Содержание

***Введение …………………………………………………………………………………………………………………………………………………………………………………3***

1. ***Постановка задачи …………………………………………………………………………………………………………………………………….……………..5***
2. ***Требования к программному средству ………………………………………………………………………………..……………….…………6***
3. ***Выбор метода решения поставленной задачи …………………………………………………..………………………….…….………7***
4. ***Программная реализация ………………………………………………………………………………………………………….…..………………..……..8***
   1. *Выбор среды разработки …………………………………………………………****…………………………****…………………………………………8*
   2. *Построение общей структуры программы …………………………………………………………………………………..………9*
   3. *Описание структур, типов данных и глобальных переменных ………….………………………….…………10*
   4. *Описание основных подпрограмм ………………………………………………………………………………………………..……………13*
5. ***Тестирование и отладка программного средства …………****………………………………****………………………….…………15*** 
   1. *Тестирование и верификациям ……………………………………………………………………………………….…………….…………15*
   2. *Показатели качества программы ……………………………………………………………….………………………………………….18*

***Заключение ………………………………………………………………………………………****…………………………………………****……****……………****……………………20***

Содержание

***Введение***

*Большинство хороших программистов делают свою работу не потому, что ожидают оплаты или признания, а потому что получают удовольствие от программирования.*

*— Linus Torvalds*

Строго говоря, калькуляторы были изобретены сразу после того, как человек научился считать. Древнейший артефакт такого рода — «кость Ишанго», найденная в Конго (возраст — около двадцати тысяч лет). Это берцовая кость бабуина, покрытая засечками

Простейший счет велся на пальцах, а когда их не хватало, использовались любые природные объекты, заменявшие цифру 10. Примерно пять тысяч лет назад в Вавилоне появилась счетная доска, известная ныне как абак (абакус). По полю с углублениями передвигались камушки (десятки). Вероятно, это был инструмент купцов.

Антикитерский механизм — самое передовое ручное механическое устройство древнего мира. В 2008 году было доказано, что антикитерский механизм может выполнять четыре арифметические операции (сложение, вычитание, умножение и деление).

Следующий шаг сделали китайцы, создавшие в шестом-двенадцатом веках нашей эры суньпан, известный сегодня как счеты. Большая секция костяшек называлась «земля», а малая наверху — «небо». Техника работы с суньпаном позволяла быстро производить все четыре арифметические операции, а также извлекать корни.

Юпана, калькулятор майя. Ученые долго не могли понять предназначение этой маленькой «модели крепости» до тех пор, пока итальянский инженер Николино де Паскуале не установил, что так называемые «дикари» создали матрицу этого калькулятора с использованием последовательности Фибоначчи и системы исчисления с основанием 40 (а не 10, как в Старом Свете).

Введение

Логарифмическая линейка — главный инструмент инженера до восьмидесятых годов прошлого века — была изобретена в 1622 году. Ее действие основано на том, что умножение и деление чисел можно выполнить сложением и вычитанием их логарифмов. С помощью такой линейки можно выполнить очень сложные вычисления с точностью до 3-4 десятичных знаков. Первый полет человека в космос рассчитывался именно на таких линейках.

«Паскалина» Блеза Паскаля хорошо известна. Она была построена в 1642 году. Однако это гениальное устройство десятеричного исчисления не получило признания, поскольку во Франции ходили ливры, равные 20 су, а су равнялся 12 денье.

Не менее известна и «разностная машина» Чарльза Бэббиджа, фигурировавшая в одноименном романе Стерлинга и Гибсона. Она была спроектирована в 1822 году и, будучи построенной, могла бы вычислять многочлены с точностью до восемнадцати знаков после запятой.

К концу 19 века стали популярны арифмометры — компактные механические вычислители. По легенде в 1924 году их производство было налажено в СССР по приказу Дзержинского. Такие «калькуляторы» выпускались у нас до 1970-х под маркой «Феликс».

Самым компактным в истории механическим калькулятором был «Курта» (1938). Он выпускался до 1970-х.

А вот и «убийца» счетов — первый в мире полностью электронный компактный калькулятор Anita (компания Bell, 1961 год). С него началась эра всеобщей калькуляторизации. Калькуляторы вставлялись в столы, портмоне, наручные часы, а с недавних пор всех их вытеснил калькулятор Windows.

Введение

***Введение***

***1. Постановка задачи***

*Необходимо написать программу, выполняющую функции калькулятора.*

Калькулятор должен включать в себя основные компоненты для работы с введенным числом и осуществлением основных логических и математических операций над ним. Также необходимо осуществить расширенные возможности изменения системы счисления, длины и типа (знаковый/без знаковый) числа.

***1. Постановка задачи***

2 Требования к программному средству

В ходе реализации программы были поставлены некоторые требования, которым

она должна отвечать:

1. Выполнять ряд определенных функций :

* Иметь:
  + цифровой блок для ввода чисел;
  + блок арифметических и логических операций (умножение, деление (целочисленное, остаток), сложение, вычитание; бинарные «и», «или», «исключающее или», «не», поразрядные сдвиги вправо и влево (циклические, нециклические), смена знака);
  + блок работы с памятью (сохранение, извлечение из памяти, суммирование и вычитание отображаемого числа и числа в памяти, очистка памяти);
  + блок обработки ввода/вывода (полная очистка, очистка последнего введенного числа, удаление последней введенной цифры).
* Предоставлять возможность:
  + изменения размера введенного (результативного) числа (1 байт, 2 байта, 4 байта, 8 байт);
  + изменения системы счисления (двоичная, восьмеричная, десятичная, шестнадцатеричная)
  + представления числа в знаковой либо без знаковой форме.
* Поддерживать переключение между дополнительным, прямым и обратным кодами числа для двоичной системы; также осуществлять возможность инвертирования бита после клика на соответствующем элементе в специальном поле;
* Соблюдать приоритет операций;
* Осуществлять работу (обмен данными) с буфером обмена.

2. Программа должна быть направлена на простого пользователя :

* Иметь интуитивный и всегда понятный интерфейс.
* Включать в себя руководство пользователя.
* Сведения об авторе.

2 Требования к программному средству

3. Выбор метода решения поставленной задачи

Главными компонентами программы являются главное поле вывода, поле для вывода двоичного кода отображаемого числа и поле ввода.

После передачи данных программе (через основной блок ввода или буфер обмена) эти данные проходит своеобразный фильтр, где полученное число проверяется на «переполнение» разрядной сетки и присутствие знакового бита (для буфера обмена данные дополнительно проходят проверку на корректность). Этот же «фильтр» используется для проверки результатов вычислений (после каждой операции).

Поддержка приоритета операций достигается путем разделения всех операций на три части:

1. Сверхприоритет — not, сдвиги, операция смены знака (выполняются непосредственно сразу после нажатия клавиши)
2. Высокоприоритетные — \*, div, mod, and (выполняются первыми в «очереди»)
3. Низкоприоритетные — +, -, or, xor (выполняются последними в «очереди»)

3. Выбор метода решения поставленной задачи

4. Программная реализация

4.1 Выбор среды разработки

В качестве языка разработки был выбран Delphi, о преимуществах которого

мы сейчас и поговорим :

1. Имеются множественные пересечения синтаксиса с Pascal ABC.Net.
2. Содержит удобный интерфейс.
3. Легкость подключения элементов а так же их настройка.
4. Автоматическое форматирование кода.
5. При написании программы возможно автоматическое написание необходимого текста.
6. Возможность создания мультимедийных проектов.
7. При нахождении ошибки в процессе отладки программы высвечивается соответствующее пояснение

4. Программная реализация

4.2 Построение общей структуры программы

В общем случае работу типичной программы можно рассмотреть как простейшую последовательность «Ввод – Обработка – Вывод». Но для каждой задачи «обработка» несет в себе разный смысл. В нашем случае (рис. 1):

**Рис. 1.** Общая схема обработки сообщений

Вывод

Проверка на переполнение

Действия

Проверка на переполнение

Ввод

Проверка на корректность

Клавиши

Буфер обмена

Изменение битов

4. Программная реализация

4.3. Описание структур, типов данных и глобальных переменных

В ходе реализации поставленной задачи появилась необходимость в использовании ряда переменных определенного типа и объявлении новых типов:

type

Operations = (op\_div, op\_mod, op\_and, op\_multi, op\_plus, op\_minus, op\_or, op\_xor, op\_Null);

LastOperationType = (FPOper, SPOper, Null);

TypeOfRefresh = (RefreshOnly, DoAll);

Form1: TForm1; // Собственно форма

IniFile:TIniFile;

FormHeight, FormWidth : integer;

b\_SSNow : byte;

i\_Input, i\_Memory : int64;

CanNullInput, b\_MemoryUsed, b\_CEEnabled, b\_NullInp : boolean;

lbin : array [1 .. 4, 1 .. 16] of TLabel;

bt\_SizeOfWord, bt\_PrevSizeOfWord, bt\_LabelOutputType : byte;

HighOfWord, LowOfWord: int64;

PrevHighOfWord, PrevLowOfWord: int64;

i\_Result, i\_LastFPResult, i\_LastInputed : int64;

op\_LastFPOper, op\_LastSPOper, op\_PrevOper, op\_InputedOper : Operations;

b\_TypeIsChangedFirst, b\_OperClicked: Boolean;

b\_SignWithMinus: boolean;

b\_FirstOper : boolean;

b\_ClearLwithMC : Boolean;

b\_MCwithCE : boolean;

Опишем каждую из них подробнее:

4. Программная реализация

Типы:

Operations – здесь хранятся операции (их перечисление);

LastOperationType – хранит значения типов операций (приоритетов);

TypeOfRefresh - тип обновления (передается функции в качестве параметра);

Переменные:

IniFile – необходим для работы с ini-файлами;

FormHeight, FormWidth – для хранения размеров окна;

b\_SSNow – хранит текущую систему счисления;

i\_Input – для хранения введенного числа в десятичной системе счисления,

дополнительном коде;

i\_Memory – текущее значение памяти;

CanNullInput – хранит информацию о том, можем ли мы обнулить переменную i\_Input;

b\_MemoryUsed – используется ли память;

b\_CEEnabled – можем ли все очистить;

b\_NullInp – принудительное обнуление ввода;

lbin – массив под биты двоичного кода:

bt\_SizeOfWord – текущий размер числа (в байтах);

bt\_PrevSizeOfWord – предыдущий размер числа;

bt\_LabelOutputType – тип кода (для вывода в массив (lbin)

HighOfWord, LowOfWord – наибольшее и наименьшее числа текущего размера;

PrevHighOfWord, PrevLowOfWord – предыдущие наибольшее и наименьшее числа

текущего размера;

i\_Result – число, получающееся в результате математических и логических

операций;

i\_LastFPResult – предыдущий результат высокоприоритетной последовательности;

4. Программная реализация

i\_LastInputed – последнее введенное число;

op\_LastFPOper – последняя высокоприоритетная операция;

op\_LastSPOper – последняя низкоприоритетная операция;

op\_PrevOper – предыдущая операция;

op\_InputedOper – введенная операция;

b\_TypeIsChangedFirst – для определения начала ввода;

b\_OperClicked нажата клавиша операции;

b\_SignWithMinus – число с «минусом»;

b\_FirstOper – нажатая операция - первая;

b\_ClearLwithMC – очистка главного вывода при нажатии МС (в случае если число на

выводе и число в памяти одинаковы);

b\_MCwithCE – очищать память по нажатию СЕ

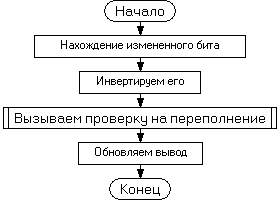
4. Программная реализация

4.4 Описание основных подпрограмм

**1.** procedure TForm1.gbBinCodeMouseActivate(Sender: TObject; Button: TMouseButton; Shift: TShiftState;

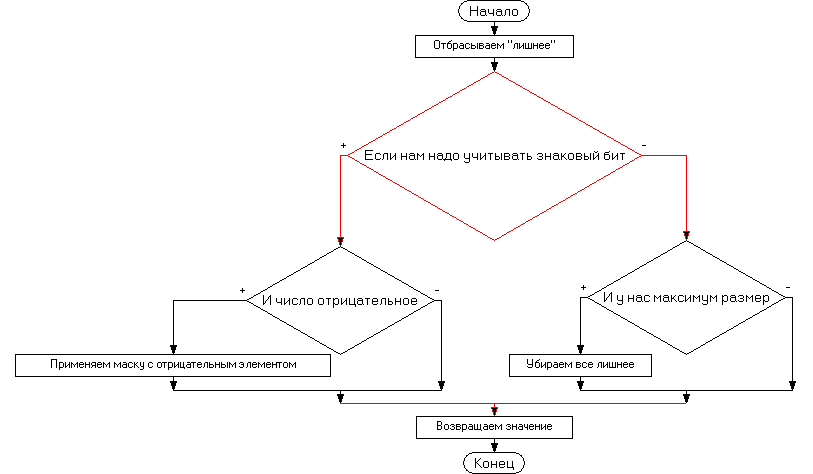
X, Y, HitTest: Integer; var MouseActivate: TMouseActivate);

Данная подпрограмма вызывается при нажатии на один из «битов» на панели двоичного кода.



**2.** function TForm1.OverflowMask(inp:int64):int64;

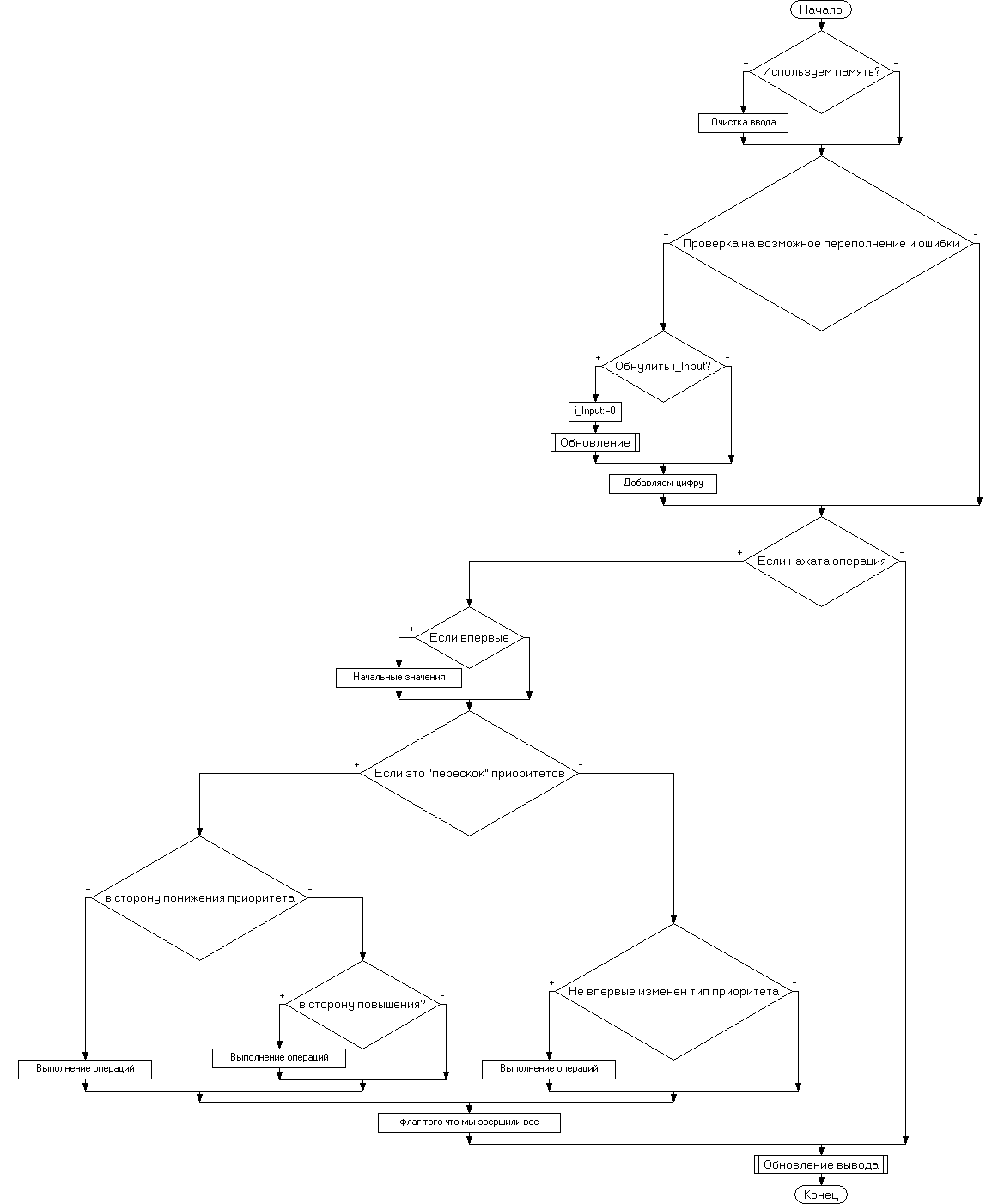
Проверка на переполнение



4. Программная реализация

**3.** procedure TForm1.AfterNumberClick(num: byte);

Обработчик нажатия цифрового блока



4. Программная реализация

5. Тестирование и отладка программного средства

5.1. Тестирование и верификация.

Отладка и верификация программы представляют собой очень важную часть процесса разработки программы. Отладка заключается в устранении ошибок программирования, ошибок перевода алгоритма на язык программирования. Отладка – это деятельность, направленная на обнаружение и исправление ошибок в программе с использованием процессов выполнения его программ. Тестирование программы – это процесс выполнения его программ на некотором наборе данных, для которого заранее известен результат применения или известны правила поведения этих программ. Указанный набор данных называется тестовым или просто тестом. Таким образом, отладку можно представить в виде многократного повторения трех процессов: тестирования, в результате которого может быть констатировано наличие в ПС ошибки, поиска места ошибки в программах и документации ПС и редактирования программ и документации с целью устранения обнаруженной ошибки. Другими словами:

Отладка = Тестирование + Поиск ошибок + Редактирование.

Верификация – это доказательство того, что программа работает «правильно», дает правильный результат. Для этого разрабатывается система тестов, которые могут представлять собой специально подобранные наборы параметров, для которых задача может быть решена точно. Это могут быть, например, какие-нибудь предельные случаи. Если результат, полученный с помощью программы, совпадает (с учетом погрешности машинного кода) с ожидаемым, есть основание полагать, что программа работает правильно. Для доказательства правильности работы программы создадим систему тестов, которая учитывает все предельные случаи и все ограничения и требования, предъявляемые к поставленной задаче.

5. Тестирование и отладка программного средства

Тест №1 Выполнение простейших операций с учетом приоритета операции:

Эталонный калькулятор - MSWindows 7 (calc.exe)

Подадим на вход программе (путем нажатий клавиш на цифровом блоке) пример:

21 + 37 – 42 \* 54 \* 12 + 35 +not(3) =

Этот тест обеспечивает проверку работы приоритета операций, так как в нем находится весь «набор» переходов между операциями.

* Расчет на калькуляторе MSWindows 7 = -27123
* Расчет на калькуляторе (курсовой) = -27123

Тест №2 Корректное инвертирование битов на разных режимах

Эталонный калькулятор - MSWindows 7 (calc.exe)

Проверка

1. Наберем в калькуляторе – 3 or 9[инвертируем младший бит] =

Этот тест обеспечивает проверку на верное считывание с панели двоичного кода во время совершения расчетов.

* Расчет на калькуляторе MSWindows 7 = 11
* Расчет на калькуляторе (курсовой) = 11

2. Поставим единичный бит в поле двоичного ввода на местах 2, 15, 31, 63 и будем уменьшать длину числа пошагово до 1 байта (в калькуляторе курсовой работы стоит режим «со знаком»)

Этот тест обеспечивает проверку на верное определение знакового бита

* Расчет на калькуляторе MSWindows 7 = -9223372034707259388, -2147450876,

-32764. 4

* Расчет на калькуляторе (курсовой) = -9223372034707259388, -2147450876,

-32764. 4

5. Тестирование и отладка программного средства

3. Введем любое отрицательное число и проверим его коды.

Наберем -99658

Дополнительный код = 1111111111111111111111111111111111111111111111100111101010110110

Обратный код = 1111111111111111111111111111111111111111111111100111101010110101

Прямой код = 100000000000000000000000000000000000000000000011000010101001010

Этот тест помогает проверить верность отображения дополнительного, обратного и прямого кодов

5. Тестирование и отладка программного средства

5.2. Показатели качества программы

Разработанное программное средство позволяет выполнять определенные

функции, следуя целям поставленной задачи. Однако хорошее программное средство

должно обладать еще целым рядом свойств, позволяющим успешно его использовать

в течение длительного периода, т.е. обладать определенным качеством. Качество

программного средства (ПС)- это совокупность его черт и характеристик, которые

влияют на его способность удовлетворять заданные потребности пользователей.

В настоящее время критериями качества ПС принято считать:

* функциональность;
* надежность;
* легкость применения;
* эффективность;
* сопровождаемость;
* мобильность.

Функциональность – это способность ПС выполнять набор функций,

удовлетворяющих заданным или подразумеваемым потребностям пользователя.

Легкость применения включает в себя возможность использования данной

программы, не зная курса информатики. Так же большую помощь при использовании

данного ПС оказывает справка, в которой написано подробное руководство по

применению и разобран алгоритм решения на нескольких примерах.

Сопровождаемость - это характеристики ПС, которые позволяют минимизировать

усилия по внесению изменений для устранения в нем ошибок и по его модификации в

соответствии с изменяющимися потребностями пользователей.

5. Тестирование и отладка программного средства

Мобильность ПС подразумевает возможность переноса данной программы с

одного компьютера на другой, использование ПС в любой ОС Windows, а так же не

требовательность к ресурсам компьютера.

Функциональность и надежность являются обязательными критериями качества

ПС. Остальные критерии используются в зависимости от потребностей

пользователей в соответствии с требованиями к ПС.

Обеспечение сопровождаемости ПС сводится к обеспечению изучаемости ПС и к

обеспечению его модифицируемости.

Изучаемость (подкритерий качества) ПС определяется составом и качеством

документации по сопровождению ПС и выражается через такие примитивы качества

ПС как информативность, понятность, структурированность и удобочитаемость.

Последние два примитива качества и, в значительной степени, понятность связаны

с текстами программных модулей.

Для обеспечения изучаемости ПС и его модифицируемости, при окончательном

оформлении текста программного продукта применялись следующие решения:

* использование в тексте программы комментариев, проясняющих особенности принимаемых решений;
* использование подпрограмм для удобочитаемости и оптимизации программного кода;
* отделение строк кода друг от друга пустой строкой для повышения читаемости и восприятия кода;
* составление инструкции для пользования данным программным средством.

5. Тестирование и отладка программного средства

Заключение

В ходе выполнения курсовой работы была написана программа «Калькулятор», удовлетворяющая ряду требований.

При написании программы широко применялось использование подпрограмм, принципов ООП.

В ходе работы были приобретены как практические, так и теоретические навыки по проектированию программного средства. Среди них, те задачи, с которыми приходится сталкиваться профессиональному программисту. И, следовательно, полученный в результате данной работы опыт может быть использован и для серьезной, профессиональной работы по разработке программ.

Заключение