Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра інформатики

КУРСОВА РОБОТА

Тема: «Розробка калькулятора великих чисел»

з дисципліни «Програмування»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Керівник \_\_\_\_\_\_\_\_\_\_\_ Руденко Д. О. (підпис) (прізвище, ініціали)

\_\_\_\_\_\_\_\_\_\_\_ Яковлева О.В. (підпис) (прізвище, ініціали)

\_\_\_\_\_\_\_\_\_\_\_ Кобилін О. А. (підпис) (прізвище, ініціали)

Студент гр. \_\_\_\_\_\_\_\_\_\_\_ Нальотов Д.Р. КІУКІ-20-2 (підпис) (прізвище, ініціали)

Харків

2021

РЕФЕРАТ

Записка пояснювальна до курсової роботи: 26 с., 4 рис., 5 розділів, 2 додатки.

Мета роботи — розробка калькулятора великих чисел.

Метод вирішення задачі - Скласти програму, що дозволяє виконувати арифметичні операції (додавання, різниця, множення, цілочисельне ділення, знаходження залишку) і операції порівняння (більше, менше, дорівнює, не більше, не дорівнює і т.д.) над великими числами

Розроблено програму, яка працює з файлом, і з її допомогою можна працювати з виразами та арифметичними діями.

Програму складено мовою C++ у середовищі програмування Visual Studio C++.

КАЛЬКУЛЯТОР, АРИФМЕТИЧНІ ВИРАЗИ, АРИФМЕТИЧНІ ОПЕРАЦІЇ, ВЕЛИКІ ЧИСЛА, КАЛЬКУЛЯТОР ВЕЛИКИХ ЧИСЕЛ

ЗМІСТ

|  |  |
| --- | --- |
| Вступ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 4 |
| 1 Постановка задачі . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 5 |
| 2 Теоретична частина. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 6 |
| 2.1 Загальні відомості. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 6 |
| 3 Вибір мови програмування та середовища розробки . . . . . . . . . . . . . . . | 8 |
| 3.1 Вибір мови програмування та середовища розробки. . . . | 8 |
| 3.2 Бібліотеки, які використовуються при написанні програми. . . | 8 |
| 4 Програмна реалізація. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  5 Інструкція користувача . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 10  11 |
| Висновки. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
| Перелік джерел посилання . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 13 |
| Додаток А. Текст програми. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 14 |
| Додаток Б. Результати роботи програми. . . . . . . . . . . . . . . . . . . . . . . . . . . | 25 |

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

ВСТУП

Метою курсового проекту є створення програми "Калькулятор", яка і буде об'єктом дослідження.

Калькулятор - пристрій для арифметичних обчислень. Вчені, інженери та статистики користуються іншими калькуляторами, здатними виконувати складні операції. Сучасні калькулятори - це електронні прилади з маленькими кремнієвими мікросхемами, що виробляють будь-які обчислення. Різновид калькулятора - касовий апарат. Він підсумовує ціни і робить роздруківку чека. Більшість касових апаратів автоматично зчитують ціни по бар коду, зазначеного на кожен товар. "Мозок" калькулятора може тільки додавати і віднімати. Множення і ділення він виконує шляхом багаторазового складання або віднімання.

Звичайний калькулятор виконує додавання і віднімання, множення і ділення, а також обчислює відсотки. Будь-яке число, яке вводиться в калькулятор, перетворюється в бінарний код, а результати в бінарному коді переводяться назад в десяткову систему і відображаються на дисплеї.

У наші часи існує безліч мов програмування. Програма, що працює на комп'ютері, нерідко ототожнюється з самим комп'ютером, так як людина, що використовує програму, вводить в комп'ютер вихідні дані з клавіатури, і комп'ютер видає результат на екран. Насправді перетворення вихідних даних виконує процесор комп'ютера. Процесор виконує перетворення відповідно до послідовності команд - програмою.

Таким чином, щоб комп'ютер виконав певну роботу, необхідно розробити послідовність команд, що забезпечує виконання цієї роботи, або, як кажуть, написати програму.

1 ПОСТАВНОВКА ЗАДАЧІ

Метою даної курсової роботи є розробити «калькулятор», що виконує зазначені в варіанті операції. Програма повинна виконувати введення даних, перевірку правильності введених даних, видачу повідомлень в разі помилок, арифметичні операції (додавання, різниця, множення, цілочисельне ділення, знаходження залишку) і операції порівняння (більше, менше, дорівнює, не більше, не дорівнює і т.д.) над великими числами. Під час створення програми необхідно виконати тестові перевірки, відмітити нормальні і аномальні результати, зробити аналіз результатів.

2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Загальні відомості

Калькулятор (лат. Calculator «лічильник») - електронне обчислювальний пристрій для виконання операцій над числами або алгебраїчними формулами.

Калькулятор замінив ручні (механічні) обчислювальні пристрої і пристосування, такі як абаки, рахунки, математичні таблиці (перш за все - таблиці логарифмів), логарифмічні лінійки, механічні або електромеханічні арифмометри.

Залежно від можливостей і цільової сфери застосування калькулятори діляться на найпростіші, бухгалтерські, інженерні (наукові), фінансові. В окремі класи зазвичай виділяють програмовані калькулятори, що дають можливість виконання складних обчислень по попередньо закладеною програмою, а також графічні - підтримують побудову та відображення графіків. Спеціалізовані калькулятори призначені для виконання обчислень в досить вузькій сфері (статистичні, медичні, спеціальні фінансові розрахунки і т. П.); такі калькулятори зараз частіше реалізуються у вигляді програм для універсальних персональних комп'ютерів, КПК, планшетів, хоча можуть виготовлятися і «в залізі».

За виконання калькулятори можуть бути настільними або компактними (кишеньковими). Окремі моделі мають інтерфейси для підключення персонального комп'ютера, друкувального пристрою, зовнішнього модуля пам'яті або інших зовнішніх пристроїв. Існують калькулятори, вбудовані в персональні комп'ютери, мобільні телефони, КПК і навіть наручний годинник. Спеціалізовані калькулятори вбудовуються в побутову техніку (наприклад, простий медичний калькулятор може вбудовуватися в спортивний тренажер).

Калькулятор можна використовувати для виконання простих операцій: додавання, віднімання, множення і ділення. У калькуляторі також передбачені можливості для виконання програмованих, інженерних і статистичних обчислень. Можна робити обчислення, натискаючи на кнопки калькулятора або вводячи символи з клавіатури. Програма написана на об`єктно-орієнтованій системі програмування MS Visual С++. Основою цієї мови, як можна побачити з назви є язик програмування С++.Головною причиною мого вибору стало те, що ця система обєктно-орієнтована і програмувати інтерфейс користувача на ній досить легко. В MS Visual С++ присутні багато різних компонентів, які надають широкий вибір при проектуванні, що дозволяє найкращим чином налагодити програму під поставлену задачу. Вбудований потужний довідник (MSDN) та різні програми, які допомагають програмісту при написані коду програми. Розробка та опис алгоритму розвязку задачі Алгоритмом називається директива, яка визначає порядок виконання дій (операцій) над даними з метою отримання шуканого результату. Процес підготовки розвязку задачі на ЕОМ називається алгоритмізацією. Розробка алгоритму складається з декількох етапів: Вивчення завдання даного для алгоритму задачі. Часто завдання представлене в описовій формі з використанням формул, таблиць, графіків. Необхідно глибоко вивчити процес, що алгоритмізується, виявити закономірності явищ, що його складають. Визначається вхідна вихідна інформація, задаються області зміни аргументів, точність обчислень. Вхідна інформація повинна бути повною. Виконується математична формалізація описової умови задачі. Її мета - побудувати масиви арифметичних і логічних операторів. До масиву логічних операторів входять всі умови що відображають закономірності процесу який алгоритмізуеться.

ВИБІР МОВИ ПРОГРАМУВАННЯ ТА СЕРЕДОВИЩА РОЗРОБКИ

3.1 Вибір мови програмування та середовища розробки

Будучи одним з найбільш популярних мов програмування, C++ широко використовується для розробки програмного забезпечення. C++ був розроблений в якості мови програмування, а не як компілятор мову перекладу, і, таким чином, менш ніж ідеально підходить для використання в якості мови-посередника. Також широко використовується для реалізації кінцевому користувачеві програми. У якості базової алгоритмічної мови обрана мова С++ як представник сучасних універсальних мов, що має достатньо зручний інструментарій візуального програмування Windows-додатків. Враховуючи обрану мову програмування для написання програми, вибір операційної системи є однозначним – Windows.

3.2 Бібліотеки, які використовуються при написанні програми:

#include <iostream> — бібліотека і відповідний заголовний файл, які використовується для організації введення-виведення в мові програмування C++. Він включений в стандартну бібліотеку C++. Назва утворена від англ. Input/Output Stream (потік введення-виведення). У мові C++ і її попереднику, мові програмування Сі, немає вбудованого вводу-виводу.

Натомість використовується бібліотека iostream яка керує вводом-виводом, як і stdio.h у Ci. iostream використовує об'єкти cin, cout, cerr і [clog](https://uk.wikipedia.org/w/index.php?title=Clog&action=edit&redlink=1) для передачі інформації між стандартними потоками введення, виведення, помилок (безбуферний) і помилок (буферний) відповідно. Будучи частиною стандартної бібліотеки C++, ці об'єкти також є частиною стандартного простору імен — std.

#include <Windows.h> - конкретний файл заголовків, що містить заяви для всіх функцій в Windows API, всі спільні макроси, що використовуються програмістами вікон, і всі типи даних, використовуваних різними функціями і підсистемами.

#include <string> - клас з методами і змінними для організації роботи з рядками в мові програмування C ++. Він включений в стандартну бібліотеку C ++. Назва походить від імен сатиричного типу даних (англ. String; з англ. - «рядок»). У мові C ++ і його попередника, мовою програмування Сі, немає вбудованої підтримки строкового типу даних, замість цього використовується масив символів. string управляє рядками, як і string.h в Сі. string використовує єдиний об'єкт string для організації роботи з рядками. Будучи частиною стандартної бібліотеки C ++, ці об'єкти також є частиною стандартного простору імен - std.

#include <deque> - шаблон класу контейнера, який реалізує двостороння чергу. Він забезпечує аналогічну обчислювальну складність з вектором для більшості операцій, з тим помітним виключенням, що він забезпечує вставку і видалення з постійною швидкістю з обох кінців послідовності елементів. . На відміну від vector, deque використовує несуміжні блоки пам'яті і не надає коштів для управління ємністю контейнера і моментом перерозподілу пам'яті. Як і vector, deque пропонує підтримку ітераторів довільного доступу, а вставка і видалення елементів робить недійсними всі ітератори в двосторонньої черги.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

При запуску програми, виводяться всі дії, які може виконати програма. Якщо число введене не коректно, то програма видасть про це відповідне повідомлення, а якщо число введено вірно, то програма виконає те, що обрав користувач.

Варіантами виконання програми є:

Приймається вирази від користувача, які треба порахувати. Потім з цифр формується об'єкти класа big\_digit і між цими об'єктами виконується арифметичні операції.

Приклади арифметичних операцій

Сума -- big\_digit operator + (const big\_digit & Other);

Різниця -- big\_digit operator- (const big\_digit & Other);

Добуток між великими числами -- big\_digit operator \* (const big\_digit & Other) const;

Добуток великого числа і цілого -- big\_digit operator \* (const int & Other) const;

Ділення -- big\_digit operator / (const big\_digit & Other);

Залишок від ділення -- big\_digit operator% (const big\_digit & Other);

Арифметичні операції виконується в порядку арифметичних законів.

Отриманий результат виводиться на екран, користувачеві пропонується ввести ще один вираз.

5 ІНСТРУКЦІЯ КОРИСТУВАЧА

1. Після запуску програми, користувача просять ввести данні для підрахунку. Нижче розташовано приклад правильного вводу. Якщо обраний варіант є можливим, почнеться виконання команди.
2. Якщо користувач введе не так, як зазначено у прикладі, або замість цифр літери, то програма повідомить про це користувачу та закінчить роботу.
3. Щоб отримати результат, треба написати задачу та після неї ввести

« = » .

1. І якщо користувач захоче вийти, то для цього вистачить лише ввести «x» та натиснути будь-яку клавішу.

ВИСНОВКИ

У курсовому проекті була розроблена програма-калькулятор, що реалізує виконання чотирьох основних арифметичних операцій. Програма працездатна і відповідає заявленим вимогам. У процесі виконання даного курсового проекту були закріплені знання з використання класів та з основ об'єктно-орієнтованого програмування. Ця курсова робота демонструє автоматизоване рішення арифметичної задачі, шляхом написання її на мові програмування С ++. Мова C ++ з'явився потужним і стрімким ривком у розвитку програмування. C ++ і донині займає панівне становище серед мов програмування в світі. Величезна безліч професійних програмістів використовує саме його при розробці різного роду проектів. Очевидно, ця мова буде зберігати своє солідне становище ще не один рік, при цьому як і раніше розвиваючись і вдосконалюючись. Завдання «Калькулятор», автоматизована за допомогою середовища програмування С ++, дозволяє обчислювати будь-які прості вирази, що дозволяє економити час на розрахунки.

СПИСОК ЛІТЕРАТУРИ

1. Бьерн Страуструп "С++".2010. 250с.
2. Бьярне Страуструп «Программирование принципы и практика использования c++». 2016. 1329с
3. <https://en.cppreference.com/w/>
4. Глушаков С.В. и др. Язык программирования С++. - Харьков: Фолио, 2002. - 500с

ДОДАТОК А. ТЕКСТ ПРОГРАМИ

Лістинг Calculator.h

#pragma once

#include "big\_digit.h"

#include <iostream>

using namespace std;

class Token {

public:

char kind; // what kind of token

big\_digit value;

Token(char ch):kind(ch), value(0) { }

Token(char ch, string temp):kind(ch), value(temp) { }

};

//——————————————————————————

class Token\_stream {

public:

Token\_stream(); // make a Token\_stream that reads from cin

Token get(); // get a Token (get() is defined elsewhere)

void putback(Token t); // put a Token back

private:

bool full; // is there a Token in the buffer?

Token buffer; // here is where we keep a Token put back using putback()

};

//——————————————————————————

// The constructor just sets full to indicate that the buffer is empty:

Token\_stream::Token\_stream()

:full(false), buffer(0) // no Token in buffer

{

}

//——————————————————————————

// The putback() member function puts its argument back into the Token\_stream’s buffer:

void Token\_stream::putback(Token t)

{

if (full) throw exception("putback() into a full buffer");

buffer = t; // copy t to buffer

full = true; // buffer is now full

}

//——————————————————————————

Token Token\_stream::get()

{

if (full) { // do we already have a Token ready?

// remove token from buffer

full = false;

return buffer;

}

char ch;

cin >> ch; // note that >> skips whitespace (space, newline, tab, etc.)

switch (ch) {

case '=': // for “print”

case 'x': // for “quit”

case '(': case ')': case '+': case '-': case '\*': case '/': case '%':

return Token(ch); // let each character represent itself

case '0': case '1': case '2': case '3': case '4':

case '5': case '6': case '7': case '8': case '9':

{

cin.putback(ch); // put digit back into the input stream

string temp;

cin >> temp; // read a floating-point number

return Token('8', temp); // let ‘8’ represent “a number”

}

default:

throw exception("Bad token");

}

}

//——————————————————————————

Token\_stream ts; // provides get() and putback()

//——————————————————————————

big\_digit expression(); // declaration so that primary() can call expression()

//——————————————————————————

// deal with numbers and parentheses

big\_digit primary()

{

Token t = ts.get();

big\_digit temp;

switch (t.kind) {

case '(': // handle '(' expression ')'

{

big\_digit d = expression();

t = ts.get();

if (t.kind != ')')

{

throw exception("')' expected");

}

return d;

}

case '-':

temp = primary();

temp.sign \*= -1;

return temp;

case '8': // we use ‘8’ to represent a number

return t.value; // return the number’s value

default:

throw exception("primary expected");

}

}

//——————————————————————————

// deal with \*, /, and %

big\_digit term()

{

big\_digit left = primary();

Token t = ts.get(); // get the next token from token stream

while (true) {

switch (t.kind) {

case '\*':

left = left\*primary();

t = ts.get();

break;

case '/':

{

big\_digit d = primary();

left =left/ d;

t = ts.get();

break;

}

case '%':

{

big\_digit d = primary();

left =left% d;

t = ts.get();

break;

}

default:

ts.putback(t); // put t back into the token stream

return left;

}

}

}

//——————————————————————————

// deal with + and –

big\_digit expression()

{

big\_digit left = term(); // read and evaluate a Term

Token t = ts.get(); // get the next token from token stream

while (true) {

switch (t.kind) {

case '+':

left =left+ term(); // evaluate Term and add

t = ts.get();

break;

case '-':

left =left- term(); // evaluate Term and subtract

t = ts.get();

break;

default:

ts.putback(t); // put t back into the token stream

return left; // finally: no more + or -: return the answer

}

}

}

Лістинг big\_digit.h

#pragma once

#include <deque>

#include <string>

using namespace std;

class big\_digit

{

long long findDiv(const big\_digit&, const big\_digit&);

deque<int>digit;

public:

int sign;

void del\_zero();

void add\_zero(int need);

big\_digit() { this->sign = 1; this->digit = deque<int>{}; }

big\_digit(int n) { this->sign = 1; this->digit = deque<int>(n,0); }

big\_digit(string s);

void show();

big\_digit (const big\_digit& Other);

big\_digit operator+(const big\_digit& Other);

big\_digit operator-(const big\_digit& Other);

big\_digit operator\*(const big\_digit& Other) const;

big\_digit operator\*(const int& Other) const;

big\_digit operator/(const big\_digit& Other);

big\_digit operator%(const big\_digit& Other);

void operator=(const big\_digit& Other);

bool operator<(const big\_digit& Other)const;

bool operator>(const big\_digit& Other)const;

bool operator==(const big\_digit& Other)const;

};

Лістинг big\_digit.cpp

#include "big\_digit.h"

#include <iostream>

using namespace std;

big\_digit::big\_digit(string s) {

if (int(s[0]) <= 1 || int(s[0]) >= 255) // isdigit крашит програму, если принимает русскую букву. Эта проверка исправляет ошибка

{

throw exception("не правильный ввод числа");

}

if (!(isdigit(s[0]) || s[0]=='-'))

{

throw exception("Неправильный ввод числа");

}

isdigit(s[0]) ? this->sign = 1 : this->sign = -1;

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

if (s[i]=='-' && i==0)

{

}

else

{

if (int(s[i]) <= 1 || int(s[i]) >= 255) // isdigit крашит програму, если принимает русскую букву. Эта проверка исправляет ошибка

{

for (int j = s.size() - 1; j >= i; j--)

{

cin.putback(s[j]);

}

return;

//throw exception("не правильный ввод числа");

}

if (!isdigit(s[i]))

{

for (int j = s.size()-1; j >= i; j--)

{

cin.putback(s[j]);

}

return;

//throw exception("Неправильный ввод числа");

}

this->digit.push\_back((s[i] - '0'));

}

}

}

void big\_digit::show() {

if (this->sign == -1)

{

cout << "-";

}

for (auto it = this->digit.begin(); it != this->digit.end(); it++)

{

cout <<\*it;

}

cout << endl;

}

big\_digit::big\_digit (const big\_digit& Other) {

this->sign = Other.sign;

this->digit.clear();

for (int i = 0; i < Other.digit.size(); i++)

{

this->digit.push\_back(Other.digit[i]);

}

}

void big\_digit::operator=(const big\_digit& Other) {

this->sign = Other.sign;

this->digit.clear();

for (int i = 0; i < Other.digit.size(); i++)

{

this->digit.push\_back(Other.digit[i]);

}

}

void big\_digit::del\_zero() {

while (this->digit.size()>1 && this->digit[0] == 0)

{

this->digit.erase(this->digit.begin());

}

}

void big\_digit::add\_zero(int need) {

int count = 0;

while (this->digit.size() + count < need)

{

this->digit.push\_front(0);

count++;

}

}

big\_digit big\_digit::operator+(const big\_digit& Other) {

// result - большое число

big\_digit result;

big\_digit add;

if (this->digit.size() == Other.digit.size())

{

for (long long i = 0; i < this->digit.size(); i++)

{

if (this->digit[i] != Other.digit[i])

{

result = this->digit[i] > Other.digit[i] ? \*this : Other;

add = this->digit[i] > Other.digit[i] ? Other : \*this;

break;

}

// два числа одинаковые, но может быть отличие в знаках

result = this->sign >= Other.sign ? \*this : Other;

add = this->sign >= Other.sign ? Other : \*this;

}

}

else

{

result = this->digit.size() > Other.digit.size() ? \*this : Other;

add = this->digit.size() > Other.digit.size() ? Other : \*this;

}

// если большое число отрицательное, выносим минус за скобки

bool minus\_before\_skobki = false;

// меняем знаки, если вынесли -1

if (result.sign==-1)

{

minus\_before\_skobki = true;

result.sign \*= -1;

add.sign \*= -1;

}

// добавляем нули

add.add\_zero(result.digit.size());

int carry = 0;

for (long long i = result.digit.size()-1, add\_i= add.digit.size()-1; add\_i >= 0; i--, add\_i--)

{

result.digit[i] +=carry;

carry = 0;

int now = result.digit[i] + (add.digit[add\_i] \*add.sign);

if (now<0)

{

now += 10;

carry = -1;

}

else if (now>=10)

{

carry = 1;

now -= 10;

}

result.digit[i] = now;

}

if (carry==1)

{

result.digit.push\_front(carry);

}

if (carry==-1)

{

result.digit[0] += carry;

}

if (minus\_before\_skobki)

{

result.sign \*= -1;

}

result.del\_zero();

return result;

}

big\_digit big\_digit::operator-(const big\_digit& Other) {

big\_digit temp = Other;

temp.sign \*= -1;

return \*this + temp;

}

big\_digit big\_digit::operator\*(const big\_digit& Other) const{

big\_digit temp\_this = this->digit.size() >= Other.digit.size() ? \*this : Other;

big\_digit temp\_Other = this->digit.size() >= Other.digit.size() ? Other : \*this;

big\_digit result(this->digit.size()+Other.digit.size());

big\_digit temp;

for (long long other\_i = temp\_Other.digit.size() - 1, number\_iteration=0; other\_i >= 0; other\_i--,number\_iteration++)

{

temp = 0;

int carry = 0;

for (long long i = temp\_this.digit.size() - 1; i >= 0; i--)

{

int now = temp\_this.digit[i] \* temp\_Other.digit[other\_i]+carry;

temp.digit.push\_front(now%10);

carry = now / 10;

}

if (carry != 0)

{

temp.digit.push\_front(carry);

}

for (int i = number\_iteration; i > 0; i--)

{

temp.digit.push\_back(0);

}

result = result + temp;

}

if (this->sign!=Other.sign )

{

result.sign = -1;

}

return result;

}

big\_digit big\_digit::operator\*(const int& Other) const {

big\_digit result;

int carry = 0;

for (long long i = this->digit.size() - 1; i >= 0; i--)

{

int now = this->digit[i] \* Other + carry;

result.digit.push\_front(now % 10);

carry = now / 10;

}

if (carry != 0)

{

result.digit.push\_front(carry);

}

if (this->sign==-1)

{

result.sign = -1;

}

return result;

}

long long big\_digit::findDiv(const big\_digit& dividend, const big\_digit& divisor) {

int i=1;

if (dividend==0)

{

return 0;

}

while (true)

{

if (dividend>divisor\*i || dividend == divisor\*i)

{

i++;

}

else

{

return i-1;

}

}

}

big\_digit big\_digit::operator/(const big\_digit& Other) {

if (Other.digit[0]==0)

{

throw(exception("Деление на 0"));

}

big\_digit result;

big\_digit temp;

big\_digit temp\_Other = Other;

temp\_Other.sign = 1;

bool first = true;

for (long long i = 0; i < this->digit.size(); i++)

{

temp.digit.push\_back(this->digit[i]);

temp.del\_zero();

if (temp>temp\_Other)

{

int div = findDiv(temp, temp\_Other);

result.digit.push\_back(div);

temp = temp - (temp\_Other \* div);

first = false;

}

else if(!first)

{

result.digit.push\_back(0);

}

}

if (this->sign != Other.sign )

{

result.sign = -1;

}

return result;

}

big\_digit big\_digit::operator%(const big\_digit& Other) {

if ((\*this < 0 || Other < 0) || (\*this == 0 || Other == 0))

{

throw(exception("Операция % только для натуральных чисел"));

}

if (\*this<Other)

{

return \*this;

}

return \*this - (\*this / Other)\*Other;

}

bool big\_digit::operator<(const big\_digit& Other)const {

if (this->sign < Other.sign)

{

return true;

}

else if (this->sign == Other.sign)

{

if (this->digit.size() < Other.digit.size())

{

return true;

}

else if (this->digit.size() == Other.digit.size())

{

for (long long i = 0; i < this->digit.size(); i++)

{

if (this->digit[i] < Other.digit[i])

{

return true;

}

else if (this->digit[i] > Other.digit[i])

{

return false;

}

}

}

}

return false;

}

bool big\_digit::operator>(const big\_digit& Other)const {

if (this->sign > Other.sign)

{

return true;

}

else if (this->sign == Other.sign)

{

if (this->digit.size() > Other.digit.size())

{

return true;

}

else if (this->digit.size() == Other.digit.size())

{

for (long long i = 0; i < this->digit.size(); i++)

{

if (this->digit[i] > Other.digit[i])

{

return true;

}

else if (this->digit[i] < Other.digit[i])

{

return false;

}

}

}

}

return false;

}

bool big\_digit::operator==(const big\_digit& Other)const {

if (this->digit.size()==Other.digit.size())

{

if (this->sign==Other.sign)

{

for (long long i = 0; i < this->digit.size(); i++)

{

if (this->digit[i] != Other.digit[i])

{

return false;

}

}

}

else

{

return false;

}

}

else

{

return false;

}

return true;

}

Лістинг Dimon\_big\_number.cpp

#include <iostream>

#include <windows.h>

#include "big\_digit.h"

#include "Calculator.h"

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Ввeдіть данні, для підрахунку.\n";

cout << "Приклад вводу: ( 2123 + 999 ) \* 3224152 / 2341 = \n";

cout << "Для сбросу введіть 'x' \n";

big\_digit val = 0;

char quit = '1';

do

{

cout << "Программа очікує вводу данних\n";

while (cin) {

try

{

Token t = ts.get();

if (t.kind == 'x') break; // 'x' for quit

if (t.kind == '=') { // '=' for “print now”

cout << "=";

val.show();

break;

}

else {

ts.putback(t);

val = expression();

}

}

catch (const std::exception& e)

{

cout << e.what() << endl;

cin.ignore(100, '\n');

break;

}

catch (...)

{

cout << "Unknown exception!" << endl;

break;

}

}

cout << "Введіть 'q' для выходe. Інший символ для продовження роботи\n";

cin >> quit;

cin.ignore(100, '\n');

} while (quit != 'x');

}

ДОДАТОК Б. РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

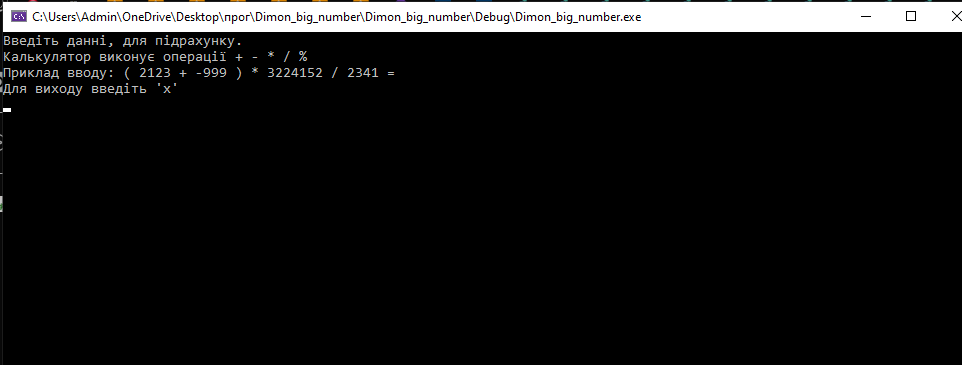


Рисунок Б.1

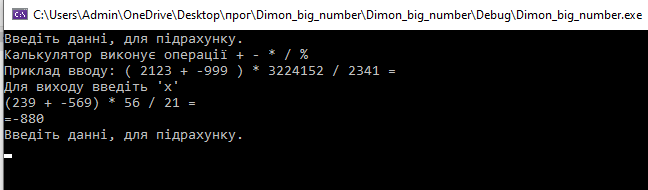


Рисунок Б.2 Відображення правильного вводу та відповіді

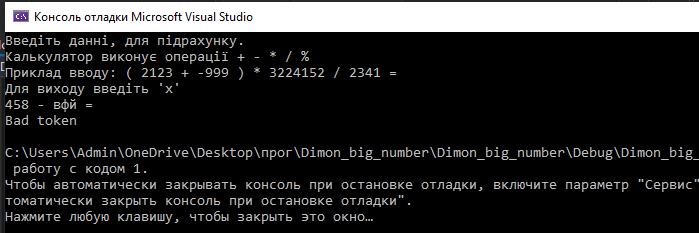


Рисунок Б.2 Відображення неправильного вводу

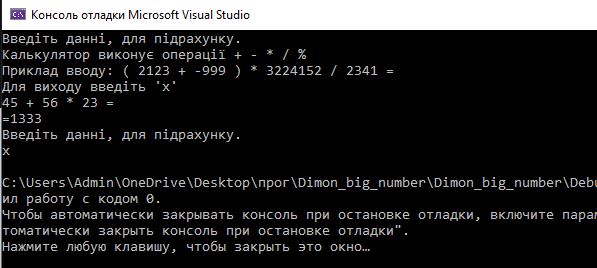


Рисунок Б.2 Відображення закриття програми через введення «х»