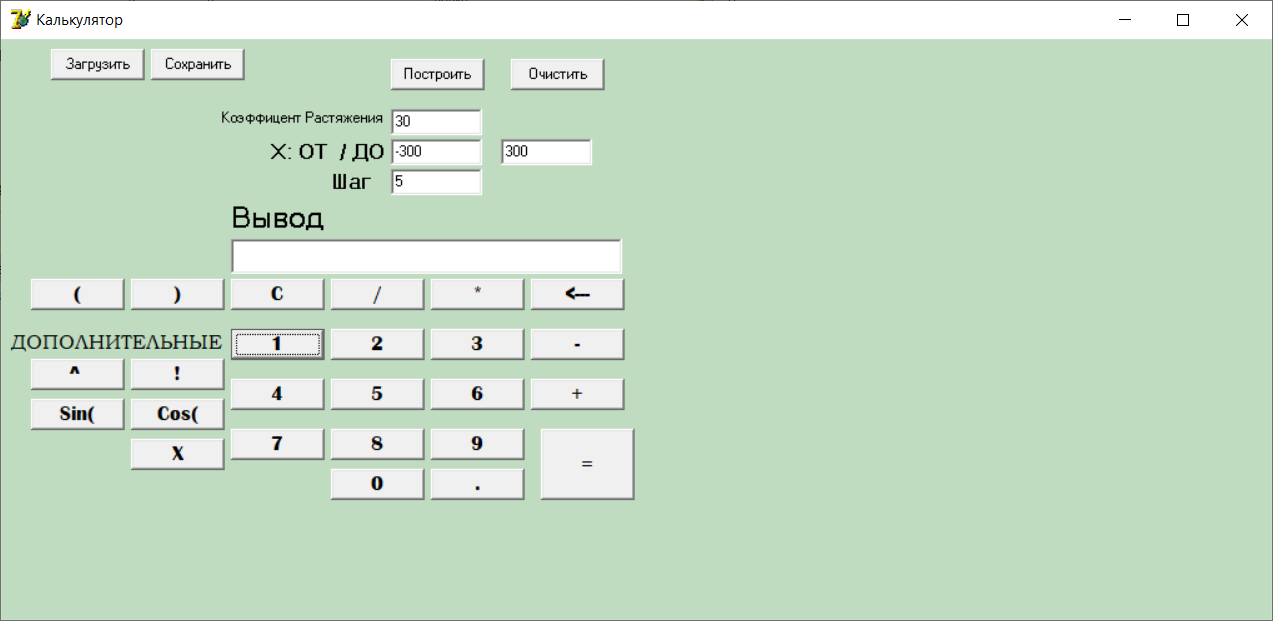
Для того, чтобы начать использование программы, необходимо запустить файл Graph.exe. После открытия программы появится окно, показанное на рисунке 6.1.

Рисунок 6.1 – окно основной программы.

После открытия главного меню программного средства, пользователь может сразу же приступить к работе с приложением.

Инструкция по эксплуатации программы:

* Введите первое число в плде ввода (рисунок 6.2);
* Нажмите кнопку «+» (сложение), «-» (вычитание), «\*» (умножение) или «/» (деление), тригонометрические функции и др.;
* Введите следующее число;
* Введите необходимый оператор;
* Введите остальные числа и операторы
* Нажмите кнопку «=» (рисунок 6.2.1);

Рисунок 6.2 – Окно ввода

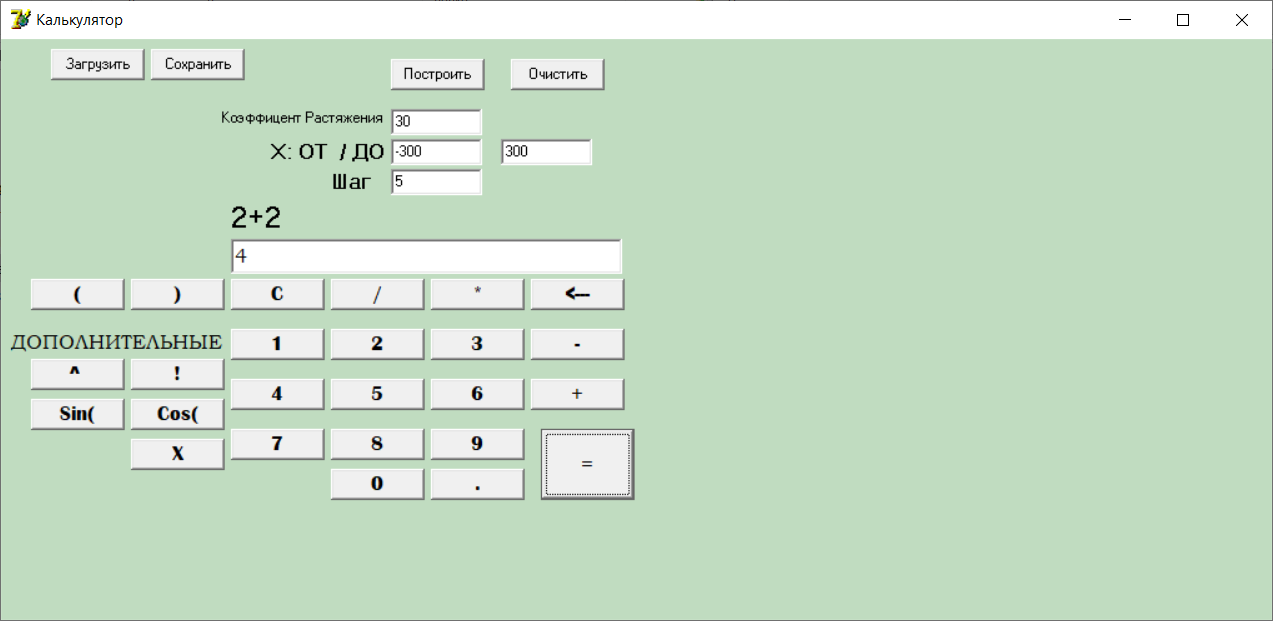


Рисунок 6.2.1 – Подсчет выражения

Для визуализации введенного выражения используйте кнопку построить, предварительно выставив коэффициент Растяжения, X: от/до, Шаг. По умолчанию выставлены значения Коэффициента растяжения = 30; X: от -300 до 300 и шаг = 5. Коэффициент растяжения отвечает за приближение и отдаление графика, X: от/до отвечает за границы X, шаг отвечает за промежуток через который будет следующая точка X. Пример построения графика Cos(Sin(X)), скриншот которого отображен на рисунке 6.3. С помощью кнопки очистить, очищается поле графика и очищается поля ввода выражения.

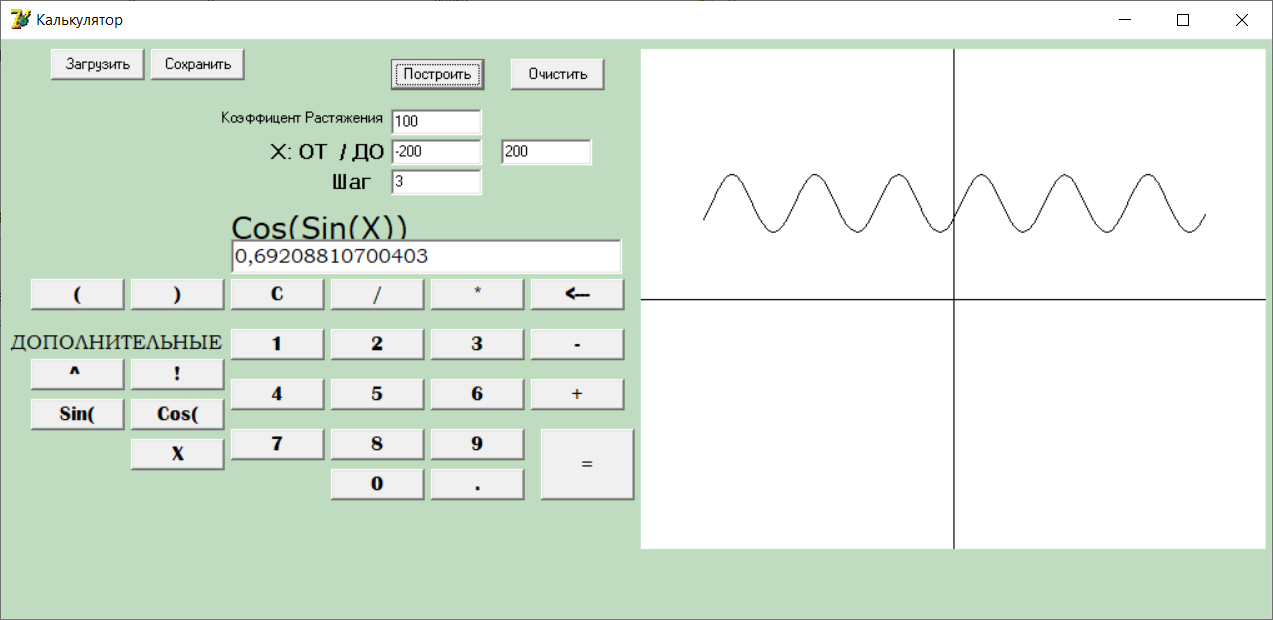
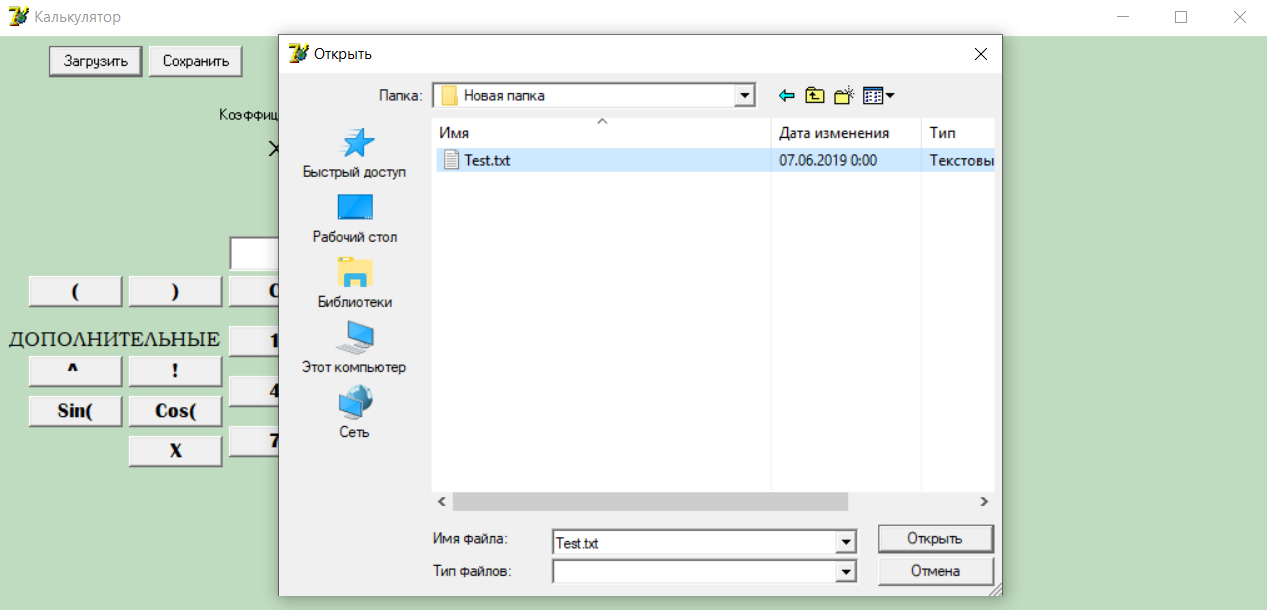


Рисунок 6.3 – Построения графика

Для сохранения либо загрузки из текстового файла используйте соответственно кнопки Сохранить и загрузить. Для этого нажмите на соответствующую кнопку и выберите подходящий текстовый файл, как показано на рисунке 6.4.

Рисунок 6.4 – Загрузка / Сохранение в файл

## 6.2 Завершение работы программы

Логическим завершением работы программы является сохранение результата вычисления сохранения в текстовый файл.

Выход из программы осуществляется одним способом в программе: по нажатию кнопки «Выход» в верхнем правом углу программы рисунок 6.5



Рисунок 6.5 – Верхний угол программы

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта было разработано приложение, имеющий достаточный функционал для нахождения значения выражения и визуализации данноговыражения. Программа содержит не только основные функции, но и ряд дополнительных, способствующих расширить функционал программного средства и улучшить взаимодействие с пользователем.

Простой и удобный пользовательский интерфейс позволяет пользователю легко привыкнуть к приложению.

Написанный код легко модифицируется для добавления новых функций. В дальнейшем возможны улучшения и доработки, вносящие новый функционал в программу.

В процессе разработки данного программного средства мной были изучены основные принципы работы с жинамическим стеком на языке программирования Delphi, принципы работы с файлами, принципы разработки простейшего пользовательского интерфейса. Знания, полученные в ходе изучения курса ОАиП были закреплены, и позволят применять их при создании новых проектов.

# СПИСОК ЛИТЕРАТУРЫ

[1] Глухова, Л.А. Основы алгоритмизации и программирования: учебное пособие. В 2 Ч. / Л.А. Глухова. – БГУИР, 2006 – Ч. 1. – 195 с.

[2] Тюкачев Н.А. Программирование графики в Delphi: учебное пособие. / Н.А. Тюкачев, И.В. Илларионов, В.Г. Хлебострелов. – СПб. : БХВ-Петербург, 2008. – 784с.

[3] Фараонов В.В. Delphi 6: учебное пособие. /В.В.Фараонов. – М.: Издатель Молгачёва С.В., 2003. – 672 с.

[4] Алгоритм перевода в обратную польскую запись [Электронный ресурс] Режим доступа: <https://habr.com/ru/post/100869/>

[5] Обратная польская запись [Электронный ресурс] Режим доступа: <https://foxford.ru/wiki/informatika/obratnaya-polskaya-notatsiya>

# 

# ПРИЛОЖЕНИЕ 1

**Исходный код программы**

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Grids, StdCtrls, ExtCtrls ;

type

TForm1 = class(TForm)

Button1: TButton;

Label1: TLabel;

Button2: TButton;

Button3: TButton;

Button4: TButton;

Button5: TButton;

Button6: TButton;

Button7: TButton;

Button8: TButton;

Button9: TButton;

Button10: TButton;

Button11: TButton;

Button12: TButton;

Button13: TButton;

Button14: TButton;

Button15: TButton;

Button16: TButton;

Button17: TButton;

Button18: TButton;

Button19: TButton;

Button20: TButton;

Button22: TButton;

Button23: TButton;

Label2: TLabel;

Button21: TButton;

Label3: TLabel;

Button24: TButton;

A: TEdit;

Button26: TButton;

DrawGraph: TImage;

Button27: TButton;

Button28: TButton;

Edit1: TEdit;

Edit2: TEdit;

Edit3: TEdit;

Label4: TLabel;

Label5: TLabel;

Edit4: TEdit;

Label6: TLabel;

OpenDialog1: TOpenDialog;

Button30: TButton;

Button31: TButton;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure Button6Click(Sender: TObject);

procedure Button7Click(Sender: TObject);

procedure Button8Click(Sender: TObject);

procedure Button9Click(Sender: TObject);

procedure Button10Click(Sender: TObject);

procedure Button11Click(Sender: TObject);

procedure Button12Click(Sender: TObject);

procedure Button14Click(Sender: TObject);

procedure Button15Click(Sender: TObject);

procedure Button16Click(Sender: TObject);

procedure Button17Click(Sender: TObject);

procedure Button13Click(Sender: TObject);

procedure Button19Click(Sender: TObject);

procedure Button23Click(Sender: TObject);

procedure Button22Click(Sender: TObject);

procedure Button18Click(Sender: TObject);

procedure Button20Click(Sender: TObject);

procedure Button21Click(Sender: TObject);

procedure Button24Click(Sender: TObject);

procedure Button25Click(Sender: TObject);

procedure AKeyPress(Sender: TObject; var Key: Char);

procedure Button26Click(Sender: TObject);

procedure Button27Click(Sender: TObject);

procedure DrawGraphClick(Sender: TObject);

procedure Button28Click(Sender: TObject);

procedure Button30Click(Sender: TObject);

procedure Button31Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

const

prs = '+-\*/(';

pri: array [1 .. 5] of byte = (1, 1, 2, 2, 0);

var

s, s1, s2, s3: String;

w: array of Char;

n, len, len2: int64;

t: extended;

ch: Char;

b,q3,c1: boolean;

k1: integer;

Function Count0 (s4:string):string; Forward;

Procedure PreCount(var s1:string; var g:boolean); Forward;

/////////////////////////////////////////////////////////////////calculations start

procedure DeleteSpaces(var a:string); //удаление пробелов из строки

var p: integer;

Begin

setconsolecp(1251);

setconsoleoutputcp(1251);

p := pos(' ',a);

while p <> 0 do

begin

delete(a, p, 1);

p := pos(' ', a);

end;

End;

procedure vizual (var s: string ); // красивое представление ответа(строки)

var

i:integer;

begin

if pos('.', s) <> 0 then

begin

i:=pos('.', s)-1;

while i>3 do

begin

i:=i-3;

insert(' ',s,i+1);

end;

end

else

if pos(',', s) <> 0 then

begin

i:=pos(',', s)-1;

while i>3 do

begin

i:=i-3;

insert(' ',s,i+1);

end;

end

else

begin

i:=length(s);

while i>3 do

begin

i:=i-3;

insert(' ',s,i+1);

end;

end;

end;

////////////////////////////////////////////////////////////////////////////Cos

function Cos(var x: Real): Real; //подсчет косинуса

var

Prev, Ch, Zn, Sign: Real;

dPi, n: Integer;

begin

dPi:= Trunc(x/pi);

if Abs(dPi mod 2) = 1 then

if x > 0 then

x:= x - (dPi - 1) \* pi

else

x:= x - (dPi - 1) \* pi + 2 \* pi

else

if x > 0 then

x:= x - dPi \* pi

else

x:= x - dPi \* pi + 2 \* pi;

Sign:= 1;

n:= 1;

Result:= 0;

Prev:= 1;

Ch:= 1;

Zn:= 1;

while (Abs(Result - Prev) > 0.0000000000001) do

begin

Prev:= Result;

Result:= Result + Sign \* Ch / Zn;

Sign:= -Sign;

Ch:= Ch \* x \* x;

Zn:= Zn \* (n) \* (n + 1);

n:= n + 2;

end;

end;

Procedure PreCos (var Str:string); //нахождение в строке Cos( и передача в функцию Cos

var s,s2,s3:string;

k,i,n,n1,n2,p,g,chisl,i2:integer;

k3,k2:real;

a:integer;

begin

p:=pos('Cos(',str);

If p<>0 then

while p<>0 do

Begin

str:=StringReplace(str, '.', ',', [rfReplaceAll, rfIgnoreCase]);

k:= length(Str);

s:=''; s2:='';

n:=0;

n1:=0;

n2:=0;

for i := k downto 1 do

begin

if (str[i-4] ='C') and (str[i-3] ='o') and (str[i-2] ='s') and (str[i-1] ='(')then

begin

g:=i;

while g<=k do

begin

if str[g] in (['1','2','3','4','5','6','7','8','9','0',',','-','+','\*','/']) then s2:=s2+str[g];

if str[g+1] in [')'] then break else inc(g);

end;

delete(Str, i-4, length(s2) + 5);

s2:= Count0(s2);

k3:= StrToFloat(s2);

if (k3<0.001) and (k3>-0.001) then k2:=1 else k2:= Cos(k3);

if length(str)>0 then

for i2:=0 to length(str) do

begin

if str[i2]='(' then inc(chisl);

if str[i2]=')' then dec(chisl);

end;

if (length(str)>0) and (Str[length(str)]=')') and (chisl<0) then delete(Str,length(str),1);

s3:=FloatToStr(k2);

Insert(s3, str, i-4);

end;

s2:='';

end;

p:=pos('Cos(',str);

End;

str:=StringReplace(str, ',', '.', [rfReplaceAll, rfIgnoreCase]);

end;

////////////////////////////////////////////////////////////////////////////// SIn

function Sin(var x: Real): Real; // подсчет синуса

var

Prev, Ch, Zn, Sign: Real;

dPi, n: Integer;

begin

dPi:= Trunc(x/pi);

if Abs(dPi mod 2) = 1 then

if x > 0 then

x:= x - (dPi - 1) \* pi

else

x:= x - (dPi - 1) \* pi + 2 \* pi

else

if x > 0 then

x:= x - dPi \* pi

else

x:= x - dPi \* pi + 2 \* pi;

Sign:= 1;

n:= 1;

Result:= 0;

Prev:= 1;

Ch:= x;

Zn:= 1;

while (Abs(Result - Prev) > 0.0000000000001) do

begin

Prev:= Result;

Result:= Result + Sign \* Ch / Zn;

Sign:= -Sign;

Ch:= Ch \* x \* x;

Zn:= Zn \* (n + 1) \* (n + 2);

n:= n + 2;

end;

end;

Procedure PreSin (var Str:string); //нахождение в строке sin( и передача в функцию Sin

var s,s2,s3:string;

k,i,n,n1,n2,p,g:integer;

i2,chisl:integer;

k3,k2:real;

begin

p:=1; k2:=0; k3:=0;

p:=pos('Sin(',str);

If p<> 0 then

while p<>0 do

Begin

str:=StringReplace(str, '.', ',', [rfReplaceAll, rfIgnoreCase]);

p:=pos('Sin(',str); p:=p+4;

k:= length(Str);

s:=''; s2:='';

n:=0;

n1:=0;

n2:=0;

for i := k downto 1 do

begin

if (str[i-4] ='S') and (str[i-3] ='i') and (str[i-2] ='n') and (str[i-1] ='(')then

begin

g:=i;

while (g<=k\*2) do

begin

if str[g] in (['0'..'9',',','-','+','\*','/']) then s2:=s2+str[g];

if str[g+1] in [')'] then break else inc(g);

end;

delete(Str, i-4, length(s2) + 5);

s2:= Count0(s2);

k3:= StrToFloat(s2);

if (k3<0.001) and (k3>-0.001) then k2:=0 else k2:= Sin(k3);

if length(str)>0 then

for i2:=0 to length(str) do

begin

if str[i2]='(' then inc(chisl);

if str[i2]=')' then dec(chisl);

end;

if (length(str)>0) and (Str[length(str)]=')') and (chisl<0) then delete(Str,length(str),1);

s3:=FloatToStr(k2);

Insert(s3, str, i-4);

end;

s2:='';

end;

p:=pos('Sin(',str);

End;

str:=StringReplace(str, ',', '.', [rfReplaceAll, rfIgnoreCase]);

end;

/////////////////////////////////////////////////////////////////////////////STepen

procedure Stepen( n,n1:integer; out s:string); // подсчет степени

var

i:real;

i1:integer;

begin

s:=''; i:=1;

if n1=0 then begin s:='0'; exit; end;

if n1<0 then

Begin

n1:=n1\*(-1);

if (n mod 2) = 0 then i:=1 else i:=-1;

end;

i:=exp(n\*ln(n1))\*i;

s:=FloatToStr(i);

end;

Procedure preStepen (var Str:string); //нахождение в строке ^( и передача в функцию Stepen

var s,s2:string;

k,i,n,n1,n2,n3,p,g:integer;

begin

p:=pos('^',str);

while p<>0 do

Begin

k:= length(Str);

s:=''; s2:='';

n:=0;

n1:=0;

n2:=0;

for i := 1 to k do

begin

if str[i] ='^' then

begin

g:=i;

while g<=k do

begin

if str[g+1] in (['1','2','3','4','5','6','7','8','9','0','(','-']) then s2:=s2+str[g+1];

if str[g+2] in [')','-','+','\*','/'] then break else inc(g);

end;

end;

end;

for i := 1 to k do

begin

if str[i] in (['1','2','3','4','5','6','7','8','9','0','(','-']) then s:=s+str[i];

if (str[i] in (['+','\*','/'])) or ((str[i] ='-') and (str[i-2] <> '(')) then s:='';

if (str[i-1] in [')']) and (str[i] in ['^']) then begin s:=s+')'; n:=length(s); s:= Count0(s); end else n:=length(s);

n3:=n;

if (str[i] ='^')and (length(s)>0 ) then

begin

delete(Str, p-n, n+1+length(s2));

n1:=StrToInt(s);

n:=StrToInt(s2);

Stepen(n,n1,s);

Insert(s, str, p-n3);

End;

end;

p:=pos('^',str);

End;

end;

/////////////////////////////////////////////////////////////////////factorial

var

a:array[1..500] of integer;

procedure Factorial2(const n:integer;out s:string); // подсчет факториала

var

i,j,fl:integer;

begin

for i:=1 to 500 do

a[i]:=0;

s:='';

a[1]:=1;

for i:=2 to n do

begin

for j:=1 to 500 do

a[j]:=a[j]\*i;

for j:=1 to 500 do

if a[j]>999999 then

begin

a[j+1]:=a[j+1]+a[j] div 1000000;

a[j]:=a[j] mod 1000000;

end;

end;

fl:=1;

for i:=500 downto 1 do

begin

if fl=0 then

Begin

if a[i]<10000 then s:=s+'00';

if a[i]<1000 then s:=s+'00';

if a[i]<100 then s:=s+'00';

if a[i]<10 then s:=s+'00';

s:=s+IntToStr(a[i]);

end;

if (a[i]>0) and (fl=1) then

begin

fl:=0;

s:=s+IntToStr(a[i]);// writeln('s2!=',s);

end;

end;

// writeln('s!=',s);

end;

Procedure PreFactorial (var Str:string); //нахождение в строке ! и передача в функцию Factorial

var s:string;

k,i,n,n1,p:integer;

begin

p:=pos('!',str);

while p<>0 do

Begin

k:= length(Str);

s:='';

n:=0;

n1:=0;

for i := 1 to k do

begin

if str[i] in (['0'..'9','(',')']) then s:=s+str[i];

if str[i] in ['+','-', '\*', '/'] then s:='';

if (str[i] ='!')and (length(s)>0 ) then

begin

n:=length(s);

s:= Count0(s);

delete(Str, p-n, n+1);

n1:=StrToInt(s);

Factorial2(n1,s);

Insert(s, str, p-n);

End;

end;

p:=pos('!',str);

End;

end;

procedure PreX(Var S:string); // вместь X поставить число

var

p:integer;

Begin

if not b then k1:=StrToInt(Form1.Edit1.Text) else k1:=k1+StrToInt(Form1.Edit2.Text); // увеличиваем на шаг

b:=true;

While pos('X',s)<>0 do

Begin

p:=pos('X',s);

Delete(s,p,1);

Insert(IntToStr(k1) ,s,p);

End;

End;

///////////////////////////////////////////////////////////////////// MAIN

type //стек на динамическом списке

TPElem = ^TElem;

TElem = record

Data : extended;

PNext : TPElem;

end;

var

PSt1, PElem : TPElem;

//Добавление элемента на вершину стека.

procedure StPush(var aPStack, aPElem : TPElem);

begin

if aPElem = nil then Exit;

aPElem^.PNext := aPStack;

aPStack := aPElem

end;

//Изъятие элемента с вершины стека.

function StPop(var aPStack, aPElem : TPElem) : extended;

begin

if aPStack = nil then Exit;

aPElem := aPStack;

aPStack := aPElem^.PNext;

Result := aPElem^.Data;

end;

procedure PushC(x: Char); //стек на операций на динамическом массиве

begin

SetLength(w,length(w)+1);

Inc(len2);

w[len2] := x; //знаки

end;

function Popc: Char;

begin

Popc := w[len2];

Dec(len2);

SetLength(w,length(w)-1);

end;

function vychislenie(s1, s2: extended; s3: Char): extended; //вычисление значения операндов s1,s2 и опреации s3

var

x, y, z: extended;

begin

x := s1;

y := s2;

case s3 of

'+': z := x + y;

'-': z := x - y;

'\*': z := x \* y;

'/': if y=0 then begin Form1.label1.Caption:=Form1.label1.Caption+'Было деление на 0'; vychislenie := 0; exit; end else

z := x / y;

end;

vychislenie := z;

end;

procedure PreChange(var s: String; out g:boolean); // проверка строки на ошибки

var

i,k,Scob,i2: integer;

begin

i2:=0;

g:=true;

n:=length(s);

if s[1] = '-' then

s := '0' + s

else

if s[1] = '0' then

s := '0+' + s;

k:=0;

for i:= 1 to n do

Begin

if ( (s[i] = '\*') or (s[i] = '/') ) and (s[i + 1] in ['+','-']) then

Begin

insert('(', s, i + 1);

k:=i+2;

repeat

inc(k);

if s[k+1] in (['0'..'9']) then

g:=true else g:=false;

until not g;

// end;

insert(')', s, k + 1);

end;

End;

g:=true;

n:=length(s);

Scob:=0;

for i:=1 to n do

Begin

If s[i] = '(' then inc(Scob);

If s[i] = ')' then dec(Scob);

End;

if Scob <>0 then

Begin

Form1.label1.Caption:=Form1.label1.Caption+'неправильная скобчная последовательность';

g:=false;

exit;

end;

if (length(S1)=1) and (not (s[1] in (['0'..'9']))) then

begin

Form1.label1.Caption:=Form1.label1.Caption+'ошибка ввода(error 3) ';

g:=false;

exit;

End;

i:=1;

while i <= n do

Begin

if (s[i] = '(') and (s[i + 1] = '-') then

insert('0', s, i + 1);

if (s[i] = '+') and ((s[i + 1] = '-') or (s[i + 1] = '+')) then

insert('0', s, i + 1);

if (s[i] = '-') and (s[i + 1] = '-') then

begin

delete(s, i, 2);

insert('+', s, i );

end;

if not (s[i] in (['0'..'9','+','-','\*','/','.',',','(',')','C','o','s','S','i','n','!','^','E'])) then

begin

Form1.label1.Caption:=Form1.label1.Caption+'ошибка ввода(error 1) ';

g:=false;

exit;

end;

if ((s[i] = 'C') and (s[i+1] <> 'o') and (s[i+2] <> 's')) or ((s[i] = 'S') and (s[i+1] <> 'i') and (s[i+2] <> 'n')) then

begin

Form1.label1.Caption:=Form1.label1.Caption+'ошибка ввода(error 2) ';

g:=false;

exit;

end;

if (s[i] = '/') and (s[i + 1] = '0') and (s[i + 2] <> '.') and (s[i + 2] <> ',') then

begin

Form1.label1.Caption:=Form1.label1.Caption+' Деление на 0 ';

g:=false;

exit;

end;

if (s[i] in (['0'..'9','X'])) then inc(i2);

if (i=length(s)) and (i2=0) then

begin

Form1.label1.Caption:=Form1.label1.Caption+' Выражение без чисел ';

g:=false;

exit;

end;

Inc(i);

end;

end;

function Change(s: String): String; // составляем обратную польскую запись

var

i: integer;

rezs: String;

c: Boolean;

begin

c := false;

for i := 1 to n do

begin

if not(s[i] in ['+', '-', '\*', '/', '(', ')']) then

begin

if c then rezs := rezs + ' ';

rezs := rezs + s[i];

c := false;

end

else

begin

c := true;

if s[i] = '(' then

PushC(s[i])

else

if s[i] = ')' then

begin

while w[len2] <> '(' do

begin

rezs := rezs + ' ' + Popc;

end;

Popc;

end

else

if s[i] in ['+', '-', '\*', '/'] then

begin

while pri[Pos(w[len2], prs)] >= pri[Pos(s[i], prs)] do

rezs := rezs + ' ' + Popc;

PushC(s[i]);

end;

end;

end;

while len2 <> 0 do

rezs := rezs + ' ' + Popc;

Result:=rezs;

end;

//////////////////////////////////////////////////////////////////////// из строки s2 делаем число (f)

function Count(s: String): extended; //подсчет строки (с обр польской записью)

var

ss: String;

s1, s2: extended;

x:extended;

chh, s3: Char;

p: extended;

i, j: integer;

tmp: Integer;

begin

i := 0;

repeat

j := i + 1;

repeat

Inc(i)

until s[i] = ' ';

ss := copy(s, j, i - j); //числа из строки

chh := ss[1];

if not(chh in ['+', '-', '\*', '/']) then

begin

Val(ss, p, tmp); //слово в целое число

New(PElem);

PElem^.Data := p;

StPush(PSt1, PElem); //заносим в стек числа

end

else

begin

s2 := StPop(PSt1, PElem);

s1 := StPop(PSt1, PElem);

s3 := chh;

New(PElem);

PElem^.Data := vychislenie(s1, s2, s3);

StPush(PSt1, PElem);

end;

until i >= n;

x := StPop(PSt1, PElem);

Count := x;

end;

/////////////////////////////////////////////////////////////////main \_ end

////////////////////////////////////////////////////////////////calculations end/ visual start

procedure TForm1.Button1Click(Sender: TObject);

begin

s1:=s1+'1';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

s1:=s1+'2';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button3Click(Sender: TObject);

begin

s1:=s1+'3';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button4Click(Sender: TObject);

begin

s1:=s1+'4';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button5Click(Sender: TObject);

begin

s1:=s1+'5';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button6Click(Sender: TObject);

begin

s1:=s1+'6';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button7Click(Sender: TObject);

begin

s1:=s1+'7';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button8Click(Sender: TObject);

begin

s1:=s1+'8';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button9Click(Sender: TObject);

begin

s1:=s1+'9';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button10Click(Sender: TObject);

begin

s1:=s1+'0';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button12Click(Sender: TObject);

begin

s1:=s1+'+';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button14Click(Sender: TObject);

begin

b:=False;

c1:=false;

k1:=1;

s1:='';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button15Click(Sender: TObject);

begin

s1:=s1+'/';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button16Click(Sender: TObject);

begin

s1:=s1+'\*';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button17Click(Sender: TObject);

begin

delete(s1,length(s1),1);

A.Text:=s1;

label1.Caption:= s1;

end;

procedure TForm1.Button13Click(Sender: TObject);

begin

s1:=s1+'-';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button19Click(Sender: TObject);

begin

s1:=s1+'!';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button23Click(Sender: TObject);

begin

s1:=s1+')';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button22Click(Sender: TObject);

begin

s1:=s1+'(';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button18Click(Sender: TObject);

begin

s1:=s1+'.';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button20Click(Sender: TObject);

begin

s1:=s1+'^';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button21Click(Sender: TObject);

begin

s1:=s1+'Sin(';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button24Click(Sender: TObject);

var str:string;

g,i:integer;

begin

s1:=s1+'Cos(';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button26Click(Sender: TObject);

begin

s1:=s1+'X';

label1.Caption:=s1;

A.Text:=s1;

end;

procedure TForm1.Button25Click(Sender: TObject);

begin

s1:=s1+'Ln(';

label1.Caption:=s1;

A.Text:=s1;

end;

Procedure PreCount(var s1:string; var g:boolean);

Begin

SetLength(w,1);

DeleteSpaces(s1);

s1:=StringReplace(s1, ',', '.', [rfReplaceAll, rfIgnoreCase]);

PreX(s1);

PreChange(s1,g);

if g=false then exit;

PreFactorial(s1);

preStepen(s1);

if pos('C',s1) > pos('S',s1) then

Begin

PreCos(s1);

PreSin(s1);

end else

Begin

PreSin(s1);

PreCos(s1);

end;

end;

Function Count0 (s4:string):string;

var g:boolean;

Begin

s4:=StringReplace(s4, ',', '.', [rfReplaceAll, rfIgnoreCase]);

n := Length(s4);

PreChange(s4,g);

if g=false then exit; //добавляет 0 перед -

n := Length(s4);

s2 := Change(s4); //составляем rezs = s2 Составляем обратную польскую запись

if s2[1] = ' ' then

delete(s2, 1, 1);

s2 := s2 + ' ';

n := Length(s2);

t := Count(s2); // считаем строку

s4:=FloatToStr(t);

result:=s4;

ENd;

procedure TForm1.Button11Click(Sender: TObject);

var

p:integer;

g:boolean;

begin

if length(s1) <=0 then exit;

p:=pos('X',s1);

if p <>0 then q3:=true;

PreCount(s1,g); //удаляем пробелы, считаем X, считаем factorial, stepen, Sin, Cos,

if g=false then exit;

s1:=Count0(s1); // считаем строку(выражение)

if g=false then exit;

if q3 then //рисуем график функции

Begin

Form1.Edit4.Text:=StringReplace(Form1.Edit4.Text, '.', ',', [rfReplaceAll, rfIgnoreCase]);

If not c1 then DrawGraph.Canvas.MoveTo(k1 + round(DrawGraph.Width/2), Round(-1\*t\*StrToFloat(Form1.Edit4.Text)+ DrawGraph.height/2) );

If c1 then DrawGraph.Canvas.LineTo(k1 + round(DrawGraph.Width/2),Round(-1\*t\*StrToFloat(Form1.Edit4.Text)+ DrawGraph.height/2) );

End;

vizual(s1);

A.Text:=s1;

end;

procedure TForm1.AKeyPress(Sender: TObject; var Key: Char);

begin

Case Key of

#13: Button11Click(Self);

#8: Button17Click(Self);

Else

s1:=s1 + key; //записываем нажатую кнопку в строку

A.Text:=s1;

label1.Caption:=s1;

key:= #0;

A.SelStart:= Length(A.Text);

end;

key:= #0;

A.SelStart:= Length(A.Text);

end;

procedure TForm1.Button27Click(Sender: TObject); // кнопка построить

var i2:integer;

begin

if (pos('!',s1)<>0 ) and ((StrToInt(Form1.Edit1.Text)<0) or (StrTOint(Form1.Edit3.Text)>20)) then

Begin

Form1.label1.Caption:='при этом X нельзя построить график';

exit;

End;

DrawGraphClick(Self);

s3:=s1; q3:=true;

k1:=StrToInt(Form1.Edit1.Text);

c1:=false;

while k1 < StrToInt(Form1.Edit3.Text) do

begin

s1:=s3;

Button11Click(Self);

c1:=true;

End;

end;

procedure TForm1.DrawGraphClick(Sender: TObject); // строим оси координат

begin

DrawGraph.Canvas.MoveTo(0,round(DrawGraph.height/2));

DrawGraph.Canvas.LineTo(DrawGraph.Width,round(DrawGraph.height/2));

DrawGraph.Canvas.MoveTo(round(DrawGraph.Width/2),0);

DrawGraph.Canvas.LineTo(round(DrawGraph.Width/2),DrawGraph.height);

end;

procedure TForm1.Button28Click(Sender: TObject); // кнопка очистить

begin

Button14Click(Self);

DrawGraph.Canvas.REctangle(0,0,DrawGraph.Width,DrawGraph.height)

end;

procedure TForm1.Button30Click(Sender: TObject); // кнопка загрузить

Var

i:integer;

List:tStringList;

begin

If (openDialog1.Execute) then

Begin

List:=tStringList.Create;

List.LoadFromFile(openDialog1.FileName);

For i:=0 to List.Count-1 do

Begin

s1:=List.Strings[i];

A.Text:=List.Strings[i];

end;

List.SaveToFile(openDialog1.FileName);

List.Free;

End;

End;

procedure TForm1.Button31Click(Sender: TObject); //кнопка сохранить

Var

i:integer;

List:tStringList;

begin

i:=1;

If (openDialog1.Execute) then

Begin

List:=tStringList.Create;

List.LoadFromFile(openDialog1.FileName);

List.Strings[i]:=List.Strings[i] + ' = ' + s1;

List.SaveToFile(openDialog1.FileName);

List.Free;

End;

inc(i);

end;

end.