Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и

системное программирование

ОТЧЁТ

к лабораторной работе №1

на тему

Основы программирования в Win32 API. Оконное приложение Win32 с минимальной функциональной достаточностью. Обработка основных оконных сообщений

Выполнила: студентка группы 153502

Шайкова Вероника Андреевна

Проверил: Гриценко Никита Юрьевич

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc145670720)

[2 Результаты выполнения лабораторной работы 4](#_Toc145670721)

[Выводы 6](#_Toc145670722)

[Список использованных источников 7](#_Toc145670723)

[Приложение А (обязательное) Листинг кода 8](#_Toc145670724)

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API, обладающее минимальным функционалом, позволяющим отработать базовые навыки написания программы на Win32 API, использования виджетов и обработки оконных сообщений (как базовых, так и пользовательских). Реализовать вышеупомянутые требования на примере калькулятора, способного производить основные математические операции, а также вычислять значения некоторых тригонометрических и элементарных функций.

## 2 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы был реализован калькулятор, последовательно (без учета приоритета операций) выполняющий базовые математические операции (сложение, вычитание, умножение, деление) при использовании трех переменных (first, second типа double и sign типа char) и представляющий результат в типе double. Набор чисел осуществляется пользователем при использовании расположенных на окне кнопок (Рисунок 1).

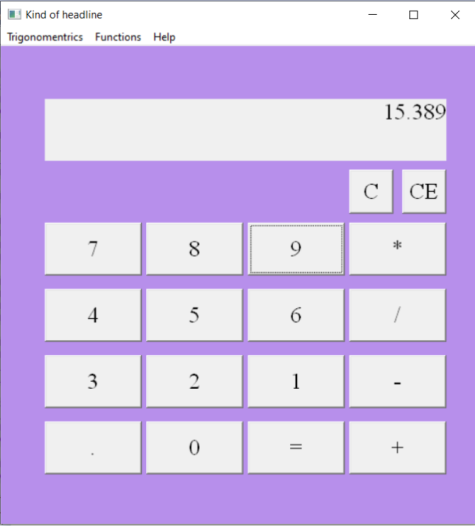
****

Рисунок 1 - Главное окно

Пользователь может применить некоторые тригонометрические (синус, косинус, тангенс, котангенс) или элементарные (извлечение корня, поиск обратного числа, возведение в квадрат, возведения числа *е* в степень *х*) функции, выбрав их из выскакивающих (pop-up) меню ‘Trigonometrics’ и ’Functions’ соответственно (Рисунок 2).

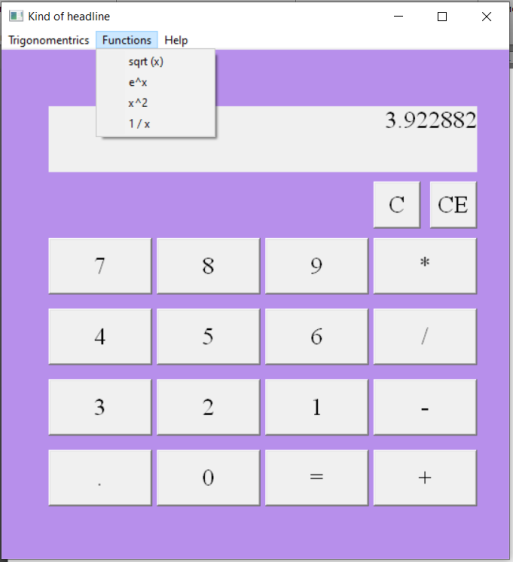
****

Рисунок 2 ─ Меню

Также доступно меню help, вызывающее диалоговое окно с небольшой инструкцией для пользователя (рисунок 3).

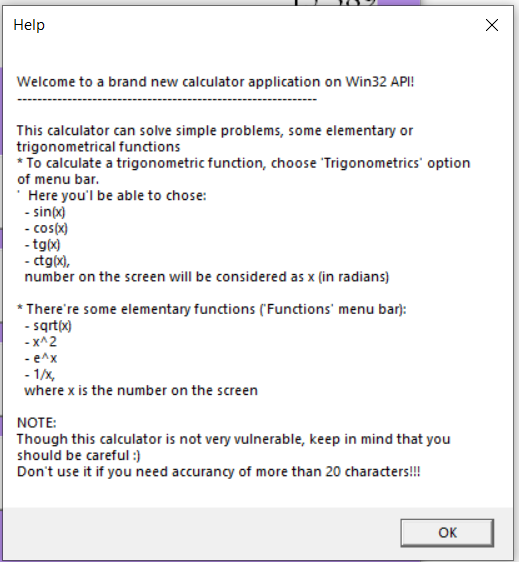


Рисунок 3 - Инструкция

Ограничения: строка ввода/вывода числа может содержать максимально 20 символов, что влияет на точность результата. При использовании калькулятора для больших чисел может возникнуть потеря точности из-за обрезки строки.

## ВЫВОДЫ

В ходе выполнения лабораторной работы было разработано приложение-калькулятор на Win32 API. Особое внимание было уделено созданию базового окна приложения, иерархического меню, работе с элементами управления: считывание и запись данных в окно, обработка сообщений кнопок, создание пользовательских шрифтов, вызов диалоговых окон. Результатом стало работоспособное простое приложение, написанное на языке C++, имеющее удобный интерфейс пользователя и пригодное для вычислений в бытовых условиях.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Щупак Ю. Win32 API. Разработка приложений для Windows. ─ СПб: Питер, 2008. ─ 592 с.: ип.

[2] Создание классических приложений для Windows с использованием API Win32 [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api ─ Дата доступа 10.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

**lab1.cpp**

#include <windows.h>

#include <string>

// #include "calculator\_funcs.cpp"

#include "globals\_defines.h"

double eq(char sign, double first, double second) {

switch (sign) {

case '+':

return first + second;

break;

case '-':

return first - second;

break;

case '/':

if (second != 0) return first / second;

break;

case '\*':

return first \* second;

break;

default:

return 0;

}

}

void func(std::string f) {

GetWindowTextA(number\_line, buff, 20);

std::string number = buff;

if (number == "error" || number == "-") return;

if (number[number.size() - 1] == '.') number = number + '0';

double d\_number = atof(number.c\_str());

if (f == "sin") {

d\_number = sin(d\_number);

}

else if (f == "cos") {

d\_number = cos(d\_number);

}

else if (f == "tan") {

d\_number = tan(d\_number);

}

else if (f == "ctg") {

d\_number = 1 / tan(d\_number);

}

else if (f == "sqrt") {

d\_number = sqrt(d\_number);

}

else if (f == "exp") {

d\_number = exp(d\_number);

}

else if (f == "x2") {

d\_number = d\_number \* d\_number;

}

else if (f == "1/x") {

d\_number = 1 / d\_number;

}

number = std::to\_string(d\_number);

if (number.size() > 20) number = number.substr(0, 20);

bool is\_numeric = true;

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

char d = number[i];

if ((d < '0' || d > '9') && d != '.' && d != '-') {

is\_numeric = false;

break;

}

}

if (!is\_numeric) {

first = second = 0;

sign = '+';

SetWindowTextA(number\_line, "error");

return;

}

SetWindowTextA(number\_line, number.c\_str());

}

void add\_character(char digit) {

if (must\_be\_cleared) {

first = second = 0;

sign = '+';

SetWindowTextA(number\_line, "0");

must\_be\_cleared = false;

}

sign\_changed = false;

GetWindowTextA(number\_line, buff, 20);

std::string number = buff;

std::string output = number;

if (digit == '0') {

if (number == "error") output = "0";

else if (number.size() == 1 && number != "0") output = number + "0";

else if (number == "0") return;

else if (number.size() < 20) output = number + "0";

}

else if (digit == '.') {

if (number == "-") output = "-0.";

else if (number == "error") output = "0.";

else if (number.size() < 19) {

bool contain = false;

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

char a = number[i];

if (a == 46) {

contain = true;

break;

}

}

if (!contain) {

output = number + '.';

}

}

}

else {

if (number == "error") output = digit;

else if (number.size() == 1 && number == "0") output = digit;

else if (number.size() < 20) output = number + digit;

}

SetWindowTextA(number\_line, output.c\_str());

}

void remove\_character() {

GetWindowTextA(number\_line, buff, 20);

std::string number = buff;

std::string output;

if (number.size() == 1 || number == "error") output = "0";

else {

number.pop\_back();

output = number;

}

SetWindowTextA(number\_line, output.c\_str());

}

// функция, позволяющая создать новое окно (объект класса WNDCLASS) с определенным цветом фона, формой курсора и т.д.

WNDCLASS createWindowClass(HBRUSH background\_color, HCURSOR cursor\_type, HINSTANCE hInstance, HICON Icon, LPCWSTR window\_name, WNDPROC procedure) {

// создание пустого класса

WNDCLASS new\_window = { 0 };

// наполняем пустой класс параметрами, требуемыми для конкретного окна

new\_window.hbrBackground = background\_color;

new\_window.hCursor = cursor\_type;

new\_window.hInstance = hInstance;

new\_window.hIcon = Icon;

new\_window.lpszClassName = window\_name;

new\_window.lpfnWndProc = procedure;

return new\_window;

}

// create menu foe app

void CreateMenus(HWND wnd) {

HMENU root = CreateMenu();

HMENU elem\_funcs = CreateMenu();

HMENU trigonometrics = CreateMenu();

AppendMenu(root, MF\_POPUP, (UINT\_PTR)trigonometrics, L"Trigonomentrics");

AppendMenu(trigonometrics, MF\_STRING, M\_SIN\_CHECKED, L"sin (x)");

AppendMenu(trigonometrics, MF\_STRING, M\_COS\_CHECKED, L"cos (x)");

AppendMenu(trigonometrics, MF\_STRING, M\_TAN\_CHECKED, L"tg (x)");

AppendMenu(trigonometrics, MF\_STRING, M\_CTG\_CHECKED, L"ctg (x)");

AppendMenu(root, MF\_POPUP, (UINT\_PTR)elem\_funcs, L"Functions");

AppendMenu(elem\_funcs, MF\_STRING, M\_SQRT\_CHECKED, L"sqrt (x)");

AppendMenu(elem\_funcs, MF\_STRING, M\_EX\_CHECKED, L"e^x");

AppendMenu(elem\_funcs, MF\_STRING, M\_X2\_CHECKED, L"x^2");

AppendMenu(elem\_funcs, MF\_STRING, M\_1DELX\_CHECKED, L"1 / x");

AppendMenu(root, MF\_STRING, M\_HELP\_CHECKED, L"Help");

SetMenu(wnd, root);

}

void AddWidgets(HWND wnd) {

number\_line = CreateWindowA("static", "0", WS\_VISIBLE | WS\_CHILD | ES\_RIGHT , 50, 60, 455, 70, wnd, NULL, NULL, NULL);

SendMessageA(number\_line, WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "C", WS\_VISIBLE | WS\_CHILD, 50 + 345, 140, 50, 50, wnd, (HMENU)BC\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "CE", WS\_VISIBLE | WS\_CHILD, 50 + 345 +60, 140, 50, 50, wnd, (HMENU)BCE\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "7", WS\_VISIBLE | WS\_CHILD, 50, 200, 110, 60, wnd, (HMENU)B7\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "8", WS\_VISIBLE | WS\_CHILD, 50+115, 200, 110, 60, wnd, (HMENU)B8\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "9", WS\_VISIBLE | WS\_CHILD, 50+230, 200, 110, 60, wnd, (HMENU)B9\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "\*", WS\_VISIBLE | WS\_CHILD, 50+345, 200, 110, 60, wnd, (HMENU)BMUL\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "4", WS\_VISIBLE | WS\_CHILD, 50, 200+75, 110, 60, wnd, (HMENU)B4\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "5", WS\_VISIBLE | WS\_CHILD, 50+115, 200+75, 110, 60, wnd, (HMENU)B5\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "6", WS\_VISIBLE | WS\_CHILD, 50+230, 200+75, 110, 60, wnd, (HMENU)B6\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "/", WS\_VISIBLE | WS\_CHILD, 50+345, 200+75, 110, 60, wnd, (HMENU)BDIV\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "3", WS\_VISIBLE | WS\_CHILD, 50, 200+150, 110, 60, wnd, (HMENU)B3\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "2", WS\_VISIBLE | WS\_CHILD, 50+115, 200+150, 110, 60, wnd, (HMENU)B2\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "1", WS\_VISIBLE | WS\_CHILD, 50+230, 200+150, 110, 60, wnd, (HMENU)B1\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "-", WS\_VISIBLE | WS\_CHILD, 50 + 345, 200 + 150, 110, 60, wnd, (HMENU)BMIN\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", ".", WS\_VISIBLE | WS\_CHILD, 50, 200+225, 110, 60, wnd, (HMENU)BPOINT\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "0", WS\_VISIBLE | WS\_CHILD, 50+115, 200+225, 110, 60, wnd, (HMENU)B0\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "=", WS\_VISIBLE | WS\_CHILD, 50+230, 200+225, 110, 60, wnd, (HMENU)BEQ\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

SendMessageA(CreateWindowA("button", "+", WS\_VISIBLE | WS\_CHILD, 50 + 345, 200+225, 110, 60, wnd, (HMENU)BPLUS\_CLICKED, NULL, NULL), WM\_SETFONT, (WPARAM)human\_font, TRUE);

}

// функция, позволяющая создать процедуру

LRESULT CALLBACK createNewProcedure(HWND window, UINT message, WPARAM wp, LPARAM lp) {

switch (message) {

case WM\_CREATE:

CreateMenus(window);

AddWidgets(window);

break;

case WM\_DESTROY:

// выход из программы по закрытию окна

PostQuitMessage(0);

break;

case WM\_COMMAND:

switch (wp)

{

case M\_SIN\_CHECKED:

func("sin");

break;

case M\_COS\_CHECKED:

func("cos");

break;

case M\_TAN\_CHECKED:

func("tan");

break;

case M\_CTG\_CHECKED:

func("ctg");

break;

case M\_SQRT\_CHECKED:

func("sqrt");

break;

case M\_EX\_CHECKED:

func("exp");

break;

case M\_X2\_CHECKED:

func("x2");

break;

case M\_1DELX\_CHECKED:

func("1/x");

break;

case M\_HELP\_CHECKED:

MessageBox(window, help\_message, L"Help", NULL);

break;

case B0\_CLICKED:

add\_character('0');

break;

case B1\_CLICKED:

add\_character('1');

break;

case B2\_CLICKED:

add\_character('2');

break;

case B3\_CLICKED:

add\_character('3');

break;

case B4\_CLICKED:

add\_character('4');

break;

case B5\_CLICKED:

add\_character('5');

break;

case B6\_CLICKED:

add\_character('6');

break;

case B7\_CLICKED:

add\_character('7');

break;

case B8\_CLICKED:

add\_character('8');

break;

case B9\_CLICKED:

add\_character('9');

break;

case BPOINT\_CLICKED:

add\_character('.');

break;

case BEQ\_CLICKED:

{

GetWindowTextA(number\_line, buff, 20);

std::string num = buff;

if (num == "error" || num == "-") {

first = second = 0;

sign = '+';

SetWindowTextA(number\_line, "0");

}

else if (num[num.size() - 1] == '.') {

num = num + '0';

}

second = atof(num.c\_str());

first = eq(sign, first, second);

second = 0;

num = std::to\_string(first);

if (num.size() > 20) num = num.substr(0, 20);

SetWindowTextA(number\_line, num.c\_str());

must\_be\_cleared = true;

break;

}

case BMUL\_CLICKED:

if (!sign\_changed) {

GetWindowTextA(number\_line, buff, 20);

std::string number = buff;

if (number == "error" || number == "-") {

SetWindowTextA(number\_line, "0");

break;

}

else if (number[number.size() - 1] == '.') number = number + '0';

if (!must\_be\_cleared) {

first = eq(sign, first, atof(number.c\_str()));

}

else {

first = atof(number.c\_str());

must\_be\_cleared = false;

}

SetWindowTextA(number\_line, "0");

sign\_changed = true;

}

sign = '\*';

break;

case BDIV\_CLICKED:

if (!sign\_changed) {

GetWindowTextA(number\_line, buff, 20);

std::string number = buff;

if (number == "error" || number == "-") {

SetWindowTextA(number\_line, "0");

//return;

}

else if (number == "0") {

SetWindowTextA(number\_line, "error");

//return;

}

else if (number[number.size() - 1] == '.') number = number + '0';

if(!must\_be\_cleared)

first = eq(sign, first, atof(number.c\_str()));

else {

first = atof(number.c\_str());

must\_be\_cleared = false;

}

SetWindowTextA(number\_line, "0");

sign\_changed = true;

}

sign = '/';

break;

case BPLUS\_CLICKED:

if (!sign\_changed) {

GetWindowTextA(number\_line, buff, 20);

std::string number = buff;

if (number == "error" || number == "-") {

SetWindowTextA(number\_line, "0");

//return;

}

else if (number[number.size() - 1] == '.') number = number + '0';

if (!must\_be\_cleared) {

first = eq(sign, first, atof(number.c\_str()));

}

else {

first = atof(number.c\_str());

must\_be\_cleared = false;

}

SetWindowTextA(number\_line, "0");

sign\_changed = true;

}

sign = '+';

break;

case BMIN\_CLICKED:

if (!sign\_changed) {

GetWindowTextA(number\_line, buff, 20);

std::string number = buff;

if (number == "error" || number == "-") {

SetWindowTextA(number\_line, "0");

//return;

}

else if (number[number.size() - 1] == '.') number = number + '0';

if (!must\_be\_cleared) {

first = eq(sign, first, atof(number.c\_str()));

}

else {

first = atof(number.c\_str());

must\_be\_cleared = false;

}

SetWindowTextA(number\_line, "0");

sign\_changed = true;

}

sign = '-';

break;

case BC\_CLICKED:

remove\_character();

break;

case BCE\_CLICKED:

first = second = 0;

sign = '+';

SetWindowTextA(number\_line, "0");

break;

default:

break;

}

break;

default: return DefWindowProc(window, message, wp, lp);

}

}

// главная функция

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR args, int ncmdshow) {

// создаем основное окно (класс), используя описанную выше функцию

human\_font = CreateFontA(30, 10, 0, 0, FW\_LIGHT, FALSE, FALSE, FALSE, DEFAULT\_CHARSET, OUT\_OUTLINE\_PRECIS, CLIP\_DEFAULT\_PRECIS, ANTIALIASED\_QUALITY,FF\_ROMAN, "Font1");

brush = CreateSolidBrush(RGB(183, 143, 235));

WNDCLASS main\_window = createWindowClass(brush, LoadCursor(NULL, IDC\_CROSS), hInst, LoadIcon(NULL, IDI\_WINLOGO), L"MainWindow", createNewProcedure);

// регистрируем класс главного окна (нужно, ибо создаем через имя, заданное при инициализации переменной типа WNDCLASS)

if(!RegisterClassW(&main\_window)){

return -1;

}

MSG main\_window\_message = { 0 };

// создаем само основное окно с меню, 3мя главнами кнопками, возможностью перемещать/менять размер

CreateWindow(L"MainWindow", L"Kind of headline", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 200, 200, 555, 600, NULL, NULL, NULL, NULL);

while (GetMessage(&main\_window\_message, NULL, NULL, NULL)) {

TranslateMessage(&main\_window\_message);

DispatchMessage(&main\_window\_message);

}

return 0;

}

**globals\_defines.h**

#pragma once

#define M\_SIN\_CHECKED 1

#define M\_COS\_CHECKED 2

#define M\_TAN\_CHECKED 3

#define M\_CTG\_CHECKED 4

#define M\_SQRT\_CHECKED 5

#define M\_EX\_CHECKED 6

#define M\_X2\_CHECKED 7

#define M\_1DELX\_CHECKED 8

#define M\_HELP\_CHECKED 9

#define B0\_CLICKED 10

#define B1\_CLICKED 11

#define B2\_CLICKED 12

#define B3\_CLICKED 13

#define B4\_CLICKED 14

#define B5\_CLICKED 15

#define B6\_CLICKED 16

#define B7\_CLICKED 17

#define B8\_CLICKED 18

#define B9\_CLICKED 19

#define BMUL\_CLICKED 20

#define BDIV\_CLICKED 21

#define BPLUS\_CLICKED 22

#define BMIN\_CLICKED 23

#define BPOINT\_CLICKED 24

#define BEQ\_CLICKED 25

#define BC\_CLICKED 26

#define BCE\_CLICKED 27

HWND number\_line;

HFONT human\_font;

HBRUSH brush;

char sign = '+';

double first = 0, second = 0;

char buff[20];

bool must\_be\_cleared = false;

bool sign\_changed = false;

LPCWSTR help\_message = L"Welcome to a brand new calculator application on Win32 API! \

\n------------------------------------------------------------ \n\n\

This calculator can solve simple problems, some elementary or trigonometrical functions\n\

\* To calculate a trigonometric function, choose 'Trigonometrics' option of menu bar.\n'\

Here you'l be able to chose:\n\

- sin(x)\n\

- cos(x)\n\

- tg(x)\n\

- ctg(x),\n\

number on the screen will be considered as x (in radians)\n\n\

\* There're some elementary functions ('Functions' menu bar):\n\

- sqrt(x)\n\

- x^2\n\

- e^x\n\

- 1/x,\n\

where x is the number on the screen\n\n\

NOTE:\n\

Though this calculator is not very vulnerable, keep in mind that you should be careful :)\n\

Don't use it if you need accurancy of more than 20 characters!!!\

";