Курсовая работа

Разработка приложения «Инженерный калькулятор»

2018

СОДЕРЖАНИЕ

[1 ЗАДАНИЕ 4](#_Toc516500826)

[2 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc516500827)

[3 ОПИСАНИЕ ПРОЕКТНОГО РЕШЕНИЯ 6](#_Toc516500828)

[3.1 Формы 6](#_Toc516500829)

[3.2 Элементы управления 6](#_Toc516500830)

[4 РАЗРАБОТКА ПРОГРАММЫ 8](#_Toc516500831)

[4.1 Описание функций 8](#_Toc516500832)

[4.2 Описание классов 9](#_Toc516500833)

[4.3 Файлы с исходным кодом 9](#_Toc516500834)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 11](#_Toc516500835)

[5.1 Назначение системы и выполняемые функции 11](#_Toc516500836)

[5.2 Условия применения системы 12](#_Toc516500837)

[5.3 Сценарий взаимодействия с пользователем 12](#_Toc516500838)

[6 ТЕСТИРОВАНИЕ 16](#_Toc516500839)

[ПРИЛОЖЕНИЕ A 18](#_Toc516500840)

[А.1 Исходный файл main.cpp 18](#_Toc516500841)

[А.2 Заголовочный файл Operation.h 23](#_Toc516500842)

[А.3 Исходный файл Operation.cpp 24](#_Toc516500843)

[А.4 Заголовочный файл Display.h 25](#_Toc516500844)

[А.5 Исходный файл Display.cpp 26](#_Toc516500845)

[А.6 Заголовочный файл Resource.h 27](#_Toc516500846)

[А.7 Файл ресурсов calc.rc 28](#_Toc516500847)

# 1 ЗАДАНИЕ

Разработка графического интерфейса приложения «Инженерный калькулятор». Создать макет калькулятора (цифры от 0 до 9). С помощью меню Вид и диалогового окна задавать вид калькулятора Простейший и Инженерный.

# 2 ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать приложение, моделирующее работу арифметического калькулятора. Один из ближайших примеров такого приложения – стандартный калькулятор Windows 7, имеющий 4 режима работы: «Обычный», «Инженерный», «Программист» и «Статистика».

Стандартный калькулятор можно взять в качестве ориентировочного образца разрабатываемого приложения. В данной работе необходимо реализовать только два режима – «Обычный» и «Инженерный».

В любом режиме окно калькулятора должно иметь цифровой дисплей для отображения вычислительной информации, кнопки ввода данных (цифры от 0 до 9), кнопки выполнения арифметических операций и вспомогательные кнопки (например, сброс результата).

В базовом режиме должны быть доступны основные арифметические операции – как минимум, сложение, вычитание, умножение и деление. В расширенном режиме к ним должны добавляться дополнительные операции – например, тригонометрические, степенные и логарифмические функции.

Приложение разрабатывается для использования под управлением операционной системы Microsoft Windows. В качестве средств разработки необходимо использовать среду Microsoft Visual Studio, язык программирования C++ и прикладной интерфейс Win32 API.

# 3 ОПИСАНИЕ ПРОЕКТНОГО РЕШЕНИЯ

## 3.1 Формы

Главным окном приложения является диалоговое окно с кнопками и дисплеем. В таблице 1 указаны характеристики главной формы.

Таблица 1 – Формы

|  |  |  |
| --- | --- | --- |
| Идентификатор | Надпись | Назначение |
| mainWnd | Калькулятор | Главное окно приложения «Калькулятор» |

Внешний вид главного окна, открытого в Редакторе ресурсов, изображен на рисунке 1.

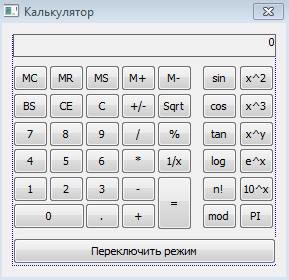


Рисунок 1 – Главное окно приложения

## 3.2 Элементы управления

Список элементов управления главного окна приложения представлен в таблице 2.

Таблица 2 – Элементы управления

|  |  |  |
| --- | --- | --- |
| Идентификатор | Надпись | Назначение |
| txtDisplay | 0 | Надпись, моделирующая дисплей калькулятора |
| btn0 ... btn9 | 0..9 | Кнопки для ввода цифр от 0 до 9 |
| btnDot | . | Кнопка для установки десятичного знака |
| btnSign | +/- | Кнопка смены знака текущего значения |
| btnResult | = | Кнопка вычисления результата |
| btnPlus | + | Кнопка действия: сложение |
| btnMinus | - | Кнопка действия: вычитание |
| btnMul | \* | Кнопка действия: умножение |
| btnDiv | / | Кнопка действия: деление |
| btnC | C | Кнопка общего сброса |
| dtnCE | CE | Кнопка сброса текущего значений |
| btnBS | BS | Кнопка «Backspace»: удаление символа |
| btnMC | MC | Кнопка сброса регистра памяти |
| btnMR | MR | Кнопка чтения регистра памяти |
| btnMS | MS | Кнопка установки регистра памяти |
| btnMPlus | M+ | Кнопка прибавления к регистру памяти |
| btnMMinus | M- | Кнопка вычитания из регистра памяти |
| btnSqrt | sqrt | Кнопка сброса регистра памяти |
| btnPercent | % | Кнопка вычисления квадратного корня |
| btn1divX | 1/x | Кнопка вычисления обратного значения |
| btnSin | sin | Кнопка вычисления синуса |
| btnCos | cos | Кнопка вычисления косинуса |
| btnTan | tan | Кнопка вычисления тангенса |
| btnLog | log | Кнопка вычисления десятичного логарифма |
| btnFact | n! | Кнопка вычисления факториала |
| btnMod | mod | Кнопка действия: остаток от деления |
| btnPowX2 | x^2 | Кнопка возведения в квадрат |
| btnPowX3 | x^3 | Кнопка возведения в куб |
| btnPowXY | x^y | Кнопка вычисления степени |
| btnPowEX | e^x | Кнопка вычисления числа e в степени x |
| btnPow10X | 10^x | Кнопка вычисления 10 в степени x |
| btnPI | PI | Кнопка отображения числа π |
| btnSwitchView | Переключить режим | Кнопка переключения между режимами «Простой» и «Инженерный» |

Внешний вид всех элементов управления изображён на рисунке 1.

# 4 РАЗРАБОТКА ПРОГРАММЫ

## 4.1 Описание функций

void SetMode(HWND hDlg, int mode)

Функция выполняет изменение размеров главного окна в зависимости от установленного режима - инженерный калькулятор несколько шире простого, поскольку имеет дополнительные операционные кнопки. Параметры:

HWND hDlg – дескриптор родительского окна;

int mode – режим.

template<typename T, size\_t N> const size\_t arrayLength(T (&array)[N])

Вспомогательная функция для вычисления размеров статических массивов в стиле C++ - за счёт выведения аргументов шаблона компилятором. Выглядит предпочтительнее макроса или функции, делящей размер массива на размер элемента.

Параметры:

T (&array)[N] – массив. Размер которого известен на этапе компиляции.

INT\_PTR CALLBACK DialogProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)

Оконная процедура главного диалога, т.е. функция, которая принимает все сообщения, отправляемые системой приложению - например, действия пользователя.

Параметры:

HWND hDlg – дескриптор окна диалога;

UINT uMsg – идентификатор сообщения;

WPARAM wParam – первый целочисленный параметр;

LPARAM lParam – второй целочисленный параметр.

int WINAPI \_tWinMain(HINSTANCE hInst, HINSTANCE h, LPTSTR lpCmdLine, int nCmdShow)

Точка входа приложения - главная функция, с которого оно начинает выполнение в операционной системе Windows. Здесь создаётся главное окно, на базе диалогового ресурса и запускается главный цикл обработки сообщений - стандартный для приложений на базе Windows API цикл, содержащий функции GetMessage, TranslateMessage и DispatchMessage.

Параметры:

HINSTANCE hInst – дескриптор текущего экземпляра приложения;

HINSTANCE h – дескриптор уже запущенного экземпляра приложения;

LPTSTR lpCmdLine – командная строка;

int nCmdShow – режим отображения главного окна.

## 4.2 Описание классов

**Display**

Этот класс моделирует дисплей калькулятора - выполняет отображение числовых значений операндов и результатов операций. Он инкапсулирует все детали алфавитно-цифровых и строковых операций, предоставляя удобный и лаконичный интерфейс для управления дисплеем.

**Operation**

Класс арифметических операций. Он инкапсулирует все действия, связанные с хранением текущего значения операндов и выполнением заданных действий над ними.

## 4.3 Файлы с исходным кодом

Все заголовочные и исходные файлы располагаются в общей папке, вместе с файлами ресурсов и проекта.

**Display.h** – заголовочный файл, содержащий объявление класса Display и функцию arrayLength.

**Display.cpp** – исходный файл, содержащий определение методов класса Display.

**Operation.h** – заголовочный файл, содержащий объявление класса Operation.

**Operation.cpp** – исходный файл, содержащий определение методов класса Operation.

**main.cpp** – исходный файл, содержащий вспомогательную функцию SetMode, точку входа приложения и основную оконную процедуру.

# 5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## 5.1 Назначение системы и выполняемые функции

Разрабатываемая программная система предназначена для проведения быстрых арифметических вычислений с плавающей точкой.

Основные функции системы:

* вычисление суммы двух вещественных чисел;
* вычисление разности двух вещественных чисел;
* вычисление произведение двух вещественных чисел;
* вычисление частного двух вещественных чисел;
* вычисление квадратного корня вещественного числа;
* вычисление обратного значения вещественного числа;
* вычисление синуса вещественного числа (радианы);
* вычисление косинуса вещественного числа (радианы);
* вычисление тангенса вещественного числа (радианы);
* вычисление десятичного логарифма вещественного числа;
* вычисление факториала целого числа;
* вычисление остатка от деления для целых чисел;
* вычисление квадрата вещественного числа;
* вычисление куба вещественного числа;
* вычисление степени числа ***e***;
* вычисление степени числа 10;
* вычисление произвольной степени вещественного числа;

Вспомогательные функции системы:

* редактирование и сброс текущего значения;
* поддержка дополнительного регистра памяти и операций с ним – сложения и вычитания;
* отображение на дисплее приближенного значения числа π.
* перевод текущего значения в процентное представление.

## 5.2 Условия применения системы

Минимальные системные требования для работы программы:

* операционная система – Microsoft Windows XP/Vista/7/8/10;
* процессор Pentium с частотой 300 МГц или более быстрый;
* не менее 4 МБ оперативной памяти;
* не менее 1 МБ свободного места на жестком диске;
* клавиатура, мышь или совместимое указывающее устройство;
* видеокарта и монитор, поддерживающие режим SVGA с разрешением не менее чем 800x600 точек.

## 5.3 Сценарий взаимодействия с пользователем

**Структура основного окна программы**

Основное окно программы представляет собой диалог, моделирующий устройство калькулятора –дисплей в верхней части окна и под ним кнопки с цифрами и действиями. Программа запускается в режиме простого калькулятора, как показано на рисунке 2.

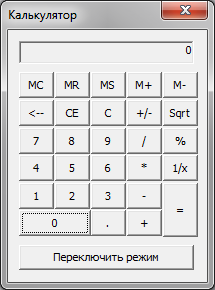


Рисунок 2 – Вид приложения в режиме «Простой калькулятор»

В режиме простого калькулятора доступны только цифровые кнопки и базовый набор арифметических операций. В нижней части окна расположена кнопка переключения режима – с «Простого» на «Инженерный» и обратно.

Вид главного окна в режиме «Инженерный калькулятор» представлен на рисунке 3.

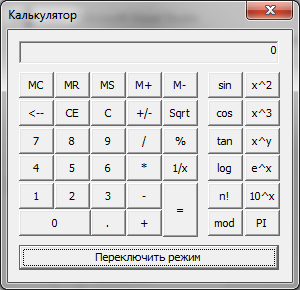


Рисунок 3 – Вид приложения в режиме «Инженерный калькулятор»

В этом режиме рабочая область окна становится шире и пользователю становится доступен ряд дополнительных операций – тригонометрические и степенные функции, десятичный логарифм, деление по модулю и отображение числа π.

**Последовательность действий пользователя**

Ввод значений производится так же, как и в обычном калькуляторе – при нажатии цифровых клавиш, соответствующие цифры добавляются к вводимому значению справа, со стороны младших разрядов. После нажатия кнопки с десятичной точкой, ввод целой части числа заканчивается, дальнейшие цифры будут дополнять мантиссу вещественного числа. Пример ввода значения 1.69 показан на рисунке 4.

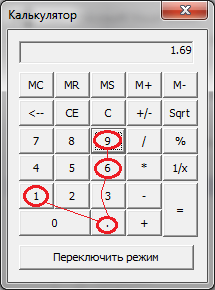


Рисунок 4 – Ввод значения операнда

Весь набор операций делится на операции с одним операндом (например, сменить знак, возвести в квадрат или извлечь корень) и операции с двумя операндами (например, сложение или умножение). Для получения на дисплее результата операции с одним операндом, пользователю достаточно ввести его значение, как показано на рисунке 4, и нажать требуемую операционную кнопку. Результат вычисления квадратного корня для введённого значения (1.69) показан на рисунке 5.

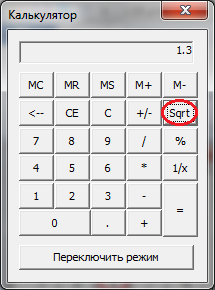


Рисунок 5 – Вычисление квадратного корня

В случае, если нажать кнопку с операцией, требующей два операнда, то ничего не произойдёт – калькулятор будет ждать ввода второго операнда. При этом нажатие кнопки с операцией планирует сброс текущего значения, т.е. следующая введённая цифра уйдёт не в младшие разряды текущего значения, а станет первой цифрой значения второго операнда. После того как будет полностью введено значение второго операнда, пользователь может нажать кнопку результата «=» и на дисплее отобразится результат операции с двумя операндами.

**Перечень сообщений пользователю**

Все выходные сообщения отображаются на дисплее в виде числового результата операции после нажатия кнопки «=», либо в виде текущего значения операнда в процессе ввода. После запуска и при нажатии кнопки сброса, дисплей инициализируется нулевым значением.

# 6 ТЕСТИРОВАНИЕ

Тестирование программы будем проводить в режиме «Инженерный калькулятор», чтобы охватить больше функций.

Для тестирования программы выберем следующий сценарий:

1. Ввод значения 2.5.
2. Сменить знак (результат: -2.5).
3. Прибавить значение 5.7 (результат: 3.2).
4. Умножить на 5 (результат: 16).
5. Вычислить факториал (результат: 20922789888000).
6. Сбросить состояние калькулятора (результат: 0).
7. Вывести на дисплей значение π (результат: 3.14159265).
8. Разделить на 2 (результат: 1.57079633).
9. Вычислить значение синуса (результат: 1).

Последовательность нажатия кнопок следующая:

2 → . → 5 → +/- → + → 5 → . → 7 → = → \* → 5 → = → n! → C → PI → / → 2 → = → sin

В результате тестирования, все промежуточные и финальное значения сценария совпали с выводом программы, т.е. со значениями на дисплее калькулятора. Результаты всей последовательности тестирования изображены на рисунке 6, слева направо и сверху вниз.

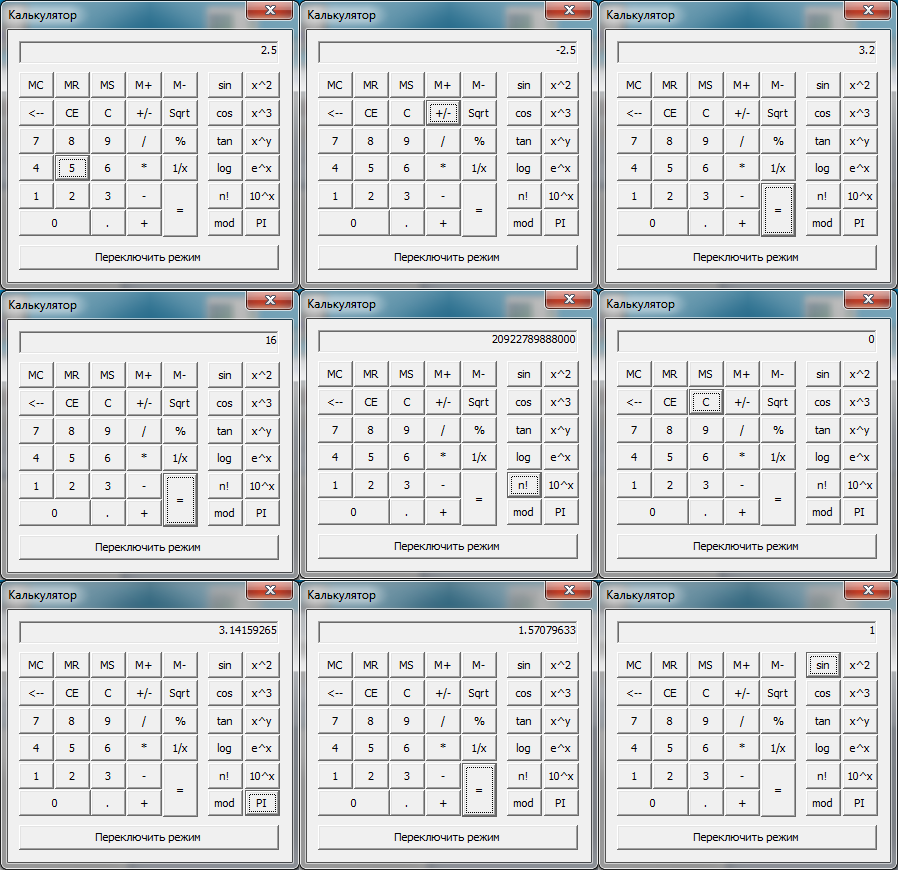


Рисунок 6 – Тестирование приложения

# ПРИЛОЖЕНИЕ A

## А.1 Исходный файл main.cpp

#include "stdafx.h"

#include "Resource.h"

#include "Display.h"

#include "Operation.h"

HINSTANCE g\_hInst = nullptr;

// Текущая операция

Operation\* op = nullptr;

// Дисплей калькулятора

Display\* display = nullptr;

// Флаг "запланирован сброс значения"

bool reset = false;

// Регистр памяти

double memory = 0.0;

// Режим калькулятора: 0 - простой, 1 - инженерный

int mode = 0;

// Размеры окна для режимов

const int modeWidth[] = { 215, 300 };

const int modeHeight[] = { 310, 310 };

// Установка размеров окна для режимов "Простой" и "Инженерный"

void SetMode(HWND hDlg, int mode)

{

SetWindowPos(hDlg, NULL, 0, 0, modeWidth[mode], modeHeight[mode], SWP\_NOMOVE | SWP\_NOOWNERZORDER | SWP\_NOZORDER);

SetWindowPos(GetDlgItem(hDlg, txtDisplay), hDlg, 0, 0, modeWidth[mode] - 40, 23, SWP\_NOMOVE | SWP\_NOOWNERZORDER | SWP\_NOZORDER);

SetWindowPos(GetDlgItem(hDlg, btnSwitchView), hDlg, 0, 0, modeWidth[mode] - 40, 25, SWP\_NOMOVE | SWP\_NOOWNERZORDER | SWP\_NOZORDER);

}

// Оконная процедура диалога выбора режима

INT\_PTR CALLBACK DlgProcMode(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)

{

UINT ctrlID = 0;

switch(uMsg) {

// Сообщение от одного из элементов управления, размещённых на диалоговом окне

case WM\_COMMAND:

// Идентификатор элемента управления, являющегося источниом сообщения

ctrlID = LOWORD(wParam);

switch(ctrlID) {

case IDCANCEL: // Нажатие ESC

EndDialog(hDlg, FALSE);

//SendMessage(hDlg, WM\_CLOSE, 0, 0);

break;

case IDOK: // Нажатие OK

if(IsDlgButtonChecked(hDlg, optSimple))

mode = 0;

else

mode = 1;

EndDialog(hDlg, TRUE);

break;

}

return TRUE;

case WM\_INITDIALOG:

switch(mode) {

case 0:

CheckDlgButton(hDlg, optSimple, BST\_CHECKED);

break;

case 1:

CheckDlgButton(hDlg, optAdvanced, BST\_CHECKED);

break;

}

return TRUE;

}

return FALSE;

}

// Главная оконная процедура (обработчик сообщений главного окна приложения)

INT\_PTR CALLBACK DialogProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)

{

UINT ctrlID = 0;

switch(uMsg) {

// Сообщение от одного из элементов управления, размещённых на диалоговом окне

case WM\_COMMAND:

// Идентификатор элемента управления, являющегося источниом сообщения

ctrlID = LOWORD(wParam);

// Обрабатываем нажатие цифровой кнопки

if(btn0 <= ctrlID && ctrlID <= btn9) {

// Сброс текущего значения (если он был запланирован)

if(reset) {

reset = false;

display->Reset();

}

// Добавляем очередную цифру к текущему значению

display->AddDigit(ctrlID - btn0);

return TRUE;

}

switch(ctrlID) {

case IDCANCEL: // Нажатие ESC

display->Reset();

return TRUE;

case IDM\_MODE:

{

// Отображаем диалог выбора режима

if(DialogBox(g\_hInst, MAKEINTRESOURCE(dlgMode), hDlg, DlgProcMode) == IDOK) {

SetMode(hDlg, mode);

}

}

return TRUE;

case btnSwitchView: // Нажатие кнопки "Переключить режим"

mode ^= 1;

SetMode(hDlg, mode);

display->Reset();

return TRUE;

case btnDot: // Нажатие кнопки "."

display->AddDecimalDot();

return TRUE;

case btnPlus: // Нажатие кнопки "+"

op->Code(opSum);

op->Left()->Value(display->Value());

reset = true;

return TRUE;

case btnMinus: // Нажатие кнопки "-"

op->Code(opSub);

op->Left()->Value(display->Value());

reset = true;

return TRUE;

case btnMul: // Нажатие кнопки "\*"

op->Code(opMul);

op->Left()->Value(display->Value());

reset = true;

return TRUE;

case btnDiv: // Нажатие кнопки "/"

op->Code(opDiv);

op->Left()->Value(display->Value());

reset = true;

return TRUE;

case btnSign: // Нажатие кнопки "+/-"

display->InvertSign();

reset = true;

return TRUE;

case btn1divX: // Нажатие кнопки "1/x"

display->Value(1.0 / display->Value());

reset = true;

return TRUE;

case btnSqrt: // Нажатие кнопки "sqrt"

display->Value(sqrt(display->Value()));

reset = true;

return TRUE;

case btnPercent: // Нажатие кнопки "%"

display->Value(display->Value() \* 100.0);

reset = true;

return TRUE;

case btnPI: // Нажатие кнопки "PI"

display->Value(M\_PI);

reset = true;

return TRUE;

case btnSin: // Нажатие кнопки "sin"

display->Value(sin(display->Value()));

reset = true;

return TRUE;

case btnCos: // Нажатие кнопки "cos"

display->Value(cos(display->Value()));

reset = true;

return TRUE;

case btnTan: // Нажатие кнопки "tan"

display->Value(tan(display->Value()));

reset = true;

return TRUE;

case btnPowEX: // Нажатие кнопки "e^x"

display->Value(exp(display->Value()));

reset = true;

return TRUE;

case btnPowX2: // Нажатие кнопки "x^2"

display->Value(display->Value() \* display->Value());

reset = true;

return TRUE;

case btnPowX3: // Нажатие кнопки "x^3"

display->Value(display->Value() \* display->Value() \* display->Value());

reset = true;

return TRUE;

case btnPow10X: // Нажатие кнопки "10^x"

display->Value(pow(10.0, display->Value()));

reset = true;

return TRUE;

case btnPowXY: // Нажатие кнопки "x^y"

op->Code(opPow);

op->Left()->Value(display->Value());

reset = true;

return TRUE;

case btnLog: // Нажатие кнопки "log"

display->Value(log10(display->Value()));

reset = true;

return TRUE;

case btnMod: // Нажатие кнопки "mod"

op->Code(opMod);

op->Left()->Value(display->Value());

reset = true;

return TRUE;

case btnFact: // Нажатие кнопки "n!"

{

UINT64 result = 1;

UINT64 value = (UINT64)(display->Value() + 0.5);

for(UINT64 i = 2; i <= value; ++i)

result \*= i;

display->Value(result);

}

reset = true;

return TRUE;

case btnMC: // Нажатие кнопки "MC" (сброс регистра памяти)

memory = 0.0;

return TRUE;

case btnMS: // Нажатие кнопки "MS" (присвоение регистру памяти)

memory = display->Value();

return TRUE;

case btnMR: // Нажатие кнопки "MR" (чтение регистра памяти)

display->Value(memory);

reset = true;

return TRUE;

case btnMPlus: // Нажатие кнопки "M+" (прибавление к регистру памяти)

memory += display->Value();

return TRUE;

case btnMMinus: // Нажатие кнопки "M-" (вычитание из регистра памяти)

memory -= display->Value();

return TRUE;

case btnResult: // Нажатие кнопки "="

op->Right()->Value(display->Value());

display->Value(op->Calculate());

reset = true;

return TRUE;

case btnBS: // Нажатие кнопки "<--" (Backspace)

display->Backspace();

return TRUE;

case btnC: // Нажатие кнопки "C"

display->Reset();

return TRUE;

case btnCE: // Нажатие кнопки "CE"

display->Reset();

return TRUE;

}

break;

case WM\_INITDIALOG:

// В обработчике этого сообщения удобно проводить начальную настройку приложения

SetMenu(hDlg, LoadMenu(g\_hInst, MAKEINTRESOURCE(IDR\_MENU1)));

// Выделяем память для вычислитеного объекта

op = new Operation(new Operation, new Operation);

// Выделяем память для объекта дисплея

display = new Display(hDlg, txtDisplay);

// Задаём размеры окна для текущего режима

SetMode(hDlg, mode);

return TRUE;

case WM\_CLOSE:

/\* Это сообщение вызывается перед закрытием окна. Здесь можно спросить пользователя,

действительно ли он хочет завершить работу (особенно при наличии несохранённых данных).

В приложении калькулятора мы просто отдаём команду на закрытие окна приложения. \*/

DestroyWindow(hDlg);

return TRUE;

case WM\_DESTROY:

/\* Мы должны обработать это сообщение, чтобы сказать системе, что хотим

завершить основной поток приложения. Для этого служит функция PostQuitMessage().

Сейчас есть последняя воможность для освобождения выделенных ресурсов. \*/

delete op;

delete display;

PostQuitMessage(0);

return TRUE;

}

return FALSE;

}

// Точка входа приложения

int WINAPI \_tWinMain(HINSTANCE hInst, HINSTANCE h, LPTSTR lpCmdLine, int nCmdShow)

{

MSG msg;

BOOL ret;

// Запоминаем дескриптор приложения

g\_hInst = hInst;

// Создаём главное окно

HWND hDlg = CreateDialogParam(hInst, MAKEINTRESOURCE(mainWnd), 0, DialogProc, 0);

// Отображаем его на экране

ShowWindow(hDlg, nCmdShow);

// Стандартным образом обрабатываем поступающие сообщения в цикле

while((ret = GetMessage(&msg, 0, 0, 0)) != 0) {

if(ret == -1)

return -1;

if(!IsDialogMessage(hDlg, &msg)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return 0;

}

## А.2 Заголовочный файл Operation.h

#pragma once

// Типы операций

enum Operations {

opNop, opSum, opSub, opMul, opDiv, opSqrt, opPow, opMod

};

// Класс, моделирующий операцию калькулятора

class Operation {

public:

// Конструктор

Operation(Operation\* l = nullptr, Operation\* r = nullptr, Operations code = opNop, double value = 0.0);

// Деструктор

~Operation();

// Метод доступа к типу операции

inline Operations Code() const {

return code;

}

// Модификатор типа операции

inline void Code(Operations val) {

code = val;

}

// Метод доступа к текущему значению

inline double Value() const {

return value;

}

// Модификатор текущего значения

inline void Value(double val) {

value = val;

}

// Метод доступа к левому операнду

inline Operation\* Left() {

return left;

}

// Метод доступа к правому операнду

inline Operation\* Right() {

return right;

}

// Вычисление результатов операции

double Calculate();

private:

// Левый операнд

Operation\* left;

// Правый операнд

Operation\* right;

// Тип операции

Operations code;

// Текущее значение (либо значение операнда, либо результат вычислений)

double value;

};

## А.3 Исходный файл Operation.cpp

#include "stdafx.h"

#include "Operation.h"

Operation::Operation(Operation\* l, Operation\* r, Operations code, double value)

: left(l), right(r), code(code), value(value)

{

}

Operation::~Operation() {

// Каждая операция управляет памятью своих операндов

if(left)

delete left;

if(right)

delete right;

}

double Operation::Calculate() {

if(left && right) {

// этот экземпляр является операцией - необходимо

// вычислить текущий результат по двум операндам

double a = left->Calculate();

double b = right->Calculate();

switch(code) {

case opSum:

value = a + b;

break;

case opSub:

value = a - b;

break;

case opMul:

value = a \* b;

break;

case opDiv:

value = a / b;

break;

case opPow:

value = pow(a, b);

break;

case opMod:

value = (UINT64)(a + 0.5) % (UINT64)(b + 0.5);

break;

}

}

// если операнды не назначены, то этот экземпляр сам является

// операндом и нужно просто вернуть текущее значение

return value;

}

## А.4 Заголовочный файл Display.h

#pragma once

// Максимальное количество знаков после запятой

static const UINT maxFract = 8;

static const TCHAR\* szRealFmt = \_T("%.8f");

// Вспомогательная функция - получить размер статического массива (на стадии компиляции)

template<typename T, size\_t N>

const size\_t arrayLength(T (&array)[N])

{

return N;

}

// Класс, моделирующий дисплей калькулятора

class Display

{

public:

// Конструктор класса

Display(HWND hDlg, int nCtlID);

// Добавление цифры

void AddDigit(int digit);

// Добавление десятичной точки

void AddDecimalDot();

// Удаление крайнего справа символа

void Backspace();

// Округление до maxFract знаков после запятой

static inline double Round(double d) {

double p = pow(10.0, double(maxFract));

return int64\_t(d \* p + 0.5) / p;

}

// Метод доступа к количеству символов

inline UINT Count() const {

return count;

}

// Метод доступа к текущему значению

inline double Value() const {

return value;

}

// Модификатор текущего значения

void Value(double val);

// Отображение текущего значения на дисплее

void Show(bool cut = false);

// Смена знака текущего значения

void InvertSign();

// Сброс в начальное состояние

void Reset();

private:

// основной текстовый буфер для отображения

TCHAR buffer[256];

// локальный буфер для алфавитно-цифровых преобразований

TCHAR str[32];

// внутреннее представление текущего значения

double value;

// количество отображаемых в данный момент символов

UINT count;

// дескриптор главного окна

HWND hDlg;

// идентификатор элемента управления для отображения дисплея

int nCtlID;

};

## А.5 Исходный файл Display.cpp

#include "stdafx.h"

#include "Display.h"

Display::Display(HWND hDlg, int nCtlID)

: value(0), count(1), hDlg(hDlg), nCtlID(nCtlID)

{

buffer[0] = \_T('0');

buffer[1] = \_T('\0');

}

void Display::AddDigit(int digit) {

\_itot\_s(digit, str, arrayLength(str), 10);

if(1 == count && \_T('0') == buffer[0]) {

buffer[0] = str[0];

} else {

buffer[count++] = str[0];

}

buffer[count] = 0;

Show();

}

void Display::AddDecimalDot() {

buffer[count++] = \_T('.');

buffer[count] =\_T('\0');

Show();

}

void Display::Backspace() {

if(count > 1) {

count--;

} else {

buffer[0] = \_T('0');

}

buffer[count] = \_T('\0');

Show();

}

void Display::Value(double val) {

value = val;

\_stprintf\_s(buffer, arrayLength(buffer), szRealFmt, val);

Show(true);

}

void Display::Show(bool cut) {

if(cut) {

for(int i = \_tcslen(buffer) - 1; i > 0; --i) {

if(\_T('.') == buffer[i]) {

// дошли то точки, не встретив значащих цифр

buffer[i] = \_T('\0');

break;

} else if(\_T('0') != buffer[i]) {

buffer[i + 1] = \_T('\0');

// дошли то крайней справа значащей цифры

break;

}

}

}

value = \_tcstod(buffer, NULL);

SetDlgItemText(hDlg, nCtlID, buffer);

}

void Display::InvertSign() {

if(Round(value) != 0) {

value = -value;

\_stprintf\_s(buffer, arrayLength(buffer), szRealFmt, value);

Show(true);

}

}

void Display::Reset()

{

value = 0.0;

count = 1;

buffer[0] = \_T('0');

buffer[1] = \_T('\0');

Show();

}

## А.6 Заголовочный файл Resource.h

//{{NO\_DEPENDENCIES}}

// Microsoft Visual C++ generated include file.

// Used by calc.rc

//

#define mainWnd 101

#define IDR\_MENU1 102

#define dlgMode 103

#define btn0 1003

#define btn1 1004

#define btn2 1005

#define btn3 1006

#define btn4 1007

#define btn5 1008

#define btn6 1009

#define btn7 1010

#define btn8 1011

#define btn9 1012

#define btnMC 1013

#define btnDot 1014

#define btnMR 1015

#define btnMS 1016

#define btnMPlus 1017

#define btnMMinus 1018

#define btnBS 1019

#define btnCE 1020

#define btnC 1021

#define btnSign 1022

#define btnSqrt 1023

#define btnDiv 1024

#define btnPercent 1025

#define btnMul 1026

#define btn1divX 1027

#define btnMinus 1028

#define btnResult 1029

#define btnPlus 1030

#define txtDisplay 1031

#define txtDbg 1032

#define btnSwitchView 1033

#define btnSin 1034

#define IDC\_RADIO1 1034

#define optAdvanced 1034

#define btnCos 1035

#define optSimple 1035

#define btnTan 1036

#define btnLog 1037

#define btnFact 1038

#define btnMod 1039

#define btnPowX2 1040

#define btnPowX3 1041

#define btnPowXY 1042

#define btnPowEX 1043

#define btnPow10X 1044

#define btnSin12 1045

#define btnPI 1045

#define ID\_40001 40001

#define IDM\_MODE 40002

// Next default values for new objects

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NEXT\_RESOURCE\_VALUE 104

#define \_APS\_NEXT\_COMMAND\_VALUE 40003

#define \_APS\_NEXT\_CONTROL\_VALUE 1036

#define \_APS\_NEXT\_SYMED\_VALUE 101

#endif

#endif

## А.7 Файл ресурсов calc.rc

// Microsoft Visual C++ generated resource script.

//

#include "resource.h"

#define APSTUDIO\_READONLY\_SYMBOLS

/////////////////////////////////////////////////////////////////////////////

//

// Generated from the TEXTINCLUDE 2 resource.

//

#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////

#undef APSTUDIO\_READONLY\_SYMBOLS

/////////////////////////////////////////////////////////////////////////////

// Russian (Russia) resources

#if !defined(AFX\_RESOURCE\_DLL) || defined(AFX\_TARG\_RUS)

LANGUAGE LANG\_RUSSIAN, SUBLANG\_DEFAULT

#ifdef APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// TEXTINCLUDE

//

1 TEXTINCLUDE

BEGIN

"resource.h\0"

END

2 TEXTINCLUDE

BEGIN

"#include ""afxres.h""\r\n"

"\0"

END

3 TEXTINCLUDE

BEGIN

"\r\n"

"\0"

END

#endif // APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// Dialog

//

mainWnd DIALOGEX 0, 0, 189, 156

STYLE DS\_SETFONT | DS\_MODALFRAME | DS\_FIXEDSYS | WS\_POPUP | WS\_CAPTION | WS\_SYSMENU

CAPTION "Калькулятор"

FONT 8, "MS Shell Dlg", 400, 0, 0x1

BEGIN

PUSHBUTTON "0",btn0,7,111,48,16

PUSHBUTTON "1",btn1,7,94,23,16

PUSHBUTTON "2",btn2,31,94,23,16

PUSHBUTTON "3",btn3,55,94,23,16

PUSHBUTTON "4",btn4,7,77,23,16

PUSHBUTTON "5",btn5,31,77,23,16

PUSHBUTTON "6",btn6,55,77,23,16

PUSHBUTTON "7",btn7,7,60,23,16

PUSHBUTTON "8",btn8,31,60,23,16

PUSHBUTTON "9",btn9,55,60,23,16

PUSHBUTTON "MC",btnMC,7,26,23,16

PUSHBUTTON ".",btnDot,55,111,23,16

PUSHBUTTON "MR",btnMR,31,26,23,16

PUSHBUTTON "MS",btnMS,55,26,23,16

PUSHBUTTON "M+",btnMPlus,79,26,23,16

PUSHBUTTON "M-",btnMMinus,103,26,23,16

PUSHBUTTON "<--",btnBS,7,43,23,16

PUSHBUTTON "CE",btnCE,31,43,23,16

PUSHBUTTON "C",btnC,55,43,23,16

PUSHBUTTON "+/-",btnSign,79,43,23,16

PUSHBUTTON "Sqrt",btnSqrt,103,43,23,16

PUSHBUTTON "/",btnDiv,79,60,23,16

PUSHBUTTON "%",btnPercent,103,60,23,16

PUSHBUTTON "\*",btnMul,79,77,23,16

PUSHBUTTON "1/x",btn1divX,103,77,23,16

PUSHBUTTON "-",btnMinus,79,94,23,16

PUSHBUTTON "=",btnResult,103,94,23,33

PUSHBUTTON "+",btnPlus,79,111,23,16

RTEXT "0",txtDisplay,7,7,175,14,WS\_BORDER

PUSHBUTTON "Переключить режим",btnSwitchView,7,132,175,16

PUSHBUTTON "sin",btnSin,133,26,23,16

PUSHBUTTON "cos",btnCos,133,43,23,16

PUSHBUTTON "tan",btnTan,133,60,23,16

PUSHBUTTON "log",btnLog,133,77,23,16

PUSHBUTTON "n!",btnFact,133,94,23,16

PUSHBUTTON "mod",btnMod,133,111,23,16

PUSHBUTTON "x^2",btnPowX2,158,26,23,16

PUSHBUTTON "x^3",btnPowX3,158,43,23,16

PUSHBUTTON "x^y",btnPowXY,158,60,23,16

PUSHBUTTON "e^x",btnPowEX,158,77,23,16

PUSHBUTTON "10^x",btnPow10X,158,94,23,16

PUSHBUTTON "PI",btnPI,158,111,23,16

END

dlgMode DIALOGEX 0, 0, 159, 66

STYLE DS\_SETFONT | DS\_MODALFRAME | DS\_FIXEDSYS | WS\_POPUP | WS\_CAPTION | WS\_SYSMENU

CAPTION "Выбор режима калькулятора"

FONT 8, "MS Shell Dlg", 400, 0, 0x1

BEGIN

DEFPUSHBUTTON "OK",IDOK,102,25,50,14

PUSHBUTTON "Cancel",IDCANCEL,102,45,50,14

GROUPBOX "",IDC\_STATIC,6,2,88,57

CONTROL "Инженерный",optAdvanced,"Button",BS\_AUTORADIOBUTTON,21,36,59,10

CONTROL "Простой",optSimple,"Button",BS\_AUTORADIOBUTTON,21,16,44,10

END

/////////////////////////////////////////////////////////////////////////////

//

// DESIGNINFO

//

#ifdef APSTUDIO\_INVOKED

GUIDELINES DESIGNINFO

BEGIN

mainWnd, DIALOG

BEGIN

LEFTMARGIN, 7

RIGHTMARGIN, 182

TOPMARGIN, 7

BOTTOMMARGIN, 150

END

dlgMode, DIALOG

BEGIN

LEFTMARGIN, 7

RIGHTMARGIN, 152

TOPMARGIN, 7

BOTTOMMARGIN, 59

END

END

#endif // APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// Menu

//

IDR\_MENU1 MENU

BEGIN

POPUP "Вид"

BEGIN

MENUITEM "Режим работы...", IDM\_MODE

END

END

#endif // Russian (Russia) resources

/////////////////////////////////////////////////////////////////////////////

#ifndef APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// Generated from the TEXTINCLUDE 3 resource.

//

/////////////////////////////////////////////////////////////////////////////

#endif // not APSTUDIO\_INVOKED