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 курсовому проекту на тему

Программное средство

для реализации вычислений типа «Длинная арифметика»

БГУИР КП 1-40 01 01 408 ПЗ

Выполнил

студент: гр. 151004 Глушаченко Н.С.

Проверил: Фадеева Е.П.

Минск 2022

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

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

(подпись)

Лапицкая Н.В. 2022г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Глушаченко Никите Сергеевичу*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы *Программное средство для реализации вычислений типа «Длинная арифметика» \_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

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

2. Срок сдачи законченной работы *27.05.2022г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Среда программирования Delphi. Возможность ввода данных с клавиатуры или чтением из файла. Вывод результата вычислений на экран и запись выражения в файл. Установка настроек вычислений. В программе доступны следующие возможности: сложение длинных чисел, вычитание длинных чисел, умножение длинных чисел, деление длинных чисел с введенной пользователем точностью. Вычисления выполняются в разных системах счисления. Реализация функций работы с историей вычислений: запись данных после совершения операции, удаление всей истории, запись истории в файл\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Аналитический обзор литературы\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Разработка программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Тестирование и проверка работоспособности программного средства\_\_\_\_*

*5 Руководство по установке и использованию программного средства\_\_\_\_\_\_*

*Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

\_*Приложения* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А1*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе *Фадеева Е.П.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *16.02.2022г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объема работы):

*Раздел 1. Введение к 28.02.2022г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 15.03.2022г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 15.04.2022г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 10.05.2022г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 5.Заключение. Приложения к 20.05.2022г. – 90% готовности работы;*

*оформление пояснительной записки и графического материала к 27.05.2022г. – 100% готовности работы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 27.05.2022г. по 01.06.2022г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Фадеева Е.П.*

*(подпись)*

Задание принял к исполнению *\_\_\_\_\_\_\_\_\_\_\_\_\_\_Глушаченко Н.С. 16.02.2022г.*

*(дата и подпись студента)*

содержание

[Введение 6](#_Toc104494009)

[1 Аналитический обзор литературы 7](#_Toc104494010)

[1.1 Обзор существующих аналогов 7](#_Toc104494011)

[1.1.1 Программное средство Google калькулятор 7](#_Toc104494012)

[1.1.2 Программное средство Калькулятор от Windows 9](#_Toc104494013)

[1.1.3 Программное средство WolframAlpha 13](#_Toc104494014)

[1.2 Анализ методов и способов разработки 15](#_Toc104494015)

[1.2.1 Линейный однонаправленный список 15](#_Toc104494016)

[1.2.2 Файлы 16](#_Toc104494017)

[1.2.3 Очередь 18](#_Toc104494018)

[2 Постановка задачи 19](#_Toc104494019)

[2.1 Назначение разработки 19](#_Toc104494020)

[2.2 Перечень функциональных требований 19](#_Toc104494021)

[2.3 Входные и выходные параметры 20](#_Toc104494022)

[2.4 Состав и параметры технических и программных средств 20](#_Toc104494023)

[3 Разработка программного средства 21](#_Toc104494024)

[3.1 Описание алгоритмов 21](#_Toc104494025)

[3.2 Структура данных 25](#_Toc104494026)

[3.2.1 Структура типов основного алгоритма 25](#_Toc104494027)

[3.2.2 Структура данных алгоритма Input 26](#_Toc104494028)

[3.2.3 Структура данных алгоритма FindResLen 26](#_Toc104494029)

[3.2.4 Структура данных алгоритма Convertion 26](#_Toc104494030)

[3.2.5 Структура данных алгоритма DeleteZero 27](#_Toc104494031)

[3.2.6 Структура данных алгоритма ToExp 27](#_Toc104494032)

[3.2.7 Структура данных алгоритма Summarise 27](#_Toc104494033)

[3.2.8 Струкутра данных алгоритма Subtraction 27](#_Toc104494034)

[3.2.9 Структура данных алгоритма Multiply 28](#_Toc104494035)

[3.2.10 Структура данных алгоритма DivFrac 28](#_Toc104494036)

[3.2.11 Структура данных алгоритма Division 29](#_Toc104494037)

[3.2.12 Структура данных алгоритма Push 29](#_Toc104494038)

[3.2.13 Структура данных алгоритма PopFile 30](#_Toc104494039)

[3.2.14 Структура данных алгоритма ReadFile 30](#_Toc104494040)

[3.2.15 Структура данных алгоритма CheckInput 30](#_Toc104494041)

[3.3 Описание графических компонентов 31](#_Toc104494042)

[3.3.1 Взаимосвязь графического интерфейса 31](#_Toc104494043)

[3.3.2 Описание графических копонентов формы Arithmetics 31](#_Toc104494044)

[3.3.3 Описание графических компонентов формы HistF 32](#_Toc104494045)

[3.3.4 Описание графических компонентов формы HelpF 33](#_Toc104494046)

[3.4 Схема алгоритма решения задачи по ГОСТ 19.701-90 34](#_Toc104494047)

[3.4.1 Схема алгоритма Input 34](#_Toc104494048)

[3.4.2 Схема алгоритма Convertion 36](#_Toc104494049)

[3.4.3 Схема алгоритма ToExp 37](#_Toc104494050)

[3.4.4 Схема алгоритма Summarise 39](#_Toc104494051)

[3.4.5 Схема алгоритма Subtraction 41](#_Toc104494052)

[3.4.6 Схема алгоритма Multiply 45](#_Toc104494053)

[3.4.7 Схема алгоритма DivFrac 48](#_Toc104494054)

[3.4.8 Схема алгоритма Division 50](#_Toc104494055)

[3.4.9 Схема алгоритма Push 54](#_Toc104494056)

[3.4.10 Схема алгоритма PopFile 55](#_Toc104494057)

[4 Тестирование и проверка работоспособности программного средства 56](#_Toc104494058)

[4.1 Проверка ввода 56](#_Toc104494059)

[4.1.1 Тест 1 56](#_Toc104494060)

[4.1.2 Тест 2 56](#_Toc104494061)

[4.1.3 Тест 3 57](#_Toc104494062)

[4.1.4 Тест 4 58](#_Toc104494063)

[4.1.5 Тест 5 58](#_Toc104494064)

[4.2 Проверка вычислений 59](#_Toc104494065)

[4.2.1 Тест 6 59](#_Toc104494066)

[4.2.2 Тест 7 60](#_Toc104494067)

[4.2.3 Тест 8 60](#_Toc104494068)

[4.2.4 Тест 9 61](#_Toc104494069)

[4.2.5 Тест 10 62](#_Toc104494070)

[4.2.6 Тест 11 62](#_Toc104494071)

[4.3 Элементы управления 63](#_Toc104494072)

[4.3.1 Тест 12 63](#_Toc104494073)

[4.3.2 Тест 13 64](#_Toc104494074)

[4.3.3 Тест 14 64](#_Toc104494075)

[4.3.4 Тест 15 65](#_Toc104494076)

[4.3.5 Тест 16 65](#_Toc104494077)

[4.3.6 Тест 17 66](#_Toc104494078)

[4.3.7 Тест 18 66](#_Toc104494079)

[4.3.8 Тест 19 67](#_Toc104494080)

[5 Руководство по установке и использованию программного средства 68](#_Toc104494081)

[5.1 Установка 68](#_Toc104494082)

[5.2 Вычисления 70](#_Toc104494083)

[5.3 Установка настроек и выполнение дополнительных операций 72](#_Toc104494084)

[5.4 Просмотр справки 72](#_Toc104494085)

[5.5 Работа с историей вычислений 73](#_Toc104494086)

[Заключение 74](#_Toc104494087)

[Список использованной литературы 75](#_Toc104494088)

[Приложение А 76](#_Toc104494089)

Введение

В большинстве компьютерных программ и вычислительных средах точность любого вычисления ограничена размером машинного слова компьютера, то есть наибольшим числом, которое может храниться в одном из регистров процессора. По состоянию на середину 2002 г., наиболее распространенный размер машинного слова процессора, составляет 32-бита, что соответствует числу = 4294967296.

В общем случае целочисленная арифметика на 32-битной машине позволяет сложить два 32-битных числа, чтобы получить 33-битное число, умножение 32-битных чисел в результате дает 64-битное число и деление 64-битного числа на 32-битное образует 32-битное частное и 32-битный остаток. [7]

Длинная арифметика в вычислительной технике – операции (сложение, умножение, вычитание, деление, возведение в степень и т.д.) над числами, разрядность которых превышает длину машинного слова данной вычислительной машины. Эти операции реализуются не аппаратно, а программно, используя базовые аппаратные средства работы с числами меньших порядков. [5]

Подсчет больших чисел вручную является довольно кропотливым процессом, занимающим много внимания и усилий, поэтому было принято решении о создании программного средства, которое упрощает данный процесс.

Целью данного проекта является создание такого продукта, с помощью которого можно за короткие промежутки времени выполнять подобные подсчеты.

В реализации проекта попутно решались следующие задачи:

* создание динамической структуры данных;
* работа с файлами (текстовыми, типизированными);
* запись/чтение данных в/из файл(а);
* разработка пользовательского интерфейса для реализации перечисленных функций.

В качестве языка программирования был выбран язык Delphi, а средой разработки – Embarcadero Delphi 10, которые изучаются в рамках предмета «Основы алгоритмизации и программирования».

# Аналитический обзор литературы

## Обзор существующих аналогов

### Программное средство Google калькулятор

Это программное средство целесообразно рассматривать его как один из ведущих аналогов, поскольку оно встроено в поисковую страницу Google и показывается пользователю, когда он вводит соответствующий запрос в поисковую строку.

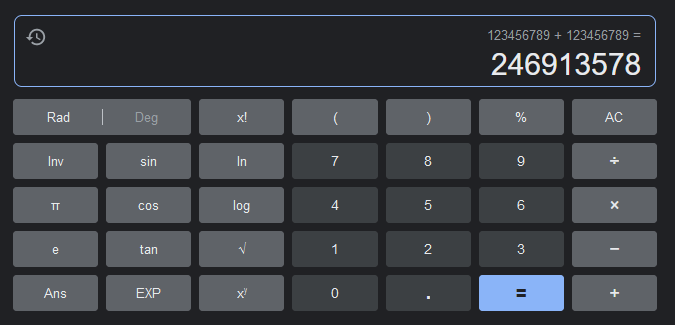


Рисунок 1 – Выполнение операции сложения длинных чисел

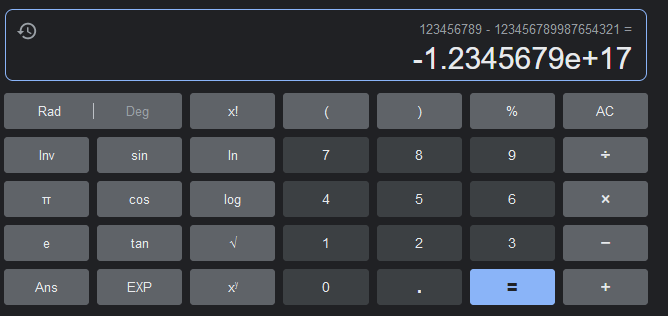


Рисунок 2 – Выполнение операции вычитания длинных чисел

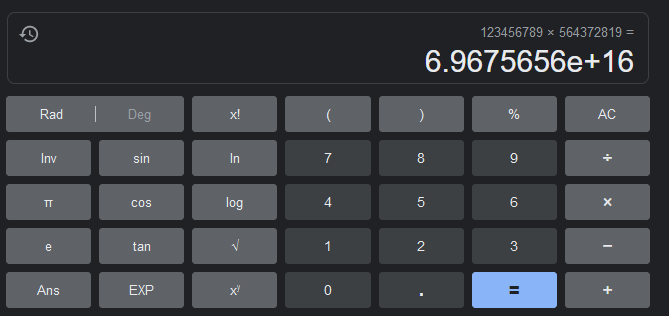


Рисунок 3 – Выполнение операции сложения динных чисел

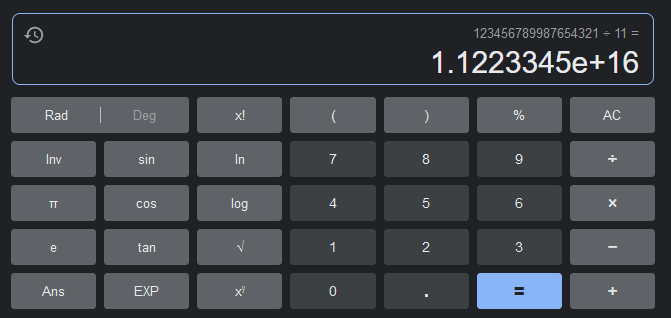


Рисунок 4 – Выполнение операции деления длинных чисел

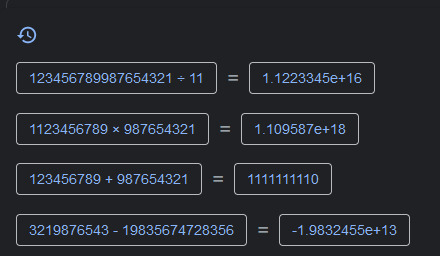


Рисунок 5 – История вычислений

Список выполняемых функций:

* выполнение операции сложения длинных чисел;
* выполнение операции вычитания длинных чисел;
* выполнение операции умножения длинных чисел;
* выполнение операции деления длинных чисел;
* сохрание истории вычислений.

Достоинства:

* выполнение основных арифметических операций над неограниченно большими числами;
* сохранение истории вычислений;
* возможность выполнения дополнительных арифметических операций над длинными числами.

Недостатки:

* необходимость подключения к Интернету;
* результат записывается исключительно в экспоненциальной форме записи;
* отсутствие сохранения истории в отдельный файл;
* отсутствие возможности выбора точности вычислений.

### Программное средство Калькулятор от Windows

Данное программное средство является встроенным в начальный набор операционной системы Windows и доступен по умолчанию. Предоставляет в том числе возможность осуществлять арифметические операции над длинными числами.

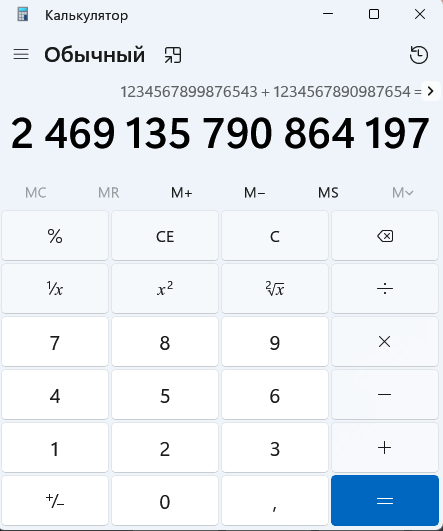


Рисунок 6 – Выполнение операции сложения длинных чисел

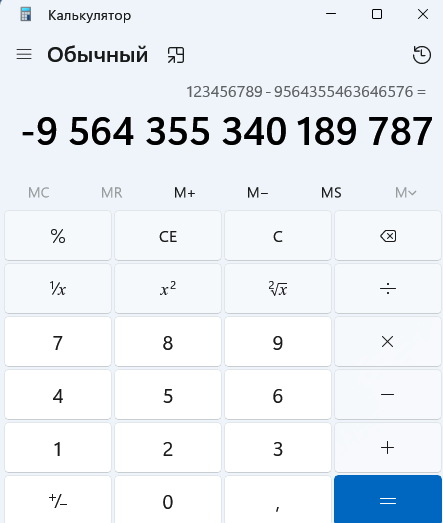


Рисунок 7 – Выполнение операции вычитания длинных чисел

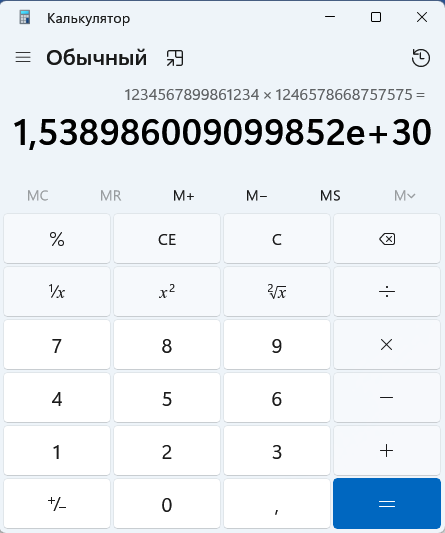


Рисунок 8 – Выполнение операции умножения длинных чисел

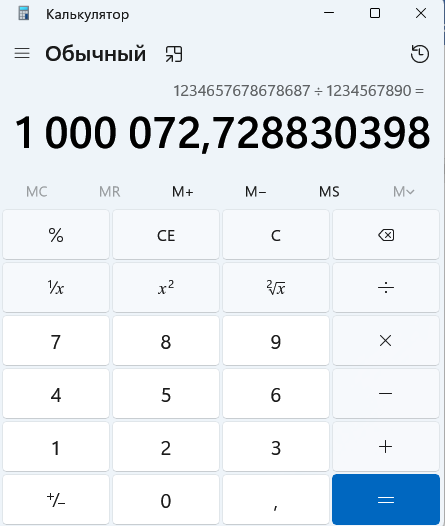


Рисунок 9 – Выполнение операции деления длинных чисел



Рисунок 10 – История вычислений

Список функций:

* выполнение операции сложения длинных чисел;
* выполнение операции вычитания длинных чисел;
* выполнение операции умножения длинных чисел;
* выполнение операции деления длинных чисел;
* сохрание истории вычислений.

Достоинства:

* автономность использования;
* сохранение истории вычислений;
* возможность сохранить историю вычислений в память компьютера;
* возможность выполнения дополнительных арифметических операций над длинными числами;
* наличие возможности очистить историю.

Недостатки:

* размер числа ограничен;
* отсутствие возможности выбора точности вычислений.

### Программное средство WolframAlpha

Данное программное средство представляет собой базу знаний и набор вычислительных алгоритмов в области математики. Позволяет выполнять основные арифметические операции с числами различной длины.

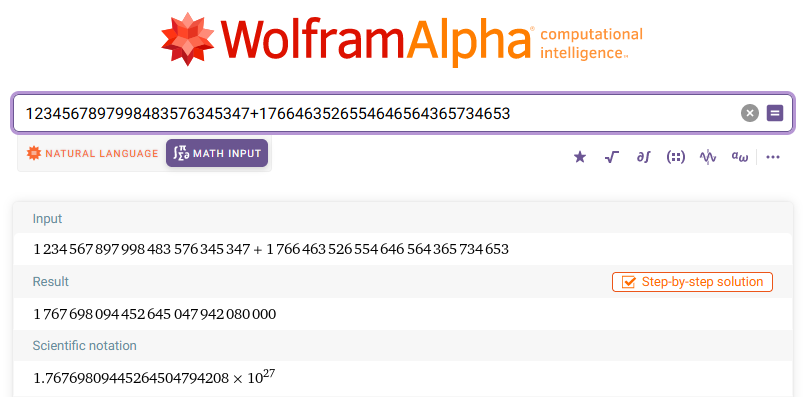


Рисунок 11 – Выполнение операции сложения длинных чисел

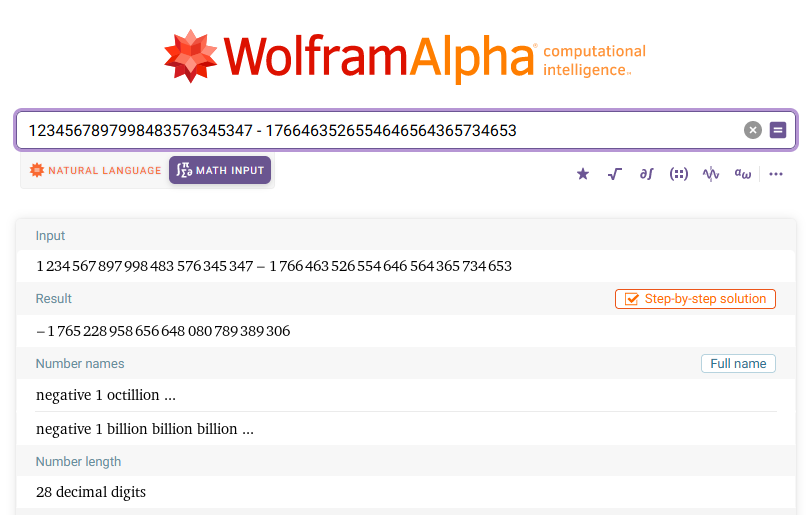


Рисунок 12 – Выполнение операции вычитания длинных чисел

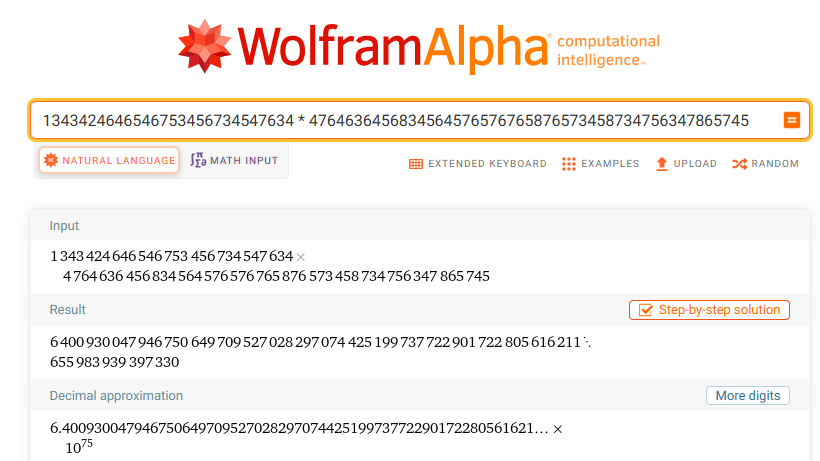


Рисунок 13 – Выполнение операции умножения длинных чисел

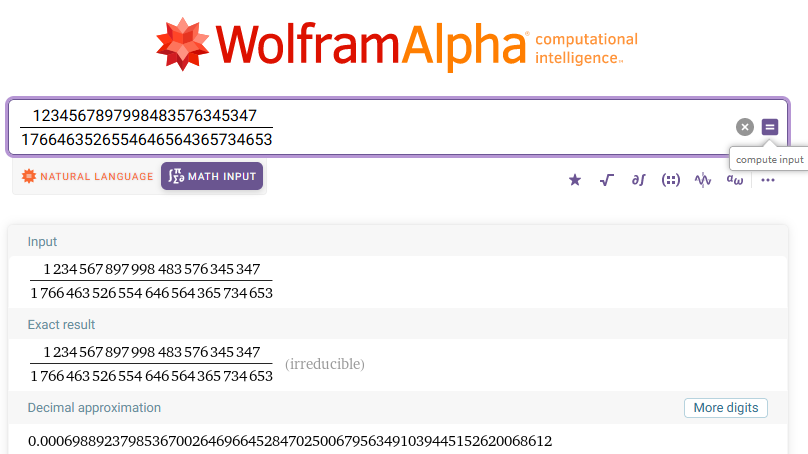


Рисунок 14 – Выполнение операции деления длинных чисел

Список функций:

* выполнение операции сложения длинных чисел;
* выполнение операции вычитания длинных чисел;
* выполнение операции умножения длинных чисел;
* выполнение операции деления длинных чисел.

Достоинства:

* возможность выполнения дополнительных арифметических операций над длинными числами;
* операции выполняются над числами любой длины;
* результат записывается в любой форме.

Недостатки:

* выбор формы записи результата зависит от размера полученного числа;
* необходимость наличия подключения к Интернету;
* отсутствие истории вычислений.

## Анализ методов и способов разработки

Предполагается, что данное программное средство помимо осуществления основных арифметических операций, будет выполнять рад дополнительных функций:

* сохранение истории вычислений в отдельный файл;
* чтение чисел из файла;
* выбор точности вычислений;
* выбор сичтемы счисления;
* вывод истории вычислений на экран.

Для решения данных задач будут использованы следующие структуры данных:

* линейный однонаправленный список;
* файлы;
* очередь.

### Линейный однонаправленный список

Линейный однонаправленный список – динамическая структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей. Каждый элемент списка состоит из поля значения и указателя на следующий элемент. Последний элемент списка указывает на nil. Элемент, на который нет указателя, становится первым (головным) элементом списка. В однонаправленном списке можно передвигаться только от начала к концу. Узнать адрес предыдущего узла исходя из значения текущего невозможно.

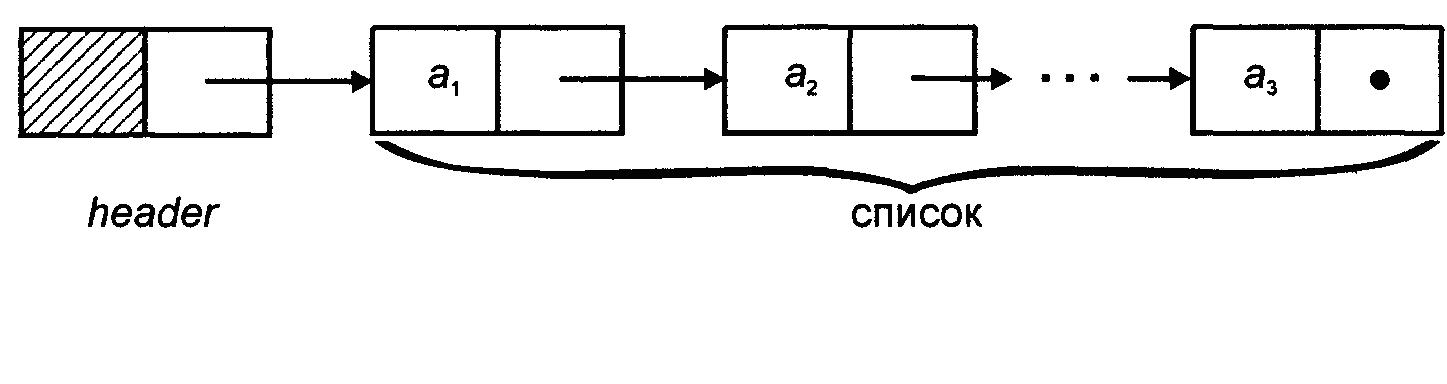


Рисунок 15 – Структура линейного однонаправленного списка

### Файлы

Файловый тип – это произвольная последовательность элементов, длина которой заранее не определена, а конкретезируется в процессе выполнения программы. Это определение логического файла, т.е. того, который используется в программе (файл с точки зрения программиста). Физический файл – это именованная область памяти на внешнем носителе, в которой хранится некоторая информация (файл с точки зрения пользователя).

В Delphi существует 3 типа файлов:

* типизированный файл;
* нетипизированный файл;
* текстовый файл.

Типизированные файлы связываются с файловыми переменными, объявленными как File of <Тип файла>. Файл считается состоящим из элементов соответсвующего типа.

Нетипизированные файлы могут быть только с теми файловыми переменными, которые были объявлены как File. Файл считается состоящим из элементов, размер которых определяется при открытии файла.

Текстовый файл представляет собой последовательность символов, однако он не эквивалентен типизированному файлу File of Char. Текстовые файлы связываются с файловыми переменными, принадлежащими к стандартному типу TextFile. Особенность текстовых файлов состоит в том, что содержащиеся в них элементы разбиваются на строки.

Строки могут иметь различную длину. В конце каждой строки помещаются специальные символы #13 – возврат каретки и #10 – переход на новую строку.

Для доступа к отдельным элементам файлов в Delphi существуют специальные стандартные процедуры и функции.

Процедура AssignFile(F,Name) связывает переменную файлового типа с внешним типом файла на диске.

F – имя файловой переменной

Name – имя файла на диске

Процедура Reset(F) (F – файловая переменная) открывает существующий файл F в режиме чтения. Фактически при этом открывается внешний файл с именем, присвоенным переменной F процедурой AssignFile. Если файл с данным именем не существует, возникает исключение. Указатель файла устанавливается на первую позицию файла. Если файл был предварительно открыт, то он закрывается и повторно открывается. Процедура может быть применена к файлу неограниченное количество раз. При выполнении данной процедуры содержимое файла не изменяется. Для текстовый файлов – в режиме только для чтения (read-only).

Процедура Rewrite(F) создает новый файл F и открывает его. Если файл уже открыт, то закрывает и открывает его снова. Указатель файла устанавливается в начало файла (файл создается пустым). Для текстовых файлов – в режиме только для записи (write-only).

Процедура Append(F) определена только для текстовых файлов. Она открывает существующий файл F для добавления. Если файл F уже открыт, то открывает его и открывает заново. Если файл с данным именем не существует, возникает исключение. В данном случае указатель файла устанавливается на конец файла. Файл открывается в режиме только для записи (write-only).

Процедура CloseFile(F) закрывает файл. F – файловая переменная.

Функция EoF(F) проверяет, является ли текущая позиция концом файла F. Возвращает переменную логического типа. True – если достигнут конец файла. False – если конец файла не достигнут.

Функция EoLn(F) проверяет, является ли текущая позиция концом строки файла F. Возвращает результат логического типа. True – если текущая позиция является концом строки. False – если текущая позиция не является концом строки.

Для типизированных и тестовых файлов определены процедуры считывания компонентов файла в переменные и записи в файл компонентов из переменных read(F,A) и write(F,A). Процедура может иметь любое количество параметров после параметра файлового типа.

F – файловая переменная.

A – переменная, соответствующая типу файла.

Для нетипизированных файлов также сузествуют процедуры для записи и чтения.

Этими процедурами являются BlockRead(F,Buf,Count,AmtTransferred) для чтения и BlockWrite(F,Buf,Count,AmtTransferred) для записи.

F – переменная файлового типа

Buf – переменная для записи/чтения значения в/из файл(а)

Count – число записей, которые необходимо записать/прочитать

AmtTranferred – число записанных/прочитанных записей

Максимальный размер прочитанного блока равен Count\*RecSize байт. RecSize - размер записи, определенный, во время открытия файла (если размер записи не был задан, то используется значение по умолчанию - 128 байт).  
Параметр AmtTransferred является необязательным. Но если данный параметр опущен, а количество прочитанных записей меньше Count, то возникнет ошибка ввода/вывода.

### Очередь

Очередь – это специальный тип списка. Очередью называют структуру, из которой элементы удаляются с одного ее конца, называемого началом (головой), а вставляются на противоположном конце, называемом (хвостом).  
Очереди считаются списками типа FIFO (аббревиатура расшифровывается  
как first in first out: первым вошел – первым вышел). Две основные операции,  
которые определены для работы с очередью: вставка и извлечение элементов.

Очередь будет использована для реализации последовательной записи результатов вычислений и дополнительных параметров в текстовый файл.

Очередь будет реализована на базе линейного однонаправленного списка.

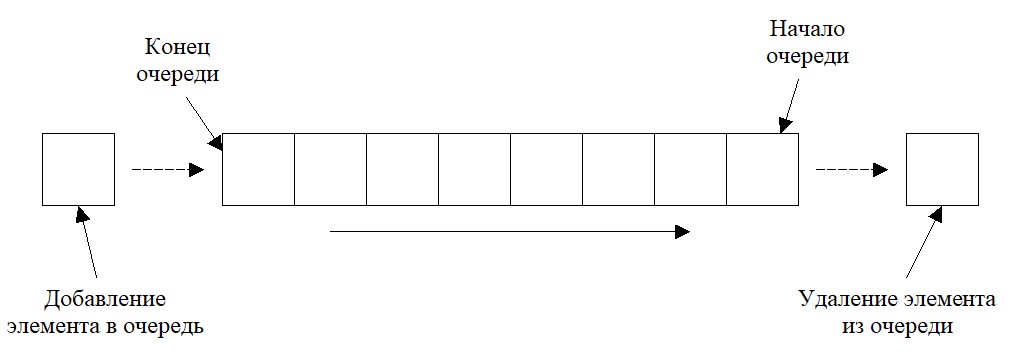


Рисунок 16 – Структура очереди

# Постановка задачи

## Назначение разработки

Длинная арифметика — выполняемые с помощью вычислительной машины арифметические операции (сложение, умножение, вычитание, деление) над числами, разрядность которых превышает длину машинного слова данной вычислительной машины. Эти операции реализуются не аппаратно, а программно, используя базовые аппаратные средства работы с числами меньших порядков.

Длинная арифметика применяется в следующих областях:

* составление кода для процессоров низкой разрядности;
* криптография;
* математическое и финансовое ПО.

На данный момент большинство популярных языков программирования (Java, Python и др.) имеют встроенную поддержку длинной арифметики, что в разы может сократить время написания программы.

## Перечень функциональных требований

Функции, выполняемые программным средством:

* сложение длинных чисел;
* вычитание длинных чисел;
* умножение длинных чисел;
* деление длинных чисел;
* вычисления могут выполняться с разной точностью для деления;
* выбор системы счисления;
* ввод данных с клавиатуры;
* ввод данных посредством чтения из файла;
* вывод результатов вычислений на экран;
* вывод результатов вычислений в файл;
* заполнение истории вычислений;
* удаление истории вычислений.

При реализации данного программного средства будут использованы типизированные файлы и динамическая структура данных типа очередь, рализованная на базе линейного однонаправленного списка.

## Входные и выходные параметры

В качестве входных данных может использоваться:

* введенное с клавиатуры значение;
* выбранный пользователем файл.

Выходными данными может быть:

* сообщение, выведенное на экран;
* файл, содержащий историю вычислений.

## Состав и параметры технических и программных средств

Программное средство для реализации вычислений типа «Длинная арифметика»должно функционировать на персональных компьютерах со следующими характеристиками:

* процессор Intel Core i5 2.4 GHz или лучше;
* оперативная память 4 GB 1600 MHz DDR3 или лучше;
* накопитель HDD или SSD объемом 128 GB или больше.

Программное средство для реализации вычислений типа «Длинная арифметика» должно функционировать в окружении операционной системы Windows 11.

# Разработка программного средства

## Описание алгоритмов

Таблица 1 – Описание основных алгоритмов программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Назначение алгоритма | Формальные параметры | Предлагаемый тип реализации |
| 1 | Инициализация главной формы | Выбор функции, выполняемой над числами, ввод данных и вывод результата, создание типизированного файла, содержащего историю вычислений, просмотр истории вычислений.  Вызываются подпрограммы:  Input(Str,Arr,  Notation,Flag)  DeleteZero(A)  ToExp(S,P)  Summarise(A,B,C,  Notation)  Subtraction(A,B,C,  Notation,Sign)  Multiply(A,B,C,  Notation)  Division(A,B,C,D,  Notation,P)  Push(X,Y,R)  PopFile(X)  ReadFile(Name)  CheckInput(Param) |  |  |

Продолжение Таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | Input  (Str,Arr,  Notation,  Flag) | Перевод числа Str в массив Arr. Проверка числа Str на соответствие системе счисления Notation. Результат - Flag | Str, Arr, Flag – получает от фактического параметра адрес.  Notation – получает от фактического параметра адрес с защитой | Процедура |
| 3 | FindResLen  (Str1,Str2,Res) | Сравнение длин строк Str1 и Str2. Res – наибольшая длина | Str1, Str2 – получает от фактического параметра адрес с защитой.  Res – получает от фактического параметра адрес | Функция. Res – возвращаемый параметр |
| 4 | Convertion  (A,Res) | Перевод значения из массива A в строку Res | A – получает от фактического параметра адрес с защитой.  Res – получает от фактического параметра адрес | Функция. Res – возвращаемый параметр |
| 5 | DeleteZero  (A) | Удаление нулей из начала числа A | A – получает от фактического параметра адрес | Процедура |
| 6 | ToExp  (S,P) | Перевод числа S с длиной дробной части P в экспоненциальную форму записи | S – получает от фактического параметра адрес.  P – получает от фактического параметра адрес с защитой | Процедура |
| 7 | Summarise  (A,B,C,  Notation) | Сложение параметров A и B. Результат записывается в параметр C.  Notation – система счисления. | A,B,Notation – получает от фактического параметра адрес с защитой.  C – получает от фактического параметра адрес | Процедура |

Продолжение Таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | Subtraction  (A,B,C,  Notation,Sign) | Вычисление разницы между параметрами A и B. Результат записывается в C. Notation – система счисления. Sign – знак результата | A,B,C,Sign – получает от фактического параметра адрес.  Notation – получает от фактического параметра адрес с защитой | Процедура |
| 9 | Multiply  (A,B,C,  Notation) | Выполнение операции сложения параметров A и B. Результат записывается в C. Notation – система счисления | A,B,Notation – получает от фактического параметра адрес с защитой.  C – получает от фактического параметра адрес | Процедура |
| 10 | DivFrac  (A,B,C,  Notation,Acc) | Вычисление дробной части от деления A и B с точностью Acc. C – результат вычислений. Notation – система счисления | A,B,C – получает от фактического параметра адрес.  Notation,Acc – получает от фактического параметра адрес с защитой | Процедура |
| 11 | Division  (A,B,C,D,  Notation,P) | Выполнение операции деления между числами A и B. C – целая часть результата. D – дробная часть результата. Notation – система счисления. P – число знаков после запятой | A,B,C,D – получает от фактического параметра адрес.  Notation,P – получает от фактического параметра адрес с защитой | Процедура |

Продолжение Таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 12 | Push  (X,Y,R) | Добавление элемента в конец очереди. X – указатель на начало очереди. Y – указатель на конец очереди.  R - добавляемый в очередь параметр | X,Y – получает от фактического параметра адрес.  R – получает от фактического параметра адрес с защитой | Процедура |
| 13 | PopFile  (X) | Запись данных из очереди в файл. X – указатель на текущий элемент очереди. | X – получает от фактического параметра значение | Процедура |
| 14 | ReadFile  (R) | Запись информации из файла в параметр R. | R – получает от фактического параметра адрес | Процедура |
| 15 | CheckInput  (Param) | Проверка числа Param на соответствие условию | Param – получает от фактического параметра адрес | Процедура |

## Структура данных

### Структура типов основного алгоритма

При разработке программного средства была использована динамическая структура данных линейный однонаправленный список.

Таблица 2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TS | String[255] | Тип полей записи, хранящих результат и исходные числа |
| TRec | Record  S1: TS;  S2: TS;  SRes: TS;  Operation: Char;  Eq: Char;  Position: Integer;  Precession: Integer;  L1: Integer;  L2: Integer;  LRes: Integer;  Ntn: Integer;  end; | Запись, содержащая информацию об арифметической операции. Используется для хранения истории вычислений.  S1 – первое число.  S2 – второе число.  SRes – результат.  Operation – знак результата.  Eq – знак равно.  Position – номер выражения.  Precession – точность вычислений для деления.  L1 – длина первого числа.  L2 – длина второго числа.  LRes – длина результата.  Ntn – система счисления |
| Pt | ^Elem | Линейный однонаправленный список, состоящий из записей типа TRec.  Data – запись, содержащая информацию об арифметической операции.  Next – указатель на следующий элемент списка |
| Elem | Record  Data: TRec;  Next: Pt;  end; |
| TFile | File of TRec | Файл, содержащий историю вычислений |
| TArray | Array of Integer | Динамический массив. Используется для выполнения арифметических операций |

### Структура данных алгоритма Input

Таблица 3 – Структура данных алгоритма Input(Str,Arr,Notation,Flag)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Str | String | Исходная строка | Формальный |
| Arr | TArray | Массив, заполняемый элементами исходной строки | Формальный |
| Notation | Integer | Система счисления | Формальный |
| Flag | Boolean | Проверка корректности ввода | Формальный |
| Error | Integer | Проверка числа на соответствие системе счисления | Локальный |
| I | Integer | Счетчик цикла | Локальный |

### Структура данных алгоритма FindResLen

Таблица 4 – Структура данных алгоритма FindResLen(Str1,Str2,Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Str1 | String | Первое сравниваемое число | Формальный |
| Str2 | String | Второе сравниваемое число | Формальный |
| Res | Integer | Наибольшая из длин чисел | Формальный |

### Структура данных алгоритма Convertion

Таблица 5 – Структура данных алгоритма Convertion(A,Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| A | TArray | Массив, элементы которого переносятся в строку | Формальный |
| Res | String | Строка, содержащая элементы массива | Формальный |
| I | Integer | Счетчик цикла | Локальный |

### Структура данных алгоритма DeleteZero

Таблица 6 – Структура данных алгоритма DeleteZero(A,I)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| A | TArray | Массив, из конца которого удаляются нулевые элементы | Формальный |
| I | Integer | Счетчик цикла | Локальный |

### Структура данных алгоритма ToExp

Таблица 7 – Структура данных алгоритма ToExp(S,P)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| S | String | Исходная строка | Формальный |
| P | Integer | Число знаков после точки | Формальный |
| Tmp | String[245] | Сохраняемая часть строки | Локальный |
| I | Integer | Удаление нулей из начала числа | Локальный |

### Структура данных алгоритма Summarise

Таблица 8 – Структура данных алгоритма Summarise(A,B,C,Notation)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| A | TArray | Первое число | Формальный |
| B | TArray | Второе число | Формальный |
| C | TArray | Сумма чисел | Формальный |
| Notation | Integer | Система счисления | Формальный |
| I,J | Integer | Счетчик цикла | Локальный |
| Temp | Integer | Перенос остатка от деления на систему счисления в следующий разряд | Локальный |

### Струкутра данных алгоритма Subtraction

Таблица 9 – Структура данных алгоритма Subtraction(A,B,C,Notation,Sign)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| A | TArray | Первое число | Формальный |
| B | TArray | Второе число | Формальный |
| C | TArray | Разность чисел | Формальный |

Продолжение Таблицы 9

|  |  |  |  |
| --- | --- | --- | --- |
| Notation | Integer | Система счисления | Формальный |
| Sign | Boolean | Знак результата | Формальный |
| I,J | Integer | Счетчик цикла | Локальный |
| Temp | Integer | Заимствование из старшего разряда | Локальный |
| Supp | TArray | Вспомогательный массив для замены чисел местами | Локальный |
| StrA,StrB | String | Строки, содержащие исходные числа | Локальный |

### Структура данных алгоритма Multiply

Таблица 10 – Структура данных алгоритма Multiply(A,B,C,Notation)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| A | TArray | Первое число | Формальный |
| B | TArray | Второе число | Формальный |
| C | TArray | Произведение чисел | Формальный |
| Notation | Integer | Система счисления | Формальный |
| I,J | Integer | Счетчик цикла | Локальный |
| Temp | Integer | Перенос остатака от деления на систему счисления в старший разряд | Локальный |

### Структура данных алгоритма DivFrac

Таблица 11 – Структура данных алгоритма DivFrac(A,B,C,Notation,Acc)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| A | TArray | Первое число | Формальный |
| B | TArray | Второе число | Формальный |
| C | TArray | Дробная часть числителя | Формальный |
| Notation | Integer | Система счисления | Формальный |
| Acc | Integer | Размер дробной части результата | Формальный |
| I | Integer | Результат деления | Локальный |
| K | Integer | Счетчик цикла |  |
| AC | TArray | Копия числителя | Локальный |
| Ten | TArray | Перенос числителя на разряд выше | Локальный |

Продолжение Таблицы 11

|  |  |  |  |
| --- | --- | --- | --- |
| Save | TArray | Копия знаменателя | Локальный |
| Flag | Boolean | Вычисление текущей цифры | Локальный |

### Структура данных алгоритма Division

Таблица 12 – Структура данных алгоритма Division(A,B,C,D,Notation,P)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| A | TArray | Первое число | Формальный |
| B | TArray | Второе число | Формальный |
| C | TArray | Целая часть частного | Формальный |
| D | TArray | Дробная часть частного |  |
| Notation | Integer | Система счисления | Формальный |
| P | Integer | Размер дробной части результата | Формальный |
| I,J | Integer | Счетчик цикла | Локальный |
| Shift | Integer | Разница длин первого и второго чисел | Локальный |
| Mul | TArray | Результат умножения цифры результата на знаменатель | Локальный |
| AC | TArray | Копия числителя | Локальный |
| Sub | TArray | Результат вычитания Mul из AC | Локальный |
| Digit | TArray | Цифра результата в соответсвующем разряде | Локальный |
| S1,S2 | String | Копии исходных числел | Локальный |
| Flag | Boolean | Вычисление текущей цифры | Локальный |

### Структура данных алгоритма Push

Таблица 13 – Структура данных алгоритма Push(X,Y,R)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| X | Pt | Указатель на начало списка | Формальный |
| Y | Pt | Указатель на конец списка | Формальный |

Продолжение Таблицы 13

|  |  |  |  |
| --- | --- | --- | --- |
| R | TRec | Данные, добавляемые в конец списка | Формальный |
| PTmp | Pt | Вспомогательный указатель | Локальный |

### Структура данных алгоритма PopFile

Таблица 14 – Структура данных алгоритма PopFile(X)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| X | Pt | Указатель на первый элемент списка | Формальный |
| F | TFile | Файл, в который записываются элементы списка | Локальный |

### Структура данных алгоритма ReadFile

Таблица 15 – Структура данных алгоритма ReadFile(R)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| R | TRec | Запись, в которую заносятся данные | Формальный |
| F | TFile | Файл, из которого записываются элементы списка | Локальный |

### Структура данных алгоритма CheckInput

Таблица 16 – Структура данных алгоритма CheckInput(Param)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Param | Integer | Введенное пользователем исходное значение | Формальный |
| MinNtn | Integer | Нижняя граница диапазона допустимых значений | Локальный |
| MaxNtn | Integer | Верхняя граница диапазона допустимых значений | Локальный |

## Описание графических компонентов

### Взаимосвязь графического интерфейса

Для организации графического интерфейса программного средства для реализации вычислений типа «Длинная арифметика» были использованы 3 формы: Arithmetics, HistF, HelpF



Рисунок 17 – Взаимосвязь графического интерфейса

### Описание графических копонентов формы Arithmetics

Форма главного меню программы Arithmetics дает доступ к основным функциям программного средства, таким как:

* выполнение операций над числами в разных системах счисления;
* сохранение результата;
* обращение к истории;
* обращение к справке.

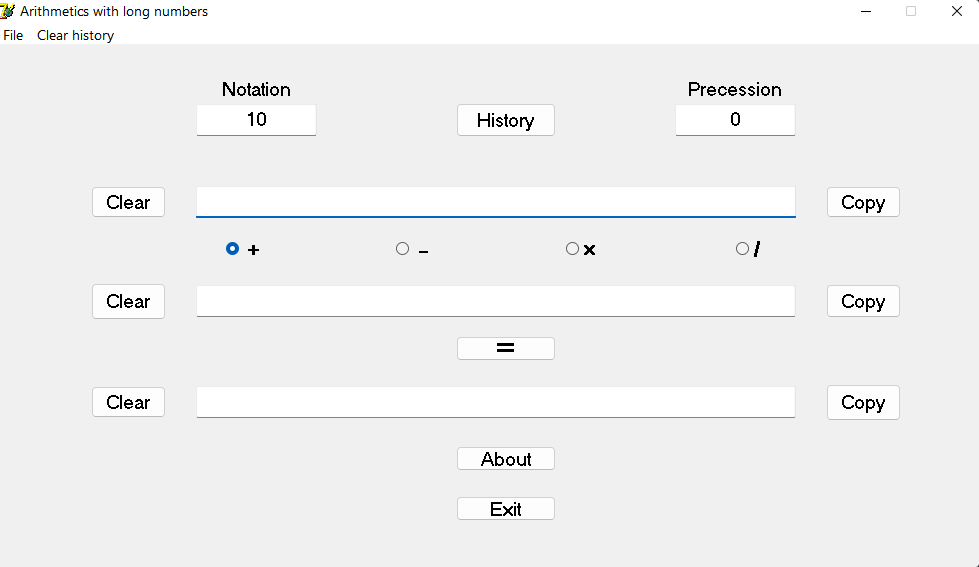


Рисунок 18 – вид формы Arithmetics

Составляющие главной формы Arithmetics:

* поле для ввода первого числа;
* поле для ввода второго числа;
* кнопка «History», предоставляет доступ к форме HistF;
* круглые кнопки выбора знака операции;
* поле выбора системы счисления;
* поле выбора точности для деления;
* кнопки «Copy», позволяют скопировать данные из соответствующего поля в буфер обмена;
* кнопки «Clear», позволяют очистить соответствующие поля;
* кнопка «=», выводит результат операции;
* вкладка меню «File», предоставляет доступ к сохранению истории в отдельный файл и чтению данных из текстового файла;
* вкладка меню «Clear history», позволяет полностью очистить историю вычислений;
* вкладка меню «About», предоставляет доступ к форме HelpF для получения информации о программном средстве.

### Описание графических компонентов формы HistF

Вспомогательная форма программы HistF предоставляет возможность просмотра истории вычислений

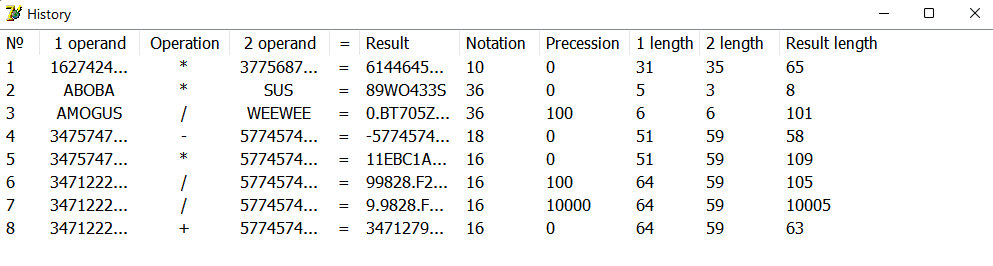


Рисунок 19 – вид формы HistF

Составляющие вспомогательной формы ComputationHistory:

* компонент TListView, предназначенный для просмотра истории.

### Описание графических компонентов формы HelpF

Вспомогательная форма программы HelpForm предоставляет пользователю возможность просмотра информации о программном средстве и о правилах работы с ним

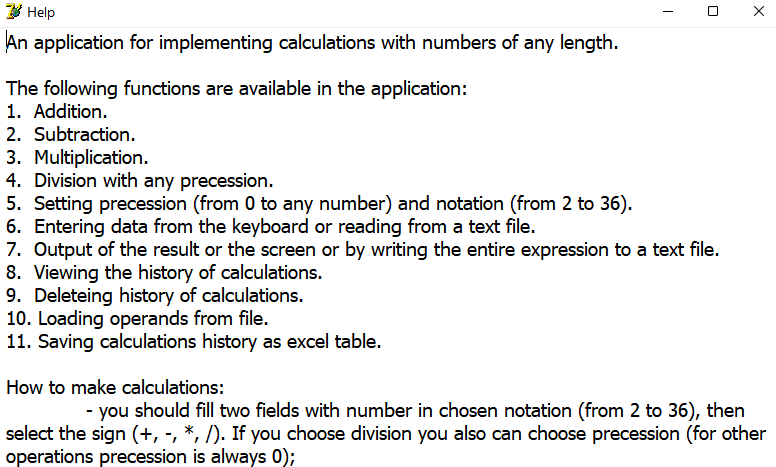


Рисунок 20 – вид формы HelpF

Составляющие вспомогательной формы HelpForm:

* компонент TMemo для отображения справки.

## Схема алгоритма решения задачи по ГОСТ 19.701-90

### Схема алгоритма Input

Схема алгоритма Input отражает процесс ввода значений пользователем, их проверку на соответствие системе счисления и занесение цифр исходного числа в массивы. Данная схема представлена на рисунке 21

Этапы алгоритма Input:

1. Схема начинается с блока вход. На данном этапе программа уже запущена.
2. Далее следует блок, в котором инициализируется переменная Flag.
3. Затем следует блок, в котором устанавливается длина массива, в который будут занесены цифры исходного числа.
4. В случае, если система счисления больше либо равна 10, осуществляется переход к блоку 12.
5. Инициализируется переменная I
6. Начало цикла А2 по переменной I. I изменяет свои значения от 1 до длины исходной строки.



Рисунок 21 – Схема алгоритма Input по ГОСТ 19.701-90(часть 1)



Рисунок 22 – Схема алгоритма Input по ГОСТ 19.701-90(часть 2)

### Схема алгоритма Convertion



Рисунок 23 – Схема алгоритма Convertion по ГОСТ 19.701-90

### Схема алгоритма ToExp



Рисунок 24 – схема алгоритма ToExp по ГОСТ 19.701-90(часть 1)



Рисунок 25 – схема алгоритма ToExp по ГОСТ 19.701-90(часть 2)

### Схема алгоритма Summarise



Рисунок 26 – Схема алгоритма Summarise по ГОСТ 19.701-90(часть 1)



Рисунок 27 – Схема алгоритма Summarise по ГОСТ 19.701-90(часть 2)

### Схема алгоритма Subtraction



Рисунок 28 – Схема алгоритма Subtraction по ГОСТ 19.701-90(часть 1)



Рисунок 29 – Схема алгоритма Subtraction по ГОСТ 19.701-90(часть 2)



Рисунок 30 – Схема алгоритма Subtraction по ГОСТ 19.701-90(часть 3)



Рисунок 31 – Схема алгоритма Subtraction по ГОСТ 19.701-90(часть 4)

### Схема алгоритма Multiply



Рисунок 32 – Схема алгоритма Multiply по ГОСТ 19.701-90(часть 1)



Рисунок 33 – Схема алгоритма Multiply по ГОСТ 19.701-90(часть 2)



Рисунок 34 – Схема алгоритма Multiply по ГОСТ 19.701-90(часть 3)

### Схема алгоритма DivFrac



Рисунок 35 – Схема алгоритма DivFrac по ГОСТ 19.701-90(часть 1)



Рисунок 36 – Схема алгоритма DivFrac по ГОСТ 19.701-90(часть 2)

### Схема алгоритма Division



Рисунок 37 – Схема алгоритма Division по ГОСТ 19.701-90(часть 1)



Рисунок 38 – Схема алгоритма Division по ГОСТ 19.701-90(часть 2)



Рисунок 39 – Схема алгоритма Division по ГОСТ 19.701-90(часть 3)



Рисунок 40 – Схема алгоритма Division по ГОСТ 19.701-90(часть 4)

### Схема алгоритма Push



Рисунок 41 – Схема алгоритма Push по ГОСТ 19.701-90

### Схема алгоритма PopFile



Рисунок 42 – Схема алгоритма PopFile по ГОСТ 19.701-90

# Тестирование и проверка работоспособности программного средства

В ходе тестирования были рассмотрены в действии все основные возможности программного средства для реализации вычислений типа «Длинная арифметика».

## Проверка ввода

### Тест 1

Таблица 17 – Тест 1

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности ввода в выбранной системе счисления |
| Исходные данные | Система счисления: 10  Первый операнд: AZ123  Второй операнд: 123 |
| Ожидаемый результат | Вывод сообщения о некорректности ввода первого операнда. Выделение первого операнда |
| Полученный результат |  |

### Тест 2

Таблица 18 – Тест 2

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности ввода в выбранной системе счисления |
| Исходные данные | Система счисления: 10  Первый операнд: 123  Второй операнд: AZ123 |
| Ожидаемый результат | Вывод сообщения о некорректности ввода первого операнда. Выделение первого операнда |

Продолжение Таблицы 18

|  |  |
| --- | --- |
| Полученный результат |  |

### Тест 3

Таблица 19 – Тест 3

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности ввода в выбранной системе счисления |
| Исходные данные | Система счисления: 35  Первый операнд: 123  Второй операнд: AZ123 |
| Ожидаемый результат | Вывод сообщения о некорректности ввода первого операнда. Выделение первого операнда |
| Полученный результат |  |

### Тест 4

Таблица 20 – Тест 4

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности ввода в выбранной системе счисления |
| Исходные данные | Система счисления: 54  Первый операнд: 123  Второй операнд: 42 |
| Ожидаемый результат | Вывод результата в 10-чной системе счисления. Система счисления: 10 |
| Полученный результат |  |

### Тест 5

Таблица 21 – Тест 5

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности ввода в выбранной системе счисления |
| Исходные данные | Система счисления: 1  Первый операнд: 123  Второй операнд: 42 |
| Ожидаемый результат | Вывод результата в 10-чной системе счисления. Система счисления: 10 |

Продолжение Таблицы 21

|  |  |
| --- | --- |
| Полученный результат |  |

## Проверка вычислений

### Тест 6

Таблица 22 – Тест 6

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности вычислений при выполнении арифметических операций в разных системах счисления |
| Исходные данные | Система счисления: 10  Операция: +  Первый операнд: 1234567890442345768767656223345  Второй операнд: 35445545657565445655654565685 |
| Ожидаемый результат | Результат: 1270013436099911214423310789030 |
| Полученный результат |  |

### Тест 7

Таблица 23 – Тест 7

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности вычислений при выполнении арифметических операций в разных системах счисления |
| Исходные данные | Система счисления: 20  Операция: \*  Первый операнд: 1234AF567890CF4234576876765ABC  Второй операнд: 3AAA655654565685 |
| Ожидаемый результат | Результат: 3I2II9HDDJ9A10H1I922721D8C7HCHG2GI3ECF5291HE0 |
| Полученный результат |  |

### Тест 8

Таблица 24 – Тест 8

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности вычислений при выполнении арифметических операций в разных системах счисления |
| Исходные данные | Система счисления: 25  Операция: -  Первый операнд: 1234AF567890CF4234576876765ABC  Второй операнд: 3AAA655654565685 |
| Ожидаемый результат | Результат: 1234AF567890CF0GHIO21222200437 |

Продолжение Таблицы 24

|  |  |
| --- | --- |
| Полученный результат |  |

### Тест 9

Таблица 25 – Тест 9

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности вычислений при выполнении арифметических операций в разных системах счисления |
| Исходные данные | Система счисления: 25  Операция: -  Первый операнд: 3AAA655654565685  Второй операнд: 1234AF567890CF4234576876765ABC |
| Ожидаемый результат | Результат: -1234AF567890CF0GHIO21222200437 |
| Полученный результат |  |

### Тест 10

Таблица 26 – Тест 10

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности вычислений при выполнении арифметических операций в разных системах счисления |
| Исходные данные | Система счисления: 30  Операция: /  Точность: 0  Первый операнд: 1234AF567890CF4234576876765ABC  Второй операнд: 3AAA655654565685 |
| Ожидаемый результат | Результат: 9HSFRA0PSP054I |
| Полученный результат |  |

### Тест 11

Таблица 27 – Тест 11

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности вычислений при выполнении арифметических операций в разных системах счисления |
| Исходные данные | Система счисления: 30  Операция: /  Точность: 15  Первый операнд: 1234AF567890CF4234576876765ABC  Второй операнд: 3AAA655654565685 |
| Ожидаемый результат | Результат: 9HSFRA0PSP054I.CRS1TT42LM3FR7L |

Продолжение Таблицы 27

|  |  |
| --- | --- |
| Полученный результат |  |

## Элементы управления

### Тест 12

Таблица 28 – Тест 12

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности работы элементов управления |
| Исходные данные | Нажатие на кнопку «Delete history» |
| Ожидаемый результат | Удаление истории вычислений |
| Полученный результат |  |

### Тест 13

Таблица 29 – Тест 13

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности работы элементов управления |
| Исходные данные | Нажатие на кнопку «Save history as excel table» |
| Ожидаемый результат | Вывод на экран истории в виде Excel таблицы |
| Полученный результат |  |

### Тест 14

Таблица 30 – Тест 14

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности работы элементов управления |
| Исходные данные | Нажатие на кнопку «Load 1 operand…»  Число в файле: 153 |
| Ожидаемый результат | Поле первого операнда заполнено числом из файла |
| Полученный результат |  |

### Тест 15

Таблица 31 – Тест 15

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности работы элементов управления |
| Исходные данные | Нажатие на кнопку «Load 2 operand…»  Число в файле: 131313115463246732457236523647632542323  82582462356235462534263547623548654814762354762354765  72547657457265427645273654265427365423645236427354265  47624762347625476254762354732654763254726354726542765 42364523765423765427365432645673457 |
| Ожидаемый результат | Поле второго операнда заполнено числом из файла |
| Полученный результат |  |

### Тест 16

Таблица 32 – Тест 16

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности работы элементов управления |
| Исходные данные | Нажатие на кнопку «About» |
| Ожидаемый результат | Вывод на экран руководства пользователя |

Продолжение Таблицы 32

|  |  |
| --- | --- |
| Полученный результат |  |

### Тест 17

Таблица 33 – Тест 17

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности работы элементов управления |
| Исходные данные | Нажатие на кнопку «Exit» |
| Ожидаемый результат | Закрытие программы |
| Полученный результат | Закрытие программы |

### Тест 18

Таблица 34 – Тест 18

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности работы элементов управления |
| Исходные данные | Нажатие на кнопку «Clear»  Первый операнд: 1234AF567890CF4234576876765ABC |
| Ожидаемый результат | Удаление исходного операнда из поля |
| Полученный результат |  |

### Тест 19

Таблица 35 – Тест 19

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка корректности работы элементов управления |
| Исходные данные | Нажатие на кнопку «Copy»  Первый операнд: 1234AF567890CF4234576876765ABC |
| Ожидаемый результат | Операнд скопирован в буфер обмена |
| Полученный результат | Операнд скопирован в буфер обмена |

# Руководство по установке и использованию программного средства

## Установка

Для установки программы необходимо запустить «Setup.exe». После запуска появляется стартовое окно с выбором языка, в котором необходимо выбрать язык установки и нажать кнопку «ОК»

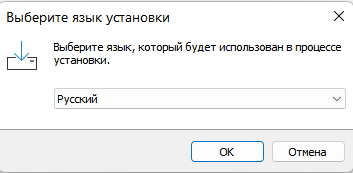


Рисунок 43 – Окно выбора языка

Появляется окно со списком функции, выполняемых программой и лицензионным соглашением.

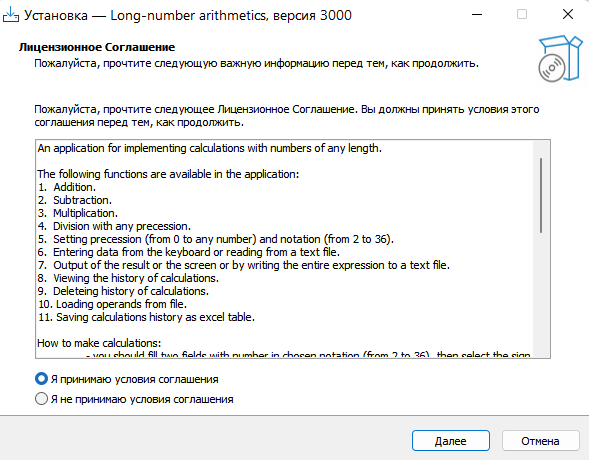


Рисунок 44 – Лицензионное соглашение

Появляется окно, которое позволяет выбрать папку, в которой создастся программа

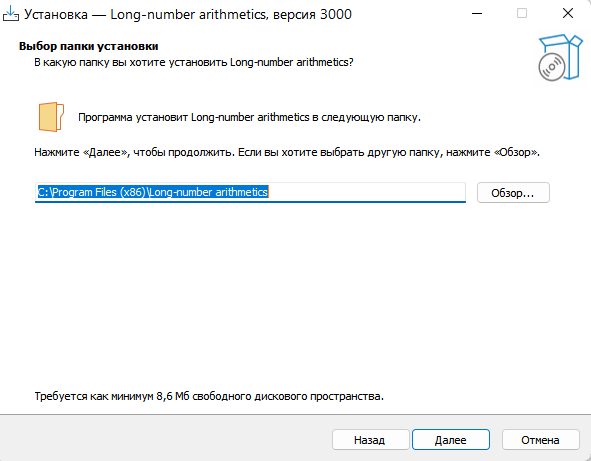


Рисунок 45 – Выбор папки установки

Появляется окно, которое посволяет пользователю выбрать, создавать ли значок программы на рабочем столе

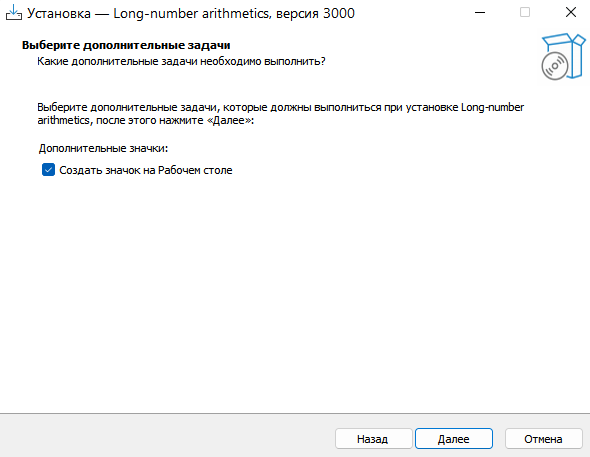


Рисунок – Дополнительные настройки

Появлется финальное окно, которое содержит информацию о выбранных пользователем пунктах. Для завершения установки нажмите «Установить»

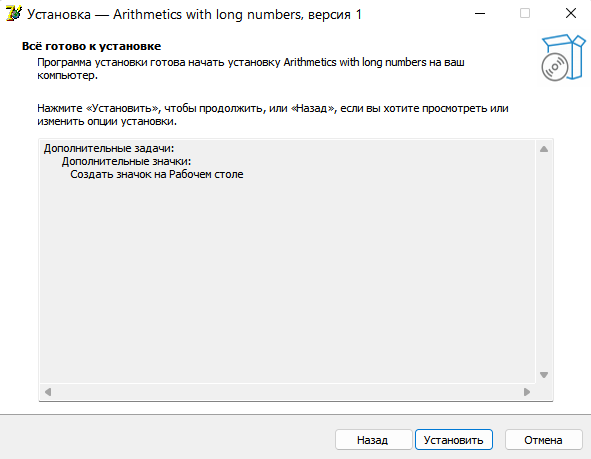


Рисунок 47 – Завершение установки

## Вычисления

Для выполнения вычислений требуется ввести значения в поля для ввода, далее выбрать одну из предложенных операций (+, - , \*, /), выбрать систему счисления (от 2-ой до 36-ой). После чего надо нажать на кнопку «=», далее выведется результат, как показано на рисунке 48

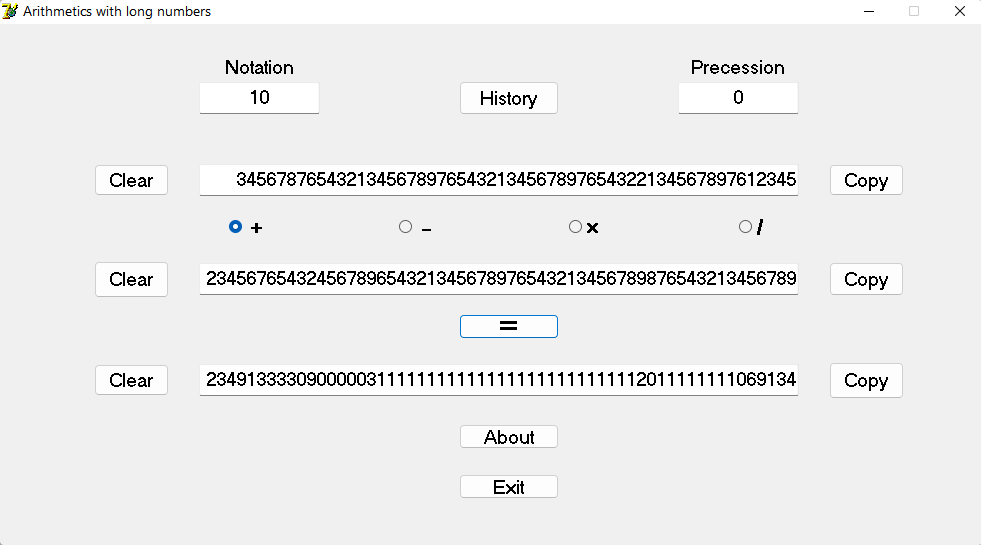


Рисунок 48 – Вычисление результата

При вводе неправильных данных будет показано окно ошибки. Некорректный опренд будет выделен, как показано на рисунке 49

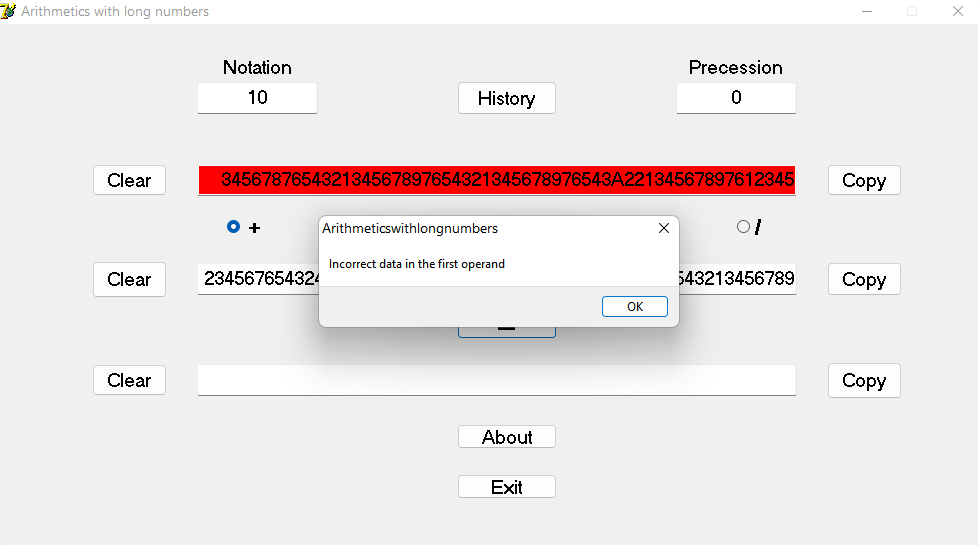


Рисунок 49 – Обработка некорректного ввода

## Установка настроек и выполнение дополнительных операций

В главном окне можно установить некоторые базовые настройки, скопировать или очистить поле, в котором находится число, открыть справку, историю, выйти из программы. При нажатии на кнопку Tab на клавиатуре можно перемещаться между элементами управления. Вид окна показан на рисунке 50

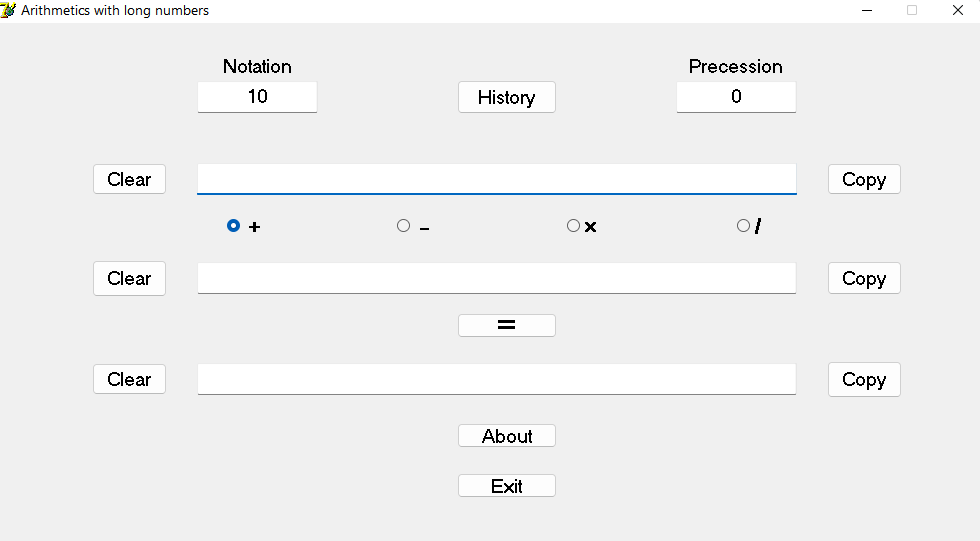


Рисунок 50 – Дополнительные настройки и операции в главном окне

«Notation» – система счисления (от 2 до 36)

«Precision» – количество знаков после запятой при делении (при выполнении остальных операции точность всегда считается равной 0)

«History» – вывод истории вычислений в виде таблицы

«About» – дополнительная информация о программе

«Clear» –очищение поля, занимаемого операндом

«Copy» – копирование операнда в буфер обмена

«Exit» – выход из программы

## Просмотр справки

Для просмотра справки необходимо нажать на кнопку «Help». Вид отображаемого окна показан на рисунке 51

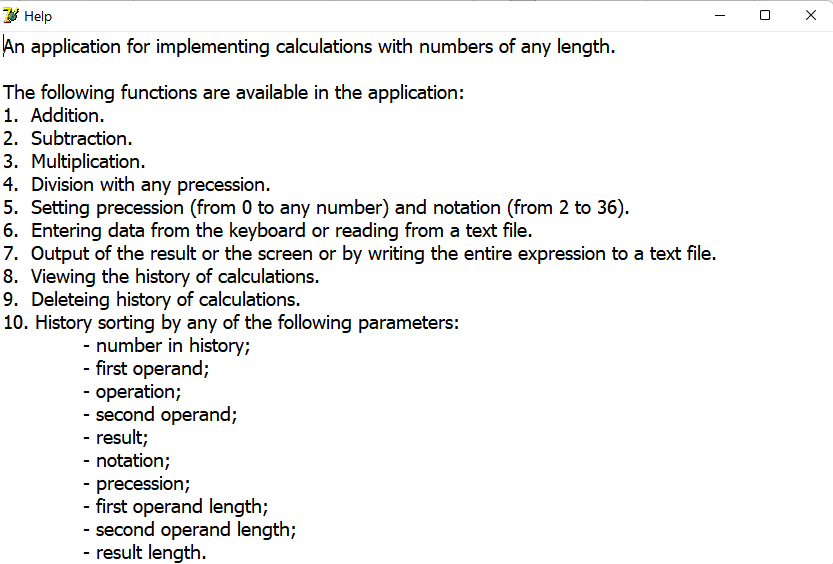


Рисунок 51 – Вид окна «Help»

Справка предоставляет краткую информацию о программном средстве и о его правильном использовании

## Работа с историей вычислений

Для просмотра истории вычислений необходимо нажать на кнопку «History». Вид окна для просмотра истории, показан на рисунке 52

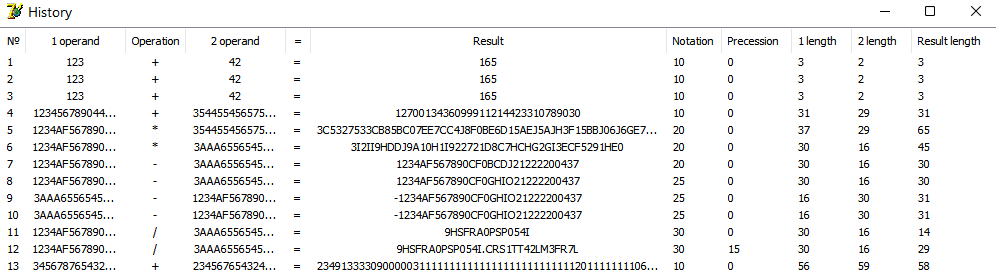


Рисунок 52 – История вычислений

В данном окне доступны все предыдущие вычисления

Заключение

По итогу работы над курсовым проектом было разработано программное средство для реализации вычислений типа «Длинная арифметика», обладающее графическим интерфейсом для взаимодействия с пользователем.

Приложение прошло все этапы тестирования, в результате которых были устранены все неполадки. Приложение имеет высокую скорость работы.

Данное программное средство, кроме основных операций над числами предоставляет пользователю возможность сохранения результата в файл, чтение данных из файла, просмотр справки, просмотр истории.

Бесценным результатом курсового проектирования является полученный опыт работы с динамической структурой данных и файлами. Был получен опыт работы с графическим интерфейсом.

При дальнейшей доработке программы, возможно появление других операции над числами (факториал, возведение в степень и др.), появление операций над числами в разных системах счисления, появление взаимодействия с вещественными числами.

Список использованной литературы

[1] Глухова, Л.А. Основы алгоритмизации и программирования ч.2: учебное пособие / Л.А.Глухова – Минск: БГУИР, 2006 – 146 с.

[2] Вирт Н. Алгоритмы и структуры данных: учебное пособие / Н.Вирт – М.: Мир, 1989 – 126 с.

[3] Ахо А. Структура данных и алгоритмы: учебное пособие / А. Ахо, Дж. Хопкрофт, Ульман Дж. – М.: Вильямс, 2003 – 45с.

[4] Длинная арифметика [Электронный ресурс]. – Режим доступа: <https://brestprog.by/topics/longarithmetics>. – Дата доступа: 19.03.2022.

[5] Длинная арифметика от Microsoft [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/207754>. – Дата доступа: 17.05.2022.

[6] Разработка способа представления длинных чисел в памяти компьютера [Электронный ресурс]. – Режим доступа: <https://moluch.ru/archive/180/46418>. – Дата доступа: 19.03.2022.

[7] Arbitrary Precision [Электронный ресурс]. – Режим доступа: <https://mathworld.wolfram.com/ArbitraryPrecision.html>. – Дата доступа: 17.05.2022.

[8] Серебряная, Л. В. Структуры и алгоритмы обработки данных : учебно-метод. пособие / Л. В. Серебряная, И. М. Марина. – Минск : БГУИР, 2013. – 51 с.

Приложение А

(обязательное)

Исходный код программы

(\* This program is developed for calculations with

numbers of any length \*)

{ Functions that program implements:

1. Addition.

2. Subtraction.

3. Multiplication.

4. Division with any precession.

5. Setting precession (from 0 to any number) and

notation (from 2 to 36).

6. Entering data from the keyboard or reading from

a text file.

7. Output of the result or the screen or by writ-

ing the entire expression to a text file.

8. Viewing the history of calculations.

9. Deleting history of calculations.

10. Loading operands from file

11. Saving calculations history as excel table }

// Main form for calculations

Unit Arithmetics;

Interface

// Libraries

Uses

Windows, Messages, SysUtils, Variants, Classes,

Graphics, Controls, Forms, Dialogs, StdCtrls, Menus,

ClipBrd, Vcl.ExtCtrls, Vcl.ComCtrls, Vcl.ExtDlgs,

Vcl.Buttons, ComObj, ActiveX;

// Type declaration

Type

TS = String[255];

// TS - string for saving history

TRec = Record

S1,S2,SRes: TS;

Operation,Eq: Char;

Position,Precession,L1,L2,LRes,Ntn: Integer;

end;

// Record that saves history

Pt = ^Elem;

Elem = Record

Data: TRec;

Next: Pt;

end;

// Queue

TFile = File of TRec;

// File for saving data

TArray = Array of Integer;

// Dynamic array

TMain = Class(TForm)

edtInput1: TEdit;

edtInput2: TEdit;

edtOutput: TEdit;

lbledtBase: TLabeledEdit;

lbledtPrec: TLabeledEdit;

rbtnAdd: TRadioButton;

rbtnSub: TRadioButton;

rbtnMul: TRadioButton;

rbtnDiv: TRadioButton;

btnEquals: TButton;

btnClearInput1: TButton;

btnClearInput2: TButton;

btnClearOutput: TButton;

btnCopyInput1: TButton;

btnCopyInput2: TButton;

btnCopyOutput: TButton;

btnInfo: TButton;

btnHist: TButton;

btnExit: TButton;

txtdialInput: TOpenTextFileDialog;

MainMenu1: TMainMenu;

Fi1: TMenuItem;

menuLoad1Operand: TMenuItem;

menuLoad2Operand: TMenuItem;

menuSavehistoryasexceltable: TMenuItem;

menuClearHistory: TMenuItem;

Procedure ReturnColor(Sender: TObject);

Procedure btnEqualsClick(Sender: TObject);

Procedure btnHistClick(Sender: TObject);

Procedure btnExitClick(Sender: TObject);

Procedure btnInfoClick(Sender: TObject);

Procedure btnCopyInput1Click(Sender: TObject);

Procedure btnCopyInput2Click(Sender: TObject);

Procedure btnCopyOutputClick(Sender: TObject);

Procedure btnClearInput1Click(Sender: TObject);

Procedure btnClearInput2Click(Sender: TObject);

Procedure btnClearOutputClick(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure FormClose(Sender: TObject; var Action:

TCloseAction);

Procedure bbtnInput1Click(Sender: TObject);

Procedure menuLoad1OperandClick(Sender: TObject);

Procedure menuLoad2OperandClick(Sender: TObject);

Procedure menuSavehistoryasexceltableClick(Sender:

TObject);

Procedure menuClearHistoryClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

end;

// Global variable declaration

Var

Main: TMain;

ListHist: Elem;

First,Last: Pt;

ExcelVisible: Boolean;

// Main - main class

// ListHist - record of current operation data

// First - pointer on the beginning of the queue

// Last - pointer of the end of the queue

// ExcelVisible - checking if an excel table already

// exists

Implementation

{$R \*.dfm}

// Modules declaration

Uses

HistF,HelpF;

(\* Start of Input subprogram. Subprogram creates

dynamic array that consists of the number in inverce

form. Converts letters into numbers \*)

// Str - beginning string

// Arr - array for input

// Notation - base of the counting system

// Flag - checking string for correct input

Procedure Input(Var Str: String; Var Arr: TArray; Const

Notation: Integer; Var Flag: Boolean);

// Local variables declaration

Var

Error,I: Integer;

// Error - check of input

// I - cycle iterator

begin

// Initialisation of starting parameters

Flag := true;

setlength(Arr,length(Str));

// If notation is less or equal then 10 then letters

// cannot be added

if Notation <= 10 then

begin

// Putting numbers from string to the array in the

// inverted form

for I := 1 to length(Str) do

begin

val(Str[I],Arr[length(Str) - I],Error);

// Checking if input is correct

if Error <> 0 then

Flag := false;

// Checking if input is correct

if Arr[length(Str) - I] >= Notation then

Flag := false;

end;

end

// If notation is more then 10 then letters can be

// added

else

begin

// Putting numbers or letters as numbers from

// string to the array in the inverted form

for I := 1 to length(Str) do

begin

val(Str[I],Arr[length(Str) - I],Error);

if Error <> 0 then

// Converting letters into numbers for the

// integer array

if (ord(Str[I]) - ord('A')) < (Notation - 10)

then

begin

Arr[length(Str) - I] := 10 + ord(Str[I]) –

ord('A');

end

else

begin

Flag := false;

end;

end;

end;

end;

// End of the Input subprogram

(\* Start of FindResLen subprogram. Subprogram counts

lengths of the strings and finds the biggest length

between them \*)

// Str1,Str2 - Starting strings

Function FindResLen(Const Str1,Str2: String): Integer;

begin

// Compareing strings

// If first string length is bigger then the second

// one, length of the first string is the searching

// result

if length(Str1) > length(Str2) then

begin

Result := length(Str1)

end

//If the second string length is bigger or equat,

//length of the second string is the searching result

else

begin

Result := length(Str2);

end;

end;

// End of FindResLen subprogram

(\* Start of Convertion subprogram. Digits from array

are being converted to string digits or letters

depending on notation \*)

// A - starting array

// Result - returnins string parameter

Function Convertion(Const A: TArray): String;

// Local variables declaration

Var

I: Integer;

// I - cycle iterator

begin

// Initializing result

Result := '';

(\* Converting digits into chars. Convertion depends

on the notation. If notation is bigger or equal

than 10 then digits must be converted into

letters \*)

for I := high(A) downto low(A) do

if A[I] >= 10 then

begin

Result := Result + chr(A[I] - 10 + ord('A'));

end

else

begin

Result := Result + chr(A[I] + ord('0'));

end;

end;

// End of the Convertion subprogram

(\* Start of the DeleteZero subprogram. Subprogram

deletes zeros from the end of the array\*)

// A - starting array

Procedure DeleteZero (Var A: TArray);

// Local variables declaration

Var

I: Integer;

// I - cycle iterator

begin

// Initializing starting parameters

I := high(A);

(\* Cycle decrements I parameter to count the length

of the array without zeros in the end\*)

while (A[I] = 0) and (I > 0) do

Dec(I);

// Setting new length

setlength(A,I + 1);

end;

// End of the DeleteZero subprogram

(\* Start of the ToExp subprogram. Subprogram converts

numbers with length more then 255 to exponencial

form to write it to file in full form \*)

// S - starting string

// P - Precession

Procedure ToExp(Var S: String; Const P: Integer);

// Local variables declration

Var

Tmp: String[245];

I: Integer;

// Tmp - Saved part of the string

// I - detets zeros from the beginning of the number

begin

// If string won't fit String[255] it will be

// converted into exponential form

if length(S) > 255 then

begin

// Copying 245 symbols of the original string

Tmp := S;

// Deleteing minus from the beginning

if Tmp[1] = '-' then

delete(Tmp,1,1);

// Deleteing the first number

delete(Tmp,1,1);

// If length of the oroginal string - precession =

// 2 and the second symbol is dot

if (length(S) - P = 2) and (S[2] = '.') then

begin

// Starting from the third symbol

I := 3;

// Counting zeros after the 3 symbol

while S[I] = '0' do

Inc(I);

// Copying Symbols from the I position

Tmp := copy(S,I,244);

// Deleteing the first symbol

delete(Tmp,1,1);

// Convertion into exponential form

S := S[I] + '.' + Tmp + ' E-' + IntToStr(I - 2);

end

else

begin

// If the first symbol of the original string is

// minus, it will be added

if S[1] = '-' then

S := S[1] + S[2] + '.' + Tmp + ' E' +

IntToStr(length(S) - P - 2)

else

S := S[1] + '.' + Tmp + ' E' +

IntToStr(length(S) - P - 2);

end;

end;

end;

// End of the ToExp subprogram

(\* Start of the Summarise subprogram. The programm adds

one number to another\*)

// A - first number

// B - second number

// C - result

// Notation - base of the scale

Procedure Summarise(Const A,B: TArray; Var C: TArray;

Const Notation: Integer);

// Local variables declaration

Var

I,J,Temp: Integer;

// I,J - cycle iterators

// Temp - transfer to the higher digit

begin

// Initialization

I := 0;

Temp := 0;

// Setting length of the result

if length(A) > length(B) then

begin

setlength(C,length(A) + 1);

end

else

begin

setlength(C,length(B) + 1);

end;

// Adding first number to the second until one of

// them is over

while (I <= high(A)) and (I <= high(B)) do

begin

C[I] := (Temp + A[I] + B[I]) mod Notation;

Temp := (Temp + A[I] + B[I]) div Notation;

Inc(I);

end;

// Adding 1 or 0 to higher digit

C[I] := C[I] + Temp;

// If the first number is bigger then the second, its

// higer digits are being put to the result without

// adding

for J := I to high(A) do

C[J] := C[J] + A[J];

// If the second number is bigger then the first, its

// higher digits are being put to the result without

// adding

for J := I to high(B) do

C[J] := C[J] + B[J];

// Deleting zeros from the beginning of the number

DeleteZero(C);

end;

// End of the Summarise subprogram

(\* Start of the Subtraction subprogram. Subprogram

subtracts one number from another\*)

// A - first number

// B - second number

// C - result

// Notation - base of the scale

// Sign - adding minus or not

Procedure Subtraction(Var A,B,C: TArray; Const

Notation: Integer; Var Sign:

Boolean);

// Local variables declaration

Var

I,J,Temp: Integer;

Supp: TArray;

StrA,StrB: String;

// I,J - cycle iterators

// Temp - borrowing from the higher digit

// Supp - swapping numbers if A < B

// StrA - string with first number

// StrB - string with the second number

begin

// Initialization

I := 0;

Temp := 0;

Sign := true;

StrA := Convertion(A);

StrB := Convertion(B);

// Setting result length

setlength(C,FindResLen(StrA,StrB));

// Setting zeros to the result

for J:=low(C) to high(C) do

C[J] := 0;

// Making the biggest of two numbers the first

// operand, finding the sign of the result

if (length(StrA) < length(StrB)) or

((length(StrA) = length(StrB)) and (StrA < StrB))

then

begin

Supp := A;

A := B;

B := Supp;

Sign := false;

end;

// Subtracting while the second number is not over

while I <= high(B) do

begin

C[I] := A[I] - B[I] - Temp;

Temp := 0;

if C[I] < 0 then

begin

C[I] := C[I] + Notation;

Temp := 1;

end;

Inc(I);

end;

// If A is longer, its unused digits will be taken

// into the result without subtracting

for J := I to high(A) do

begin

C[J] := A[J] - Temp;

Temp := 0;

if C[J] < 0 then

begin

C[J] := C[J] + Notation;

Temp := 1;

end;

end;

// Deleting zeros from the beginning of the result

DeleteZero(C);

end;

// End of the Subtraction subprogram

(\* Start of the Multiply subprogram. Subprogram

multiplies two numbers \*)

// A - the first number

// B - the second number

// C - result

// Notation - base of the scale

Procedure Multiply(Const A,B: TArray; Var C: TArray;

Const Notation: Integer);

// Local variables declaration

Var

I,J,Temp: Integer;

// I,J - cycle iterators

// Temp - multilication of digits result

begin

// Setting the result length

setlength(C,length(A) + length(B));

// Making all the elements zeros

for J:=low(C) to high(C) do

C[J] := 0;

// Multiplying two numbers

for I := low(A) to high(A) do

for J := low(B) to high(B) do

begin

Temp := A[I] \* B[J];

// Currend digit

C[I + J] := Temp mod Notation + C[I + J];

// Moving to the higher digit

C[I + J + 1] := Temp div Notation + C[I + J + 1];

end;

// Rediction to Notation

for J := low(C) to high(C) do

if C[J] >= Notation then

begin

C[J + 1] := C[J + 1] + C[J] div Notation;

C[J] := C[J] mod Notation

end;

// Deleting zeros from the beginning of the result

DeleteZero(C);

end;

// End of Multiplication subprogram

(\* Start of the DivFrac subprogram. Subprogram calcu-

lates fractional part of the number until the pre-

cession is reached\*)

// A - first number

// B - second number

// C - result

// Notation - base of the scale

// Acc - accuracy entered by user

Procedure DivFrac(Var A,B,C: TArray; Const Notation,

Acc: Integer);

// Local variables declaration

Var

I,K: Integer;

AC,Ten,Save: TArray;

Flag: Boolean;

// I - current digit of the result

// K - cycle iterator

// AC - copy of the A array

// Ten - multiplication by 10

// Save - copy of the B array

// Flag - sign while subtracting

begin

// Initialization

Setlength(Ten,2);

Ten[low(Ten)] := 0;

Ten[high(Ten)] := 1;

// Counting until accuracy is reached

for K := 1 to Acc do

begin

// Copying B array

Save := B;

// Setting new length

setlength(AC,length(B) + 1);

// Multiplying by 10 - moving to the higher digit

// the whole number

Multiply(A,Ten,AC,Notation);

I := 0;

// Setting new length of the A array

setlength(A,length(AC));

// Subtracting until AC is less then Save. Counting

// the current digit

repeat

Subtraction(AC,Save,A,Notation,Flag);

setlength(AC,length(A));

AC := A;

Inc(I);

until (I = Notation) or (not Flag);

// Return second number copy back

Save := B;

// Subtracting

Subtraction(AC,Save,A,Notation,Flag);

// Adding to the result new digit

C[Acc - K] := I - 1;

end;

end;

// End of the DivFlac subprogram

(\* Start of the Division subprogram.\*)

// A - first number

// B - second number

// C - result

// D - fractional part of the number

// Notation - base of the scale

// P - precession

Procedure Division(Var A,B,C,D: TArray; Const Notation,

P: Integer);

// Local variables declaration

Var

I,J,Shift: Integer;

Mul,AC,Sub,Digit: TArray;

S1,S2: String;

Flag: Boolean;

// I - current digit

// J - cycle iterator

// Shift - difference between lengthses of numbers

// Mul - multiplacation result

// AC - copy of the A array

// Sub - subtraction of AC and Mul

// Digit - current digit of the result number

// S1 - string variant of A array

// S2 - string variant of B array

// Flag - sign of the result of subtraction

begin

// Setting new lengthes of the result arrays

setlength(D,P);

setlength(AC,length(A));

// Creadting string variants of the operands

S1 := Convertion(A);

S2 := Convertion(B);

// Setting subtraction result length

setlength(Sub,FindResLen(S1,S2));

// Checking is C array in not zero. If the first

// number is less then the second, result will be

// only of fractional part

if ((length(S1) = length(S2)) and (S1 >= S2)) or

(length(S1) > length(S2)) then

begin

// Counting differences in lengths of operands

Shift := length(A) - length(B);

// Setting the result length

setlength(C,Shift + 1);

// Making the result array 0

for I := low(C) to high(C) do

C[I] := 0;

// Setting length for the division result

setlength(Digit,Shift + 1);

// Making calculations

while Shift >= 0 do

begin

I := 0;

repeat

// Saving A array

AC := A;

// Setting Digit array 0

for J := low(Digit) to high(Digit) do

Digit[J] := 0;

// I - result of the division

Digit[Shift] := I;

Multiply(B,Digit,Mul,Notation);

Subtraction(AC,Mul,Sub,Notation,Flag);

Inc(I);

until (not Flag) or (I > Notation);

// Filling in the result

C[Shift] := I - 2;

Digit[Shift] := I - 2;

// If it is not possible to divide, 0 will be

// written into the result array

if (length(Mul) = 1) and (Mul[low(Mul)] = 0) then

begin

C[Shift] := 0;

Digit[Shift] := 0;

end;

// Making new numerator

Multiply(B,Digit,Mul,Notation);

Subtraction(A,Mul,AC,Notation,Flag);

// Setting new numerator

A := AC;

Dec(Shift);

end;

// Counting fractional part

DivFrac(A,B,D,Notation,P);

end

else

begin

// Counting fractional part. C array is 0

setlength(C,1);

C[low(C)] := 0;

DivFrac(A,B,D,Notation,P);

end;

// Deleting zeros from the beginning

DeleteZero(C);

end;

// End of the Division subprogram

(\* Start of the Push subprogram. Adding new element to

the queue \*)

// X - first element of the queue

// Y - last element of the queue

// R - data

Procedure Push (Var X,Y: Pt; Const R: TRec);

// Local variables declaration

Var

PTmp: Pt;

// PTmp - new element of the queue

begin

// Initialization

new(PTmp);

PTmp^.data := R;

PTmp^.next := nil;

// If queue is empty, it will be created

if X = nil then

begin

X := PTmp;

end

else

begin

Y^.next := PTmp;

end;

// Adding element

Y := PTmp;

end;

// End of the Push subprogram

(\* Start of the PopFile subprogram. Subprogram writes

data to file from queue \*)

Procedure PopFile (X: Pt);

// Local variables declaration

Var

F: TFile;

// Current file

begin

// Assigning file, setting file mode

AssignFile(F,'Computations');

Rewrite(F);

// Writing queue elements to file

while X <> nil do

begin

write(F,X^.data);

X := X^.next;

end;

// Saving changes

CloseFile(F);

end;

// End of PopFile subprogram

(\* Start of the ReadFile subprogram. Subprogram pushes

elements from file to queue \*)

// Name - Name of the file

Procedure ReadFile (Const Name: String);

// Local variables declaration

Var

F: File of TRec;

Tmp: TRec;

// F - current file

// Tmp - current record

begin

// Assigning file, setting file mode

AssignFile(F,Name);

Reset(F);

// Pushing elements to the queue

while not EoF(F) do

begin

Read(F,Tmp);

Push(First,Last,Tmp);

end;

// Saving changes

CloseFile(F);

end;

// End of ReadFile subprogram

(\* Start of CheckInput subprogram. Subprogram checks

input of base \*)

// Param - checking parameter

Procedure CheckInput(Var Param: Integer);

begin

// Checking data

if (Param < 2) or (Param > 36) then

Param := 10;

end;

// End of Checkinput subprogram

(\* When form is created, file Computations is read \*)

Procedure TMain.FormCreate(Sender: TObject);

begin

ReadFile('Computations');

end;

(\* Loading first operand from file \*)

Procedure TMain.menuLoad1OperandClick(Sender: TObject);

// Local variables declaration

Var

Num,Name: String;

F: TextFile;

// Num - number in file

// Name - name of the file

// F - file itself

begin

// Loading data from file to edit field

if txtdialInput.Execute then

begin

Name := txtdialInput.FileName;

AssignFile(F,Name);

Reset(F);

read(F,Num);

edtInput1.Text := Num;

CloseFile(F);

end;

end;

(\* Loading second operand from file\*)

Procedure TMain.menuLoad2OperandClick(Sender: TObject);

// Local variables declaration

Var

Num,Name: String;

F: TextFile;

// Num - number in file

// Name - name of the file

// F - file itself

begin

// Loading data from file to edit field

if txtdialInput.Execute then

begin

Name := txtdialInput.FileName;

AssignFile(F,Name);

Reset(F);

read(F,Num);

edtInput2.Text := Num;

CloseFile(F);

end;

end;

(\* Saving history in excel \*)

Procedure TMain.menuSavehistoryasexceltable

Click(Sender: TObject);

// Local variables declaration

Var

ExcelApp,ExcelSheet: OleVariant;

X: Pt;

Raw,Col: Integer;

// ExcelApp - excel appliation

// ExcelSheet - field to fill

// X - copy of queue pointer

// Raw - current raw

// Col - current column

begin

// If excel is already visible, an error will be

// shown

if not ExcelVisible then

begin

ExcelApp := CreateOleObject('Excel.Application');

ExcelApp.Visible := True;

ExcelVisible := true;

end;

// Creating field to fill

ExcelApp.Workbooks.Add(1);

ExcelApp.Workbooks[1].WorkSheets[1].Name:='History';

ExcelSheet:=ExcelApp.Workbooks[1].WorkSheets

['History'];

// Initialization

X := First;

Raw := 1;

Col := 1;

// Creating table header

ExcelSheet.Cells[Raw,Col] := '№';

Inc(Col);

ExcelSheet.Cells[Raw,Col] := '1 operand';

Inc(Col);

ExcelSheet.Cells[Raw,Col] := 'Operation';

Inc(Col);

ExcelSheet.Cells[Raw,Col] := '2 operand';

Inc(Col);

ExcelSheet.Cells[Raw,Col] := '=';

Inc(Col);

ExcelSheet.Cells[Raw,Col] := 'Result';

Inc(Col);

ExcelSheet.Cells[Raw,Col] := 'Notation';

Inc(Col);

ExcelSheet.Cells[Raw,Col] := 'Precession';

Inc(Col);

ExcelSheet.Cells[Raw,Col] := '1 length';

Inc(Col);

ExcelSheet.Cells[Raw,Col] := '2 length';

Inc(Col);

ExcelSheet.Cells[Raw,Col] := 'Result length';

Inc(Raw);

Col := 1;

// Filling in table elements

while X <> nil do

begin

with X^.Data do

begin

ExcelSheet.Cells[Raw,Col] := Position;

Inc(Col);

ExcelSheet.Cells[Raw,Col] := S1;

Inc(Col);

ExcelSheet.Cells[Raw,Col] := Operation;

Inc(Col);

ExcelSheet.Cells[Raw,Col] := S2;

Inc(Col);

ExcelSheet.Cells[Raw,Col] := Eq;

Inc(Col);

ExcelSheet.Cells[Raw,Col] := SRes;

Inc(Col);

ExcelSheet.Cells[Raw,Col] := Ntn;

Inc(Col);

ExcelSheet.Cells[Raw,Col] := Precession;

Inc(Col);

ExcelSheet.Cells[Raw,Col] := L1;

Inc(Col);

ExcelSheet.Cells[Raw,Col] := L2;

Inc(Col);

ExcelSheet.Cells[Raw,Col] := LRes;

Col := 1;

Inc(Raw);

X := X^.Next;

end;

end;

end;

(\* Button History work logic \*)

Procedure TMain.btnHistClick(Sender: TObject);

// Local variables declaration

Var

Item: TListItem;

X: Pt;

// Item - element of ListView

// X - copy of the queue start pointer

begin

// Making ListView empty

Hist.lvHist.Clear;

// Copying pointer on the queue beginning

X := First;

// Ading elements from queue to ListView

while X <> nil do

begin

Item := Hist.lvHist.Items.Add;

with X^.Data do

begin

Item.Caption := IntToStr(Position);

Item.SubItems.Add(S1);

Item.SubItems.Add(Operation);

Item.SubItems.Add(S2);

Item.SubItems.Add(Eq);

Item.SubItems.Add(SRes);

Item.SubItems.Add(IntToStr(Ntn));

Item.SubItems.Add(IntToStr(Precession));

Item.SubItems.Add(IntToStr(L1));

Item.SubItems.Add(IntToStr(L2));

Item.SubItems.Add(IntToStr(LRes));

end;

// Move to the next element of the queue

X := X^.Next;

end;

// Showing Hist form

Hist.ShowModal;

end;

(\* Button About logic\*)

Procedure TMain.btnInfoClick(Sender: TObject);

begin

// Showing Help form

Help.ShowModal;

end;

(\* Clearing history \*)

Procedure TMain.menuClearHistoryClick(Sender: TObject);

// Local variables declaration

Var

F: TFile;

// F - file to clear

begin

// Deleting queue

First := nil;

Last := nil;

// Deleting ListView data

Hist.lvHist.Clear;

// Deleting information that was saved in file

AssignFile(F,'Computations');

Rewrite(F);

CloseFile(F);

end;

(\* = button logic. Makes calculations, checks data\*)

Procedure TMain.btnEqualsClick(Sender: TObject);

// Local variables declaration

Var

Op1,Op2,Res: String;

ZF,Flag,Op1Sign,Op2Sign: Boolean;

ArrA,ArrB,ArrC,ArrD: TArray;

Base,Prec: Integer;

Rec: TRec;

// Op1 - first number in string form

// Op2 - second number in string form

// ZF - zero flag

// Flag - sign for subtaction

// Op1Sign - check in first number is correct

// Op2Sign - check if second number is correct

// ArrA - first number in array

// ArrB - second number in array

// Arrc - result in array

// ArrD - fractional part of the number in array

// Base - corrent notation

// Prec - precession for division

// Rec - current record of data for calculations

// history

begin

// Return colors

edtInput1.Color := clWindow;

edtInput2.Color := clWindow;

// Set starting data if the field is empty

if lbledtBase.Text = '' then

lbledtBase.Text := '10';

if lbledtPrec.Text = '' then

lbledtPrec.Text := '0';

if edtInput1.Text = '' then

edtInput1.Text := '0';

if edtInput2.Text = '' then

edtInput2.Text := '0';

// Clering the result field

edtOutput.Text := '';

// Initialization

Op1Sign := true;

Op2Sign := true;

ZF := true;

Flag := true;

// Checkin input

Prec := StrToInt(lbledtPrec.Text);

Base := StrToInt(lbledtBase.Text);

CheckInput(Base);

lbledtBase.Text := IntToStr(Base);

// Filling in strings that store numbers

Op1 := edtInput1.Text;

Op2 := edtInput2.Text;

// Converiong string into array. Checking input

Input(Op1,ArrA,Base,Op1Sign);

Input(Op2,ArrB,Base,Op2Sign);

// Deleting zeros from the beginning of the arrays

DeleteZero(ArrA);

DeleteZero(ArrB);

// In numbers are correct, making calculations. If

// not, show error message and highlight incorrect

// field

if Op1Sign and Op2Sign then

begin

// Choosing operation depending on the round

// buttons

if rbtnAdd.Checked then

begin

// Setting res = 0

Res := '';

// Precession is 0 for all operations except

// division

Rec.Precession := 0;

// Filling in an operation sign

Rec.Operation := '+';

// Adding

Summarise(ArrA,ArrB,ArrC,Base);

// Filling in the result

Res := Res + Convertion(ArrC);

// Showing the result

edtOutput.Text := Res;

end;

if rbtnSub.Checked then

begin

// Setting res = 0

Res := '';

// Precession is 0 for all operations except

// division

Rec.Precession := 0;

// Filling in an operation sign

Rec.Operation := '-';

// Making calculations

Subtraction(ArrA,ArrB,ArrC,Base,Flag);

// Adding sign in the beginning if A < B

if not Flag then

Res := '-';

// Filling in the result

Res := Res + Convertion(ArrC);

// Showing the result

edtOutput.Text := Res;

end;

if rbtnMul.Checked then

begin

// Setting res = 0

Res := '';

// Precession is 0 for all operations except

// division

Rec.Precession := 0;

// Filling in an operation sign

Rec.Operation := '\*';

// Making calculations

Multiply(ArrA,ArrB,ArrC,Base);

// Filling in the result

Res := Res + Convertion(ArrC);

// Showing the result

edtOutput.Text := Res;

end;

if rbtnDiv.Checked then

begin

// Setting res = 0

Res := '';

// Saving precession that was entered by the user

Rec.Precession := Prec;

// Filling in an operation sign

Rec.Operation := '/';

// If the second number is 0, showing and error

// of division by 0

if ArrB[high(ArrB)] = 0 then

begin

edtInput2.Color := clRed;

ShowMessage('Division by zero');

// Zero flag

ZF := false;

end

else

begin

// Making calculations

Division(ArrA,ArrB,ArrC,ArrD,Base,Prec);

// Filling in the result

Res := Res + Convertion(ArrC);

// If precession > 0 then there is a fractional

// part of the number

if Prec > 0 then

begin

// Adding dot

Res := Res + '.';

// Adding fractional part

Res := Res + Convertion(ArrD);

end;

// Showing the result

edtOutput.Text := Res;

end;

end;

// Filling in the current operation history record

with Rec do

begin

L1 := length(Op1);

L2 := length(Op2);

LRes := length(Res);

// If precession <> 0, dot is not a part of

// number length

if Prec <> 0 then

Dec(LRes);

// Minus in not a part of number length

if not Flag then

Dec(LRes);

end;

// Converiong operands into exponencial form is

// they are too long

ToExp(Op1,Prec);

ToExp(Op2,Prec);

ToExp(Res,Prec);

// Filling in the current operation history record

with Rec do

begin

S1 := Op1;

S2 := Op2;

SRes := Res;

Eq := '=';

Ntn := Base;

end;

// Counting current number of operation

if First = nil then

begin

Rec.Position := 1;

end

else

begin

Rec.Position := Last^.Data.Position + 1;

end;

// If there was no zero division, add data to

// history

if ZF then

Push(First,Last,Rec);

end

else

begin

// Showing error messages

if not Op1Sign then

begin

// Painting an incorrect operand

edtInput1.Color := clRed;

ShowMessage('Incorrect data in the first

operand');

end;

if not Op2Sign then

begin

// Painting an ibcorrect operand

edtInput2.Color := clRed;

ShowMessage('Incorrect data in the second

operand');

end;

end;

end;

procedure TMain.bbtnInput1Click(Sender: TObject);

begin

end;

(\* Clear buttons logic \*)

Procedure TMain.btnClearInput1Click(Sender: TObject);

begin

edtInput1.Text := '';

end;

Procedure TMain.btnClearInput2Click(Sender: TObject);

begin

edtInput2.Text := '';

end;

Procedure TMain.btnClearOutputClick(Sender: TObject);

begin

edtOutput.Text := '';

end;

(\* Copy buttons logic \*)

Procedure TMain.btnCopyInput1Click(Sender: TObject);

begin

Clipboard.AsText := edtInput1.Text;

end;

Procedure TMain.btnCopyInput2Click(Sender: TObject);

begin

Clipboard.AsText := edtInput2.Text;

end;

Procedure TMain.btnCopyOutputClick(Sender: TObject);

begin

Clipboard.AsText := edtOutput.Text;

end;

(\* Set color from red to window color \*)

Procedure TMain.ReturnColor(Sender: TObject);

Var

Edit: TEdit;

begin

if not(Sender is TEdit) then

Exit;

Edit := TEdit(Sender);

Edit.Color := clWindow;

end;

(\* Saving history from queue to file \*)

Procedure TMain.FormClose(Sender: TObject; Var Action:

TCloseAction);

begin

PopFile(First);

end;

(\* Exit button work logic \*)

Procedure TMain.btnExitClick(Sender: TObject);

begin

Main.Close;

end;

end.

(\* Calculation history. Uses ListView to show table of

History \*)

Unit HistF;

Interface

// Libraries

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils,

System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs,Vcl.ComCtrls,

Vcl.Grids, Vcl.ExtCtrls, Vcl.Menus;

// Types declaration

Type

THist = Class(TForm)

lvHist: TListView;

Private

{ Private declarations }

Public

{ Public declarations }

end;

// Global variables declaration

Var

Hist: THist;

// Hist - form for showing history

Implementation

{$R \*.dfm}

end.

(\* Help form. Uses Memo to represent the text of

Help.txt \*)

Unit HelpF;

Interface

// Libraries

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils,

System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

// Type declaration

Type

THelp = Class(TForm)

mHelp: TMemo;

Procedure FormCreate(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

end;

// Global variables declaration

Var

Help: THelp;

// Help - help form with text

Implementation

{$R \*.dfm}

(\* Showing text from file on the memo object \*)

Procedure THelp.FormCreate(Sender: TObject);

begin

try

mHelp.Lines.LoadFromFile('Help.txt');

except

ShowMessage('Help file not found');

end;

end;

end.