

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**«Московский государственный технический университет радиотехники, электроники и автоматики»**

**МГТУ МИРЭА**

***Институт информационных технологий (ИТ)***

«наименование факультета»

Кафедра базовая автоматизированных систем организационного управления (**АСОУ**) **№239** МГТУ МИРЭА при ФГУП НИИ «Восход»

«наименование кафедры»

**КУРСОВАЯ РАБОТА**

**по дисциплине**

\_\_\_\_\_Программирование 1\_\_\_\_

<Наименование дисциплины>

**Тема курсовой работы:** «Калькулятор»

<Название темы курсовой работы/проекта>

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИВБ-3-14 |  |  | Панасенко А.В. |
| <код группы> | <подпись студента> |  | <ФИО студента> |
| Руководитель курсовой работы |  |  | ассистент |
|  |  |  | <должность, звание, уч.степень> |
|  |  |  | Хлебников А.А. |
|  | <подпись руководителя> |  | < ФИО руководителя > |
| Рецензент *(при наличии)* |  |  |  |
|  |  |  | <должность, звание, уч.степень> |
|  |  |  |  |
|  | <подпись рецензента> |  | < ФИО рецензента> |
|  |  |  |  |
|  |  |  |  |
| *Работа представлена к защите* | «\_\_\_»\_\_\_\_\_\_\_\_201\_ г. |  |  |
|  |  |  | <подпись студента> |
| *«Допущен к защите»* | «\_\_\_»\_\_\_\_\_\_\_\_201\_ г. |  |  |
|  |  |  | <подпись руководителя> |



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**«Московский государственный технический университет радиотехники, электроники и автоматики»**

**МГТУ МИРЭА**

***Институт информационных технологий (ИТ)***

«наименование факультета»

Кафедра базовая автоматизированных систем организационного управления (**АСОУ**) **№239** МГТУМИРЭА при ФГУП НИИ «Восход»

«наименование кафедры»

|  |  |
| --- | --- |
|  | **УТВЕРЖДАЮ**  Заведующий  кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Волков  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_\_г. |

**ЗАДАНИЕ**

**на выполнение курсовой работы**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **по дисциплине** | | «Программирование 1» | | | |
|  | | «Название дисциплины» | | | |
| Студент | Панасенко Андрей Викторович | | Группа | ИВБ-3-14 |
|  |  | |  |  |

**1.Тема курсовой работы**

«Калькулятор»

**2. Исходные данные**

**̶** Выражение в инфиксной форме

**2. Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

* разработка текстового интерфейса меню и подменю калькулятора
* разработка приложения калькулятора.

**4. Срок представления к защите курсового проекта (работы):** до «\_\_\_» \_\_\_\_\_\_\_\_\_\_201\_ г.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Задание на курсовой проект (работу) выдал | «\_\_\_»\_\_\_\_\_\_\_201\_ г. |  |  |  | Петров М.Ю. |
|  |  |  | <подпись руководителя проекта> |  | <Ф.И.О. руководителя проекта> |
| Задание на курсовой проект (работу) получил | «\_\_\_»\_\_\_\_\_\_\_201\_ г. |  |  |  | Иванов И.И. |
|  |  |  | <подпись студента-исполнителя проекта> |  | <Ф.И.О. студента-исполнителя проекта> |

**Мониторинг процесса выполнения курсового проекта (работы)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  э  т  апа | Наименование этапа курсового проекта, работы | Этап курсового проекта, работы выполнил и представил результаты руководителю проекта (работы),  *дата и подпись исполнителя* | Работу по этапу курсового проекта (работы) принял на рассмотрение,  *дата и подпись руководителя* | Рекомендации и замечания по этапу курсового проекта (работы) выдал исполнителю,  *дата и подпись руководителя* | Оценка выполнения этапа курсового проекта, (работы)*(в соответствии с балльно-рейтинговой системой)* | Комментарии руководителя курсового проекта (работы) |
| 1 | Разработка задания на КР |  |  |  |  |  |
| 2 | Анализ задачи |  |  |  |  |  |
| 3 | Разработка алгоритма задачи |  |  |  |  |  |
| 4 | Разработка структуры проекта |  |  |  |  |  |
| 5 | Разработка интерфейса пользователя |  |  |  |  |  |
| 6 | Разработка схемы программных взаимодействия единиц, группируя по модулям |  |  |  |  |  |
| 7 | Разработка приложения |  |  |  |  |  |
| 8 | Разработка пояснительной записки |  |  |  |  |  |
| 9 | Защита курсовой работы |  |  |  |  |  |

Реферат

**Объект исследования** – программирование на языке С++

**Предмет исследования** – Калькулятор.

**Цель работы**: закрепление и углубление практических знаний по программированию на языке С++, ориентированных на создание калькулятора.

В процессе работы проводилась разработка алгоритмов, тестирование и отладка как самого приложения, так и отдельных его модулей.

**Область возможного практического применения:** создание крупных коммерческих вычислительных приложений

**ЛИСТ ЗАМЕЧАНИЙ**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Содержание**

[Введение](#_Toc344118161) 7-8

1. [Основная часть 9](#_Toc344118162)-20

[1.1. Обоснование выбора средств и методов для выполнения задания 8](#_Toc344118163)

[1.2 Интерфейс командной строки](#_Toc344118167) 15-17

1.3 Реализация работы с RPN 18-20

2.1 [Заключение 21](#_Toc344118192)

2.2 Список литературы.............................................................................................22

2.3 Приложение...................................................................................................23-25

**Введение**

Широкое распространение **языка C** на различных типах компьютеров (иногда называемых аппаратными платформами) привело, к сожалению, ко многим вариациям языка. Они были похожи, но несовместимы друг с другом. Это было серьезной проблемой для разработчиков программ, нуждавшихся в написании совместимых программ, которые можно было бы выполнять на нескольких платформах. Стало ясно, что необходима стандартная версия *C*. В 1983г. ANSI (Американский Национальный Комитет Стандартов) сформировал технический комитет X3J11 для создания стандарта языка *C* (чтобы "обеспечить недвусмысленное и машинно-независимое определение языка"). В 1989 стандарт был утвержден. ANSI скооперировался с ISO (Международной Организацией Стандартов), чтобы стандартизовать *C* в международном масштабе; совместный стандарт был опубликован в 1990 году и назван ANSI/ISO 9899:1990. Этот стандарт совершенствуется до сих пор и поддерживается большинством фирм разработчиков компиляторов.

**Бьерн Страуструп** высвободил объектно-ориентированный потенциал *С* путем перенесения возможностей классов *Simula 67* в *С*. Первоначально новый язык носил имя "С с классами" и только потом стал называться *C++*. Язык *C++* достиг популярности, будучи разработанным в Bell Labs, позже он был перенесен в другие индустрии и корпорации. Сегодня это один из наиболее популярных языков программирования в мире. *C++* наследует как хорошие, так и плохие стороны *С*.

**Бьерн Страуструп**: "Я придумал C++, записал его первоначальное определение и выполнил первую реализацию. Я выбрал и сформулировал критерии проектирования *C++*, разработал его основные возможности и отвечал за судьбу предложений по расширению языка в комитете по стандартизации *C++*", - пишет автор самого популярного языка программирования. - "*Язык C++* многим обязан языку *C*, и язык *C* остается подмножеством *языка C++* (но в C++ устранены несколько серьезных брешей системы типов C). Я также сохранил средства *C*, которые являются достаточно низкоуровневыми, чтобы справляться с самыми критическими системными задачами. *Язык C*, в свою очередь многим обязан своему предшественнику, BCPL; кстати, стиль комментариев // был взят в C++ из BCPL. Другим основным источником вдохновения был язык Simula67. Концепция классов (с производными классами и виртуальными функциями) была позаимствована из него. Средства перегрузки операторов и возможность помещения объявлений в любом месте, где может быть записана инструкция, напоминает Algol68. "

Название **C++** выдумал Рик Масситти. Название указывает на эволюционную природу перехода к нему от C. "++" - это операция приращения в C. Чуть более короткое имя C+ является синтаксической ошибкой; кроме того, оно уже было использовано как имя совсем другого языка. Знатоки семантики C находят, что *C++* хуже, чем ++C. Названия D язык не получил, поскольку он является расширением C и в нем не делается попыток исцеляться от проблем путем выбрасывания различных особенностей...

Изначально *язык программирования C++* был разработан, чтобы автору и его друзьям не приходилось программировать на ассемблере, C или других современных языках высокого уровня. Основным его предназначением было сделать написание хороших программ более простым и приятным для отдельного программиста. Плана разработки *C++* на бумаге никогда не было; проект, документация и реализация двигались одновременно. Разумеется, внешний интерфейс *C++* был написан на *C++*. Никогда не существовало "Проекта C++" и "Комитета по разработке C++". Поэтому *C++* развивался и продолжает развиваться во всех направлениях, чтобы справляться со сложностями, с которыми сталкиваются пользователи, а также в процессе дискуссий автора с его друзьями и коллегами.

В **языке С++** полностью поддерживаются принципы объектно-ориентированного программирования, включая три кита, на которых оно стоит: инкапсуляцию, наследование и полиморфизм. *Инкапсуляция в С++* поддерживается посредством создания нестандартных (пользовательских) типов данных, называемых классами. *Язык С++* поддерживает наследование. Это значит, что можно объявить новый тип данных (класс), который является расширением существующего.

Хотя **язык программирования С++** справедливо называют продолжением С и любая работоспособная программа на языке С будет поддерживаться *компилятором С++*, при переходе от С к С++ был сделан весьма существенный скачок. *Язык С++* выигрывал от своего родства с языком С в течение многих лет, поскольку многие программисты обнаружили, что для того, чтобы в полной мере воспользоваться преимуществами *языка С++*, им нужно отказаться от некоторых своих прежних знаний и приобрести новые, а именно: изучить новый способ концептуальности и решения проблем программирования. Перед тем как начинать осваивать *С++*, *Страуструп* и большинство других программистов, использующих *С++* считают изучение языка С необязательным.

*Язык программирования C++* в настоящее время считается господствующим языком, используемым для разработки коммерческих продуктов, 90% игр пишутся на *С++* с прменением DirectX.

**Обратная польская нотация (RPN)**

Обра́тная по́льская нота́ция (ОПН) — форма записи математических выражений, в которой операнды расположены перед знаками операций. Также именуется как обратная польская запись, обратная бесскобочная запись (ОБЗ), постфиксная нотация, бесскобочная символика Лукасевича, польская инверсная запись, ПОЛИЗ.

Обратная польская нотация была разработана австралийским философом и специалистом в области теории вычислительных машин Чарльзом Хэмблином в середине 1950-х на основе польской нотации, которая была предложена в 1920 году польским математиком Яном Лукасевичем. Работа Хэмблина была представлена на конференции в июне 1957, и издана в 1957 и 1962.

Первыми компьютерами, поддерживающими обратную польскую нотацию были KDF9 от English Electric Company, который был анонсирован в 1960 и выпущен (появился в продаже) в 1963, и американский Burroughs B5000, анонсирован в 1961, выпущен в том же 1963. Один из проектировщиков B5000, Р. С. Бартон, позже написал, что разработал обратную польскую запись независимо от Хэмблина, примерно в 1958, в процессе чтения книги по символьной логике, и до того как познакомился с работой Хэмблина.

Компания Friden перенесла ОПН в настольные калькуляторы, выпустив в июне 1964 модель EC-130. А в 1968 инженеры Hewlett-Packard разработали настольный калькулятор 9100A с поддержкой ОПН. Этот калькулятор сделал обратную польскую нотацию популярной среди учёных и инженеров, даже несмотря на то, что в ранней рекламе 9100A ОПН не упоминалась. В 1972 калькулятор HP-35 с поддержкой ОПН стал первым научным карманным калькулятором.

В 1971 году появился оригинальный язык программирования Forth, языковая машина которого имеет двухстековую структуру и где все вычисления проводятся на стеке. В этом языке ОПН является естественным способом записи любых операций с данными, хотя возможна, при желании, реализация и обычной (инфиксной) записи арифметических операций.

ОПН применялась в советском инженерном калькуляторе Б3-19М (совместная разработка с ГДР), выпущенном в 1976 году. Все выпускаемые в СССР вплоть до конца 1980-х годов программируемые микрокалькуляторы, за исключением «Электроника МК-85» и «Электроника МК-90» использовали ОПН — она проще реализовывалась и позволяла обойтись в программировании вычислений меньшим числом команд, по сравнению с обычной алгебраической нотацией, а количество программной памяти в этих моделях всегда было критическим ресурсом. ОПН используется в современных российских программируемых калькуляторах «Электроника МК-152» и «ЭЛЕКТРОНИКА МК-161», что обеспечивает их совместимость с программами, написанными для советских калькуляторов.

Основная часть

Обоснование выбора средств и методов для выполнения задания

Средством для выполнения задания была выбрана среда разработки VisualStudio 2013 от Microsoft. Приложение разработано под операционную систему Windows 8.1 (на других версиях Windows приложение не тестировалoсь). Предметной областью задания является изучение обратной польской нотации RPN

Для работы, в программе подключаются следующие заголовочные файлы (библиотеки):

**<iostream>**

Операции **ввода/вывода** выполняются с помощью классов istream (потоковый ввод) и ostream (потоковый вывод). Третий класс, iostream, является производным от них и поддерживает двунаправленный ввод/вывод. Для удобства в библиотеке определены три стандартных объекта-потока:

cin – объект класса istream, соответствующий стандартному вводу. В общем случае он позволяет читать данные с терминала пользователя;

cout – объект класса ostream, соответствующий стандартному выводу. В общем случае он позволяет выводить данные на терминал пользователя;

cerr – объект класса ostream, соответствующий стандартному выводу для ошибок. В этот поток мы направляем сообщения об ошибках программы.

Вывод осуществляется, как правило, с помощью перегруженного оператора сдвига влево (<<), а ввод – с помощью оператора сдвига вправо (>>)

<string>

string — заголовочный файл с классами, функциями и переменными для организации работы со строками в языке программирования C++. Он включён в стандартную библиотеку C++. Название образовано от имени строчного типа данных (англ. string). В языке C++ и его предшественнике, языке программирования Си, нет встроенной поддержки строкового типа данных, вместо этого используется массив символов. string управляет строками, как и string.h в Си. string использует единственный объект string для организации работы со строками. Являясь частью стандартной библиотеки C++, эти объекты также являются частью стандартного пространства имён — std.

Также, что не мало важно, он имеет встроенные конструкторы:

**string();** Создает пустую строку.

**string(const string& str);** Копирующий конструктор

**string(const string& str, size\_t pos, size\_t len = npos);** Копирующий конструктор со смещением и определенной длинной считываемой строки.

**string(const char\* s);** Конструктор принимающий указатель на строку символов.

**string(const char\* s, size\_t n);** Конструктор принимающий указатель на строку символов с ограничением по длине считываемой строки.

**string(size\_t n, char c);** Конструктор заполняющий одним символом n ячеек.

**template <class InputIterator> string(InputIterator first, InputIterator last);** Конструктор копирующий последовательность символов в диапазоне.

**string(initializer\_list<char> il);** Конструктор копирующий инициализирующий список.

**string(string&& str) noexcept;** Конструктор получающий содержимое строки.

Из них в программе используются копирующий конструктор со смещением и определенной длинной считываемой строки, а также конструктор принимающий указатель на строку символов.

<stack>

В этом заголовочном файле содержится один из типов списков - стек.

Стек — структура данных, представляющая собой список элементов, организованных по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»). Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно снять верхнюю.

Зачастую стек реализуется в виде однонаправленного списка (каждый элемент списка указывает только на следующий). Но в таком случае невозможно применить операцию обхода элементов. А доступ возможен только к верхнему элементу структуры. Для обхода такой проблемы можно взять за основу двусвязный список (каждый элемент указывает на обоих соседей справа и слева). Кроме того, можно организовать его на обыкновенном массиве.

Значением переменной стека является указатель на вершину стека. Если стек пуст, то значение указателя равно NULL.

Пример реализации стека на языке С:

struct stack

{

char \*data;

struct stack \*next;

};

Возможны три операции со стеком: добавление элемента (иначе проталкивание, push), удаление элемента (pop) и чтение головного элемента top().

При проталкивании (push) указывается новый элемент, указывающий на элемент, бывший до этого головой. Новый элемент теперь становится головным.

При удалении элемента убирается первый, а головным становится тот, на который был указатель у этого объекта (следующий элемент).

Например вот, коды реализации ф-ций добавления/удаления в/из стека :

void push( STACK \*ps, int x ) // Добавление в стек нового элемента

{

if ( ps->size == STACKSIZE ) // Не переполнен ли стек?

{

fputs( "Error: stack overflow\n", stderr );

abort();

}

else {

ps->items[ps->size++] = x;

}

}

int pop( STACK \*ps ) // Удаление из стека

{

if ( ps->size == 0 ) // Не опустел ли стек?

{

fputs( "Error: stack underflow\n", stderr );

abort();

}

else {

return ps->items[--ps->size];

}

}

В программе стеки используются очень активно (Обратная польская нотация строится именно на стеках).

<vector>

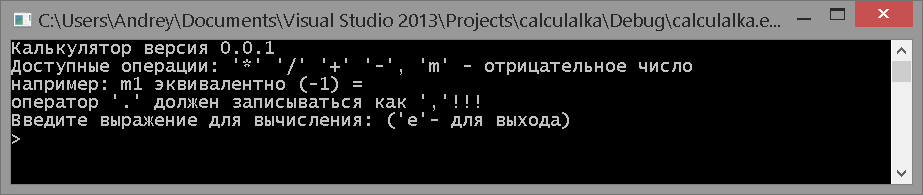
Шаблон vector расположен в заголовочном файле <vector>. Как и все стандартные компоненты, он расположен в пространстве имён std. Данный интерфейс эмулирует работу стандартного массива C (например, быстрый произвольный доступ к элементам), а также некоторые дополнительные возможности, вроде автоматического изменения размера вектора при вставке или удалении элементов.

Все элементы вектора должны принадлежать одному типу. Например, нельзя совместно хранить данные типов char и int в одном экземпляре вектора. Класс vector обладает стандартным набором методов для доступа к элементам, добавления и удаления элементов, а также получения количества хранимых элементов.

Класс vector — это контейнер. Согласно стандарту C++, любой контейнер должен содержать методы begin(), end(), size(), max\_size(), empty(), и swap(). (приводить коды ф-ций я не буду, так как их очень много).

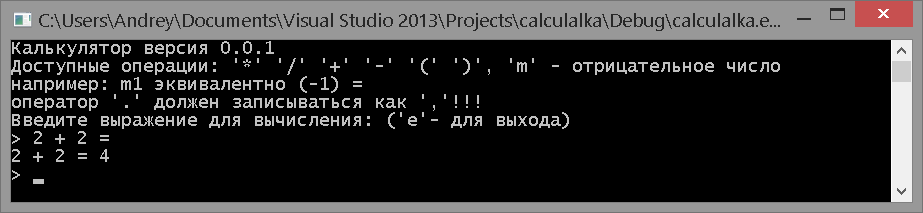
**Интерфейс командной строки.**

Приложение содержит текстовое меню выбора действий и, в некоторых случаях, подменю. Выбор действия осуществляется вводом с клавиатуры выражения в инфиксной форме. После запуска приложения перед нами открывается основное меню .

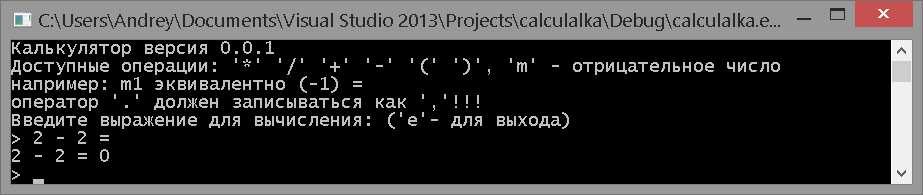


Всего программа - калькулятор поддерживает 4 математические операции:

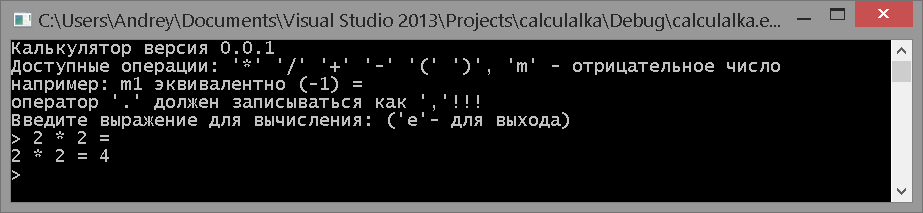
1) Сложение (+), используется для нахождения суммы чисел.



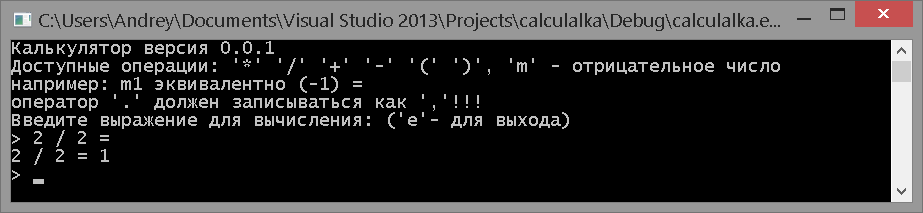
2) Вычитание (-), используется для нахождения разности чисел.



3)Умножение (\*), используется для нахождения произведения чисел.



4)Деление(/), используется для нахождения деления чисел.



Вычисления происходях циклически, поэтому для выхода из программы надо ввести 'e' (eng).

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Помимо математических операций, калькулятор поддерживает операционные скобки '(' ')'.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Отдельно хочу отметить, что отрицательные числа, записываются в виде символа 'm' и соответствующей цифре. Например m4 эквивалентно записи числа -4.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

А также для записи чисел с плавающей запятой нужно использовать следующую форму записи:

целая часть числа - **запятая!(,) -** дробная часть

Обязательно использование запятой, так как в противном случае, появятся ошибки в работе строк.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Теперь перейдем к самой интересной части проекта, его реализации...

**Реализация обратной польской нотации на примере приложения «Калькулятор»**

Инфиксная нотация — операторы записываются между операндами с которыми они взаимодействуют, а в постфиксной нотации (обратная польская запись) операторы записаны после операндов.

**Перевод выражений из инфиксной нотации в постфиксную**

Для так называемого 'парсинга', т.е перевода выражения из одной формы в другую, мне понадобились следующие 'контейнеры' данных:

1) Стек operation для временного хранения операторов.

2) Строка infN для хранения выражения в инфиксной нотации, принятой из потока ввода (cin) в ф-ции main().

3) Переменная last, типа char, для хранения последнего символа оператора, отправленного в operation.

4) Второй стек postN для хранения полученного выражения в форме обратной польской нотации.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

При работе парсера используется набор правил, соответствующий следующей таблице(также она приведена в коде программы):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **=** | **+** | **-** | **\*** | **/** | **(** | **)** |
| **=** | 4 | 1 | 1 | 1 | 1 | 1 | 5 |
| **+** | 2 | 2 | 2 | 1 | 1 | 1 | 2 |
| **-** | 2 | 2 | 2 | 1 | 1 | 1 | 2 |
| **\*** | 2 | 2 | 2 | 2 | 2 | 1 | 2 |
| **/** | 2 | 2 | 2 | 2 | 2 | 1 | 2 |
| **(** | 5 | 1 | 1 | 1 | 1 | 1 | 3 |

Для того чтобы узнать, что делать, надо действовать в соответствие с вышестоящей таблицей, а именно:

1) символ отправляется в operation

2) последний символ, направившийся в стек, направляется в postN

3) текущий символ и предыдущий символ, отправившийся в Operation, удаляются

4) Остановка. Символы, находящиеся в postN, представляют собой формулу в обратной польской записи, если читать слева направо

5) Остановка. Произошла ошибка. Изначальная формула была некорректно сбалансирована

**Пример работы алгоритма:**

выражение: (1+2)\*4 =

1) смотрим в таблицу, так символ ( , отправляем его в operation

2) 1 - число, без раздумий в postN

3) + - проверяем: предыдущий символ ( => добавляем + в operation

4) 2 - число, без раздумий в postN

5) ) - проверяем: предыдущий символ + => переносим символ + из operation в postN и удаляем все скобки ( operation пуст)

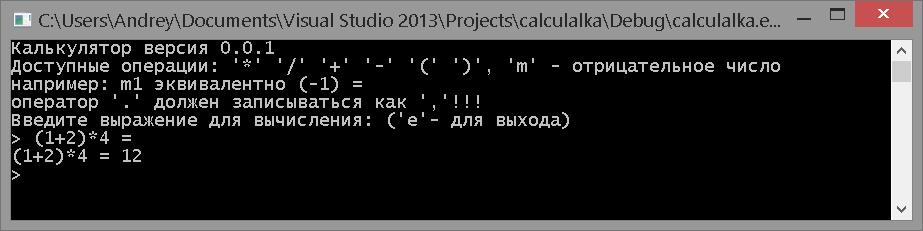
6) \* - отправляем в operation

7) 4 - число, без раздумий в postN

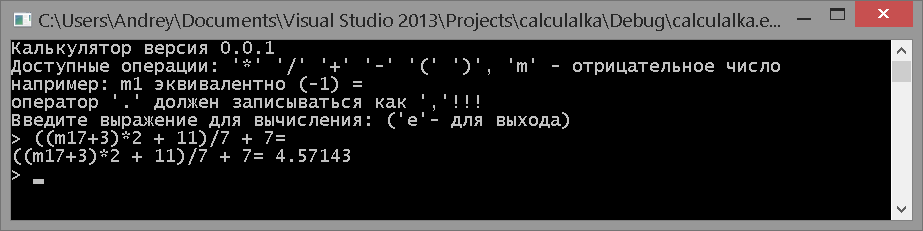
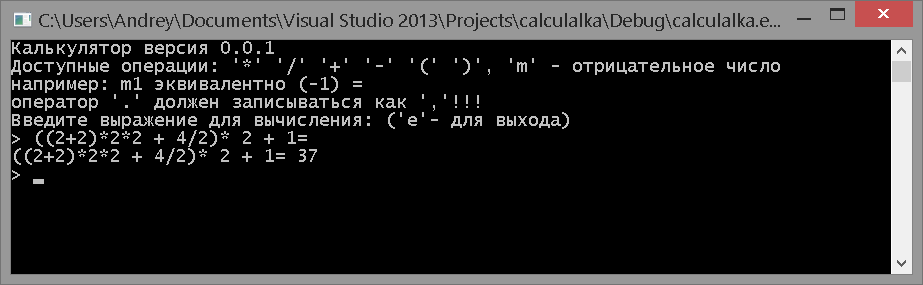
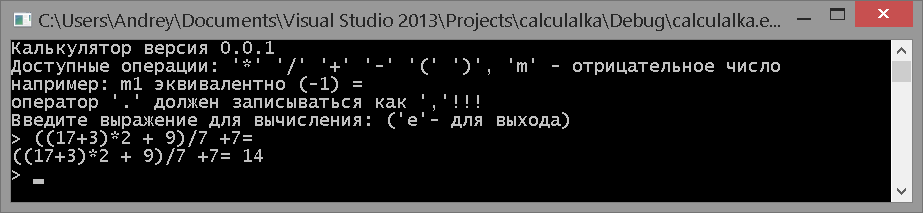
8) = - все математические операторы из operation переносим в postN

Таким образом мы получаем, что (1+2)\*4 = 1 2 + 4 \* в обратной польской нотации.

В консоли это выглядит так:



Для иллюстрации работы калькулятора возьмем какие-нибудь сложные выражения:



**Заключение**

В процессе курсовой работы проводилась разработка алгоритмов, тестирование и отладка, как самого приложения, так и отдельных его модулей. (Ну и 'подгорало' у меня, скажу вам.)

В ходе разработки выяснилось, что метод вычисления через обратную польскую нотацию, очень сложен в создании (у меня, как у не очень внимательного товарища это вызывало чувство бесконечной ярости), но удобен для добавления дополнительных математических операторов. Если сравнить его с более быстродейственным методом решения математических выражений, а именно с рекурсивным спуском, то я бы выбрал рекурсивный спуск, так как он не только быстрее, но и гораздо проще в реализации.

В общем этот проект помог мне закрепить и улучшить, как общие знания по программированию на языке C++, так и по работе со стеками и векторами, которые использовались в данном проекте.

**Список литературы**

1. Бьерн Страуструп "Язык программирования C++. 3-е издание.
2. Дейтел Х. Как программировать на C++/ Х.Дейтел, П.Дейтел. – М. Бином-Пресс, 2014 – 1200 с.
3. Крупник А. Изучаем C++/ СПб.: Питер, 2003 – 256 с.

**Приложение**

фрагмент парсера:

char calculate::parce()

{

for (int i = 0; i < infN.size(); i++) {

int size;

string s;

switch (infN[i]) {

case 'm':

s = numberM(i); // ф-ция возвращает отрицательное число

postN.push(s); // добавляем это число в стек

break; // выходим из цикла

case '+':

if (operation.empty() || last == '(') {

operation.push(infN[i]); // добавляем + в operation

last = infN[i]; // запоминаем последний символ

break;

} else if (last == '+' || last == '-' || last == '\*' || last == '/') {

str(last, infN, i); // предыдущий символ из operation отправляется в стек

break;

}

...

}

Ф-ция вычисления выражения в обратной польской нотации:

void calculate::calc()

{

double a, b;

string c;

while(postN2.size() >= 1) {

if (postN2.back() == "+") {

a = numb.top();

numb.pop();

b = numb.top();

numb.pop();

numb.push(a + b);

postN2.pop\_back();

} else if (postN2.back() == "-") {

a = numb.top();

numb.pop();

b = numb.top();

numb.pop();

numb.push(b - a);

postN2.pop\_back();

} else if (postN2.back() == "/") {

a = numb.top();

numb.pop();

b = numb.top();

numb.pop();

if (a != 0) {

numb.push(b / a);

postN2.pop\_back();

} else {

error();

}

} else if (postN2.back() == "\*") {

a = numb.top();

numb.pop();

b = numb.top();

numb.pop();

numb.push(a \* b);

postN2.pop\_back();

} else {

numb.push(atof(postN2.back().c\_str()));

postN2.pop\_back();

}

}

print(); // ф-ция вывода результата

}