МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра РЭС

ОТЧЕТ

по курсовому расчету

По дисциплине «Информационные технологии»

Тема: Вычисление определенного интеграла

Вариант 5.1

Студент гр. 9181 _____

Преподаватель _____

Санкт-Петербург

2020

Содержание

Спецификация

1. ***Общая часть задания:***

   1.1. Проект выполняется в среде Microsoft Visual Studio с использованием компонентов библиотеки Microsoft Foundation Classes (MFC) на языке C++

2. ***Задание:***

   2.1. Вычислить определенный интеграл в границах Xmin…Xmax от заданной функции методом прямоугольника.

   2.2. Построить график функции и график зависимости интеграла от параметров функции

   2.3. Обеспечить интерактивный режим изменений параметра функции и сохранения выбранного графика в файле в виде изображения

   2.4. Функция:

   y(x) = 100*sin(0.01*x+b)+a

3. ***В проекте должно быть:***

   3.1. Разработано оконное приложение для системы X86 выполняющая расчеты и графическую визуализацию результатов с поддержкой универсальной кодировки символов Unicode.

   3.2. Разработан графический пользовательский интерфейс.

   3.3. Пользовательский интерфейс должен использовать:

   3.3.1. Кнопки управления приложением

   3.3.2. Поля ввода данных для расчета

   3.3.3. Язык интерфейса – русский

   3.3.4. Информацию о программе

4. ***Результат должен быть подтвержден расчетами и графиками в системе MathCad***

5. ***В отчёте должно быть:***

   5.1. Спецификация задания.

   5.2. Постановка задачи.

   5.3. Словесное описание алгоритма

   5.4. Варианты взаимодействия оператора и программы

   5.5. Блок – схема движения данных

   5.6. Выбор и обоснование типов переменных

   5.7. Диаграмма классов

   5.8. Структура проекта

   5.9. Инструкция по использованию

   5.10. Текст программы

   5.11. Рисунки с копиями экрана при работе программы

   5.12. Контрольный пример, сравнение результатов с эталоном (MathCad)

   5.13. Ведомость соответствия программы и спецификации

Постановка задачи.

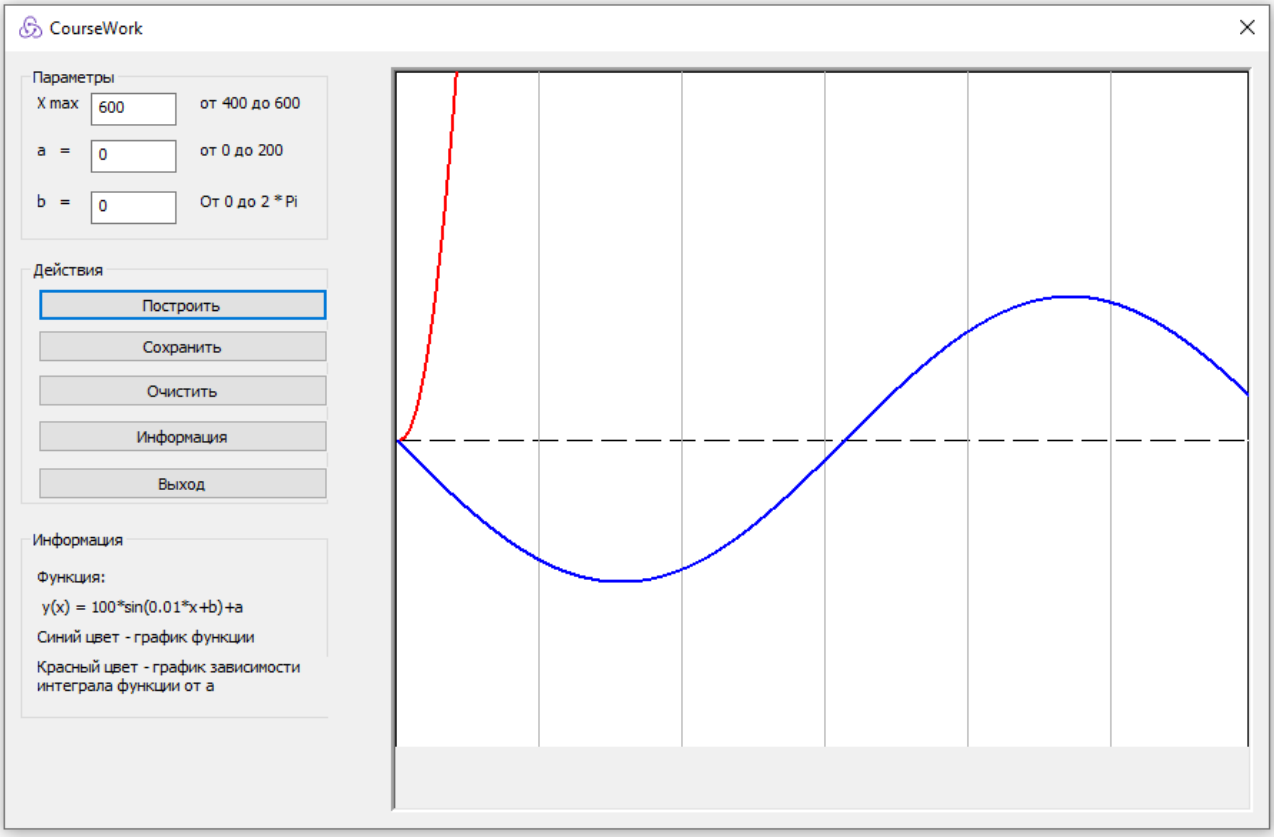
Необходимо разработать оконное приложение Windows, которое будет вычислять определенный интеграл методом прямоугольников, отображать график функции и зависимость интеграла от параметров функции и обеспечивать интерактивный режим изменений параметра функции и сохранения выбранного графика в виде изображения.

Параметры функции:

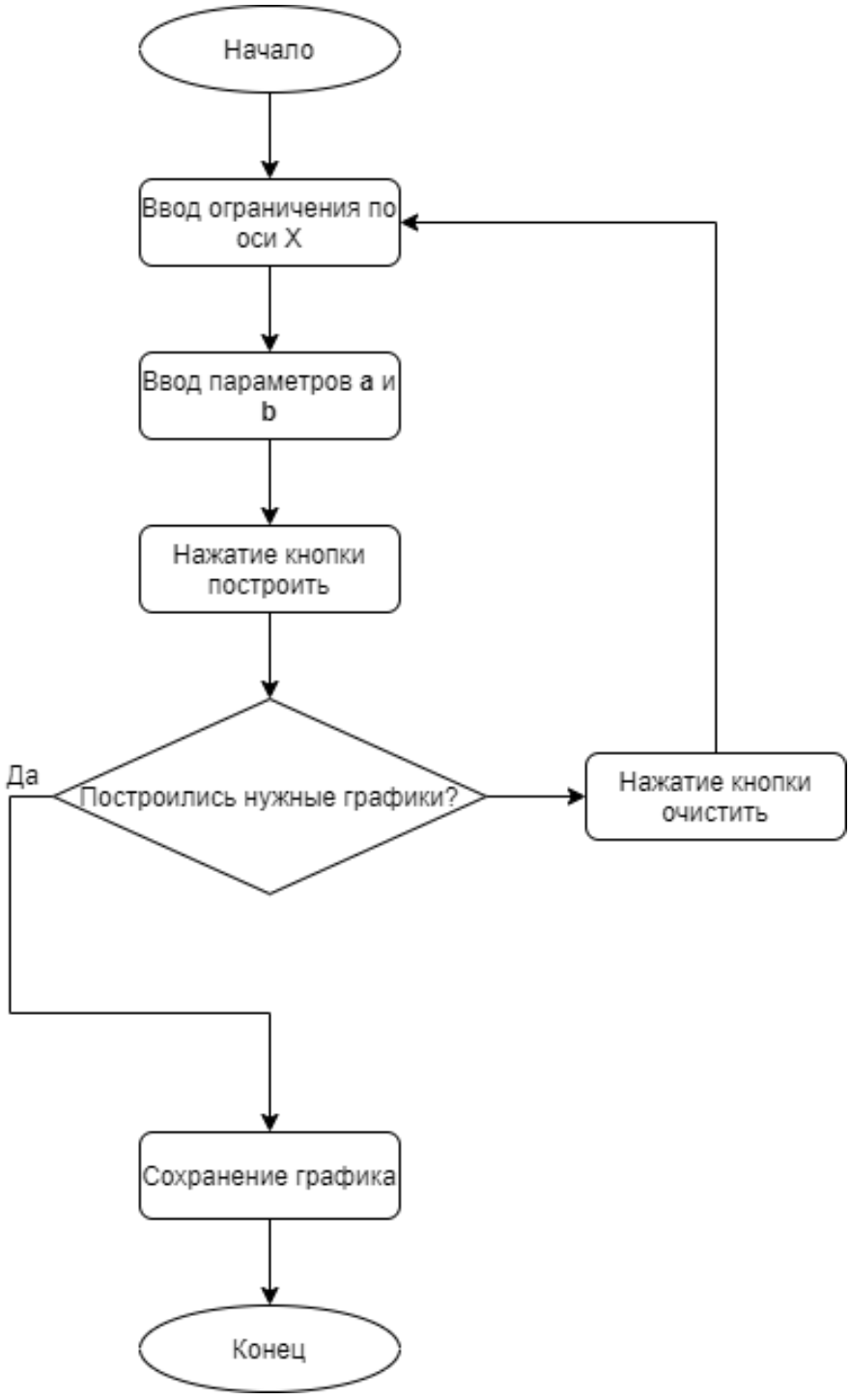
Xmin = 0; Xmax = 400..600; a = 0..200; b = 0..2 * Pi

Словесное описание алгоритма

Пользователь вводит параметры функции в соответствующие поля ввода, после чего нажимает кнопку «Построить». Данная кнопка запускает таймер, который посылает классу рисования сообщение WM_PAINT для перерисовки окна графика. Параметры, введенные пользователем, передаются классу расчёта, который рассчитывает точки графика функции и заполняет ими динамический массив. Этот динамический массив передается в класс рисования, в котором происходит построение нужных графиков. Кнопка «Очистить» обнуляет переменные, закрашивает окно графика белым и перестраивает оси.



Варианты взаимодействия оператора и программы



Выбор и обоснование переменных

CourseWorkDlg.h

CMyGraph m_Paint – переменная класса рисования графика
CMyCalc m_Calc – переменная класса расчёта
int m_Count – ограничение оси X
int m_a – параметр ***a*** функции
int m_b – параметр ***b*** функции

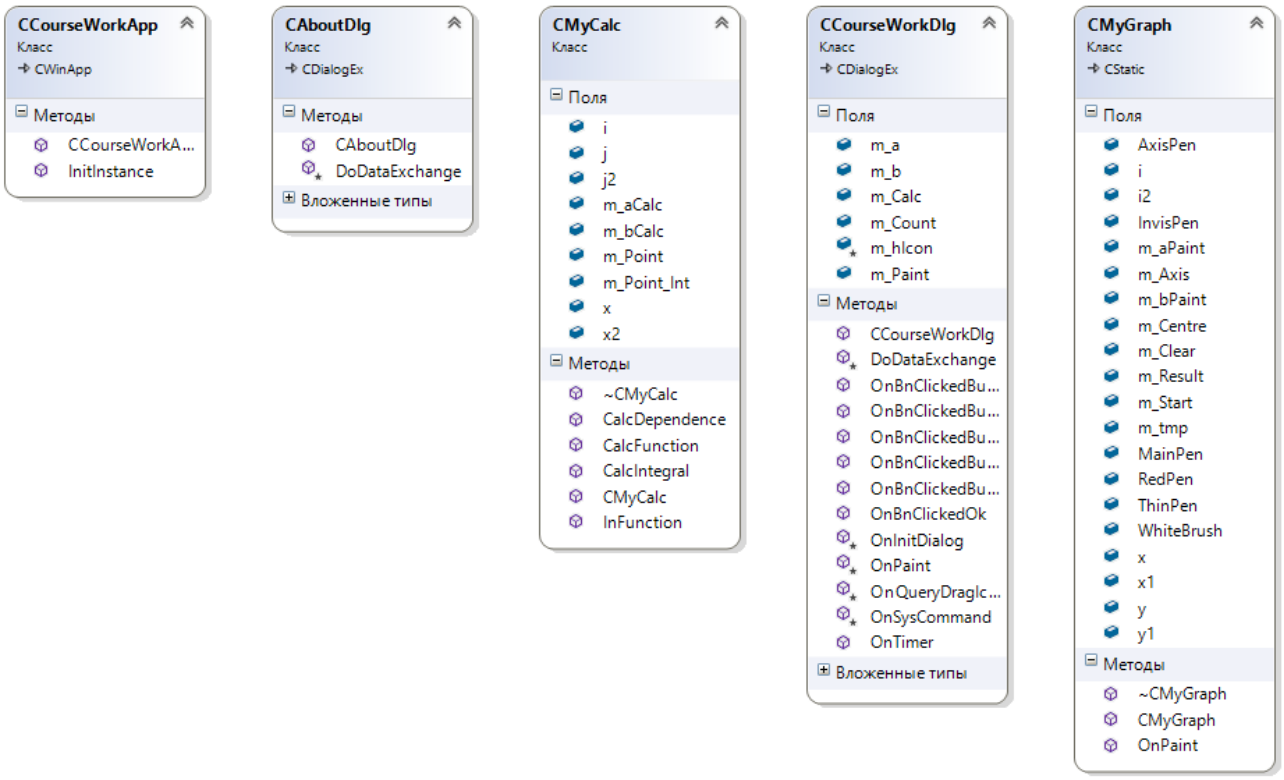
MyGraph.h

int x, x1– аргумент функции
int i, i2 – ограничение по оси X
int m_tmp – ограничение по оси X
int m_aPaint – параметр ***a*** функции
int m_bPaint – параметр ***b*** функции
double m_Centre – центр окна графика
double y, y1 – значение функции в точке
bool m_Start – обнуление переменных
bool m_Clear – очистка окна
bool m_Axis – построение осей
CMyCalc m_Result переменная класса расчёта
CPen MainPen, AxisPen, InvisPen, ThinPen, RedPen – перья для рисования
CBrush WhiteBrush – кисть для рисования

CMyCalc.h

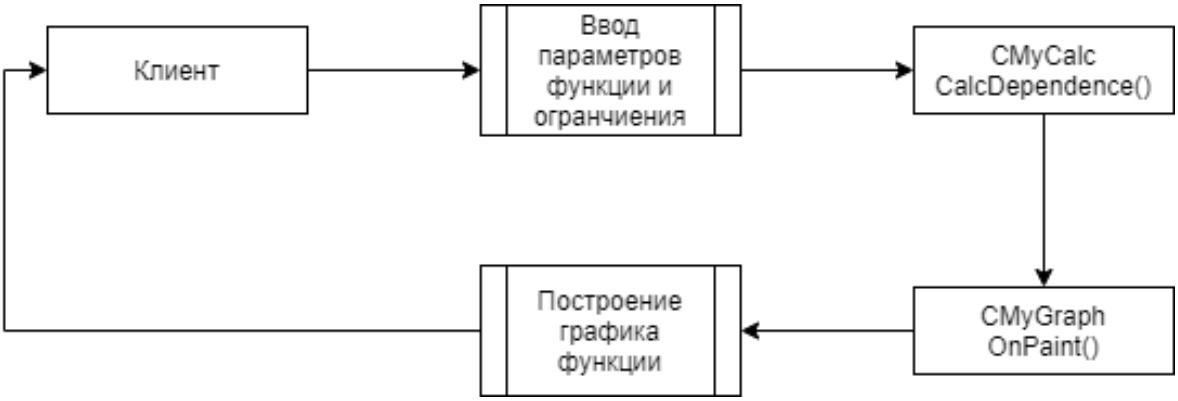
int i – ограничение по оси X
int j, j2 – счётчик
int m_aCalc – параметр ***a*** функции
int m_bCalc – параметр ***b*** функции
double x, x2 – аргумент функции
std::vector<double> m_Point – массив точек графика функции
std::vector<double> m_Point_Int – массив точек графика зависимости

Диаграмма классов



CImage - класс для организации работы с изображениями;
CButton - класс для организации работы с кнопками;
CRect - класс для организации работы с окнами;
CPen - класс для организации работы с перьями;
CString - класс для организации работы со строками;
CPaintDC – класс для рисования в контексте устройства;
CEdit - класс для работы с полями ввода и вывода;
CFileDialog – класс для открытия и сохранение файлов;
CWinApp - базовый класс библиотеки MFC
CStatic- наследник класса CWnd
CRgn – класс для создания регионов (областей);
CWindowDC - наследник CDC
CObject – базовый класс MFC
CWnd – наследник CCmdTarget
CDialog – наследник Cwnd
vector – реализация динамического массива
CCmdTarget – наследник CObject
CDialogEx – наследник CDialogEx

Диаграмма потока данных



СТРУКТУРА ПРОЕКТА

Содержимое папки c:\Users\Admin\Documents\Visual Studio 2015\Projects\CourseWork\CourseWork

```
12.06.2020  13:05             3 181 CourseWork.cpp
12.06.2020  13:05               522 CourseWork.h
13.06.2020  14:55            15 034 CourseWork.rc
12.06.2020  18:26            10 994 CourseWork.vcxproj
12.06.2020  18:26             3 005 CourseWork.vcxproj.filters
13.06.2020  12:38             7 996 CourseWorkDlg.cpp
13.06.2020  14:55             1 141 CourseWorkDlg.h
13.06.2020  14:55    <DIR>          Debug
13.06.2020  13:40               987 MyCalc.cpp
13.06.2020  13:40               451 MyCalc.h
13.06.2020  13:40             3 034 MyGraph.cpp
13.06.2020  12:38               676 MyGraph.h
12.06.2020  13:05             7 255 ReadMe.txt
12.06.2020  18:26    <DIR>          res
13.06.2020  12:23             2 396 resource.h
12.06.2020  13:05               241 stdafx.cpp
12.06.2020  13:05             1 802 stdafx.h
12.06.2020  13:05               351 targetver.h
              18 файлов        238 646 байт
               4 папок  81 530 691 584 байт свободно
```

Инструкция по использованию

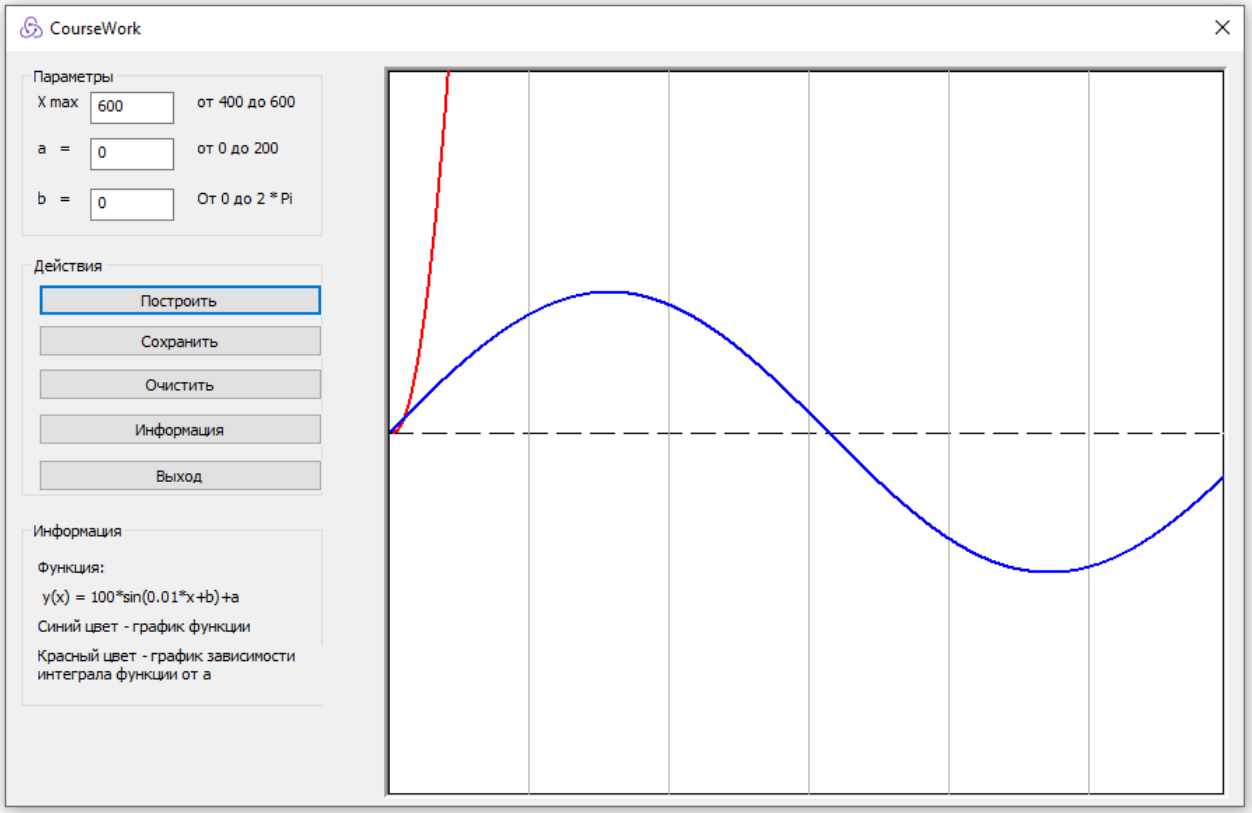
Приложение реализует построение графика функции y(x) = 100*sin (0.01*x+b) +a и графика зависимости интеграла этой функции от параметров a и b.
Приложение работает в ОС Windows (7,8,8.1,10)

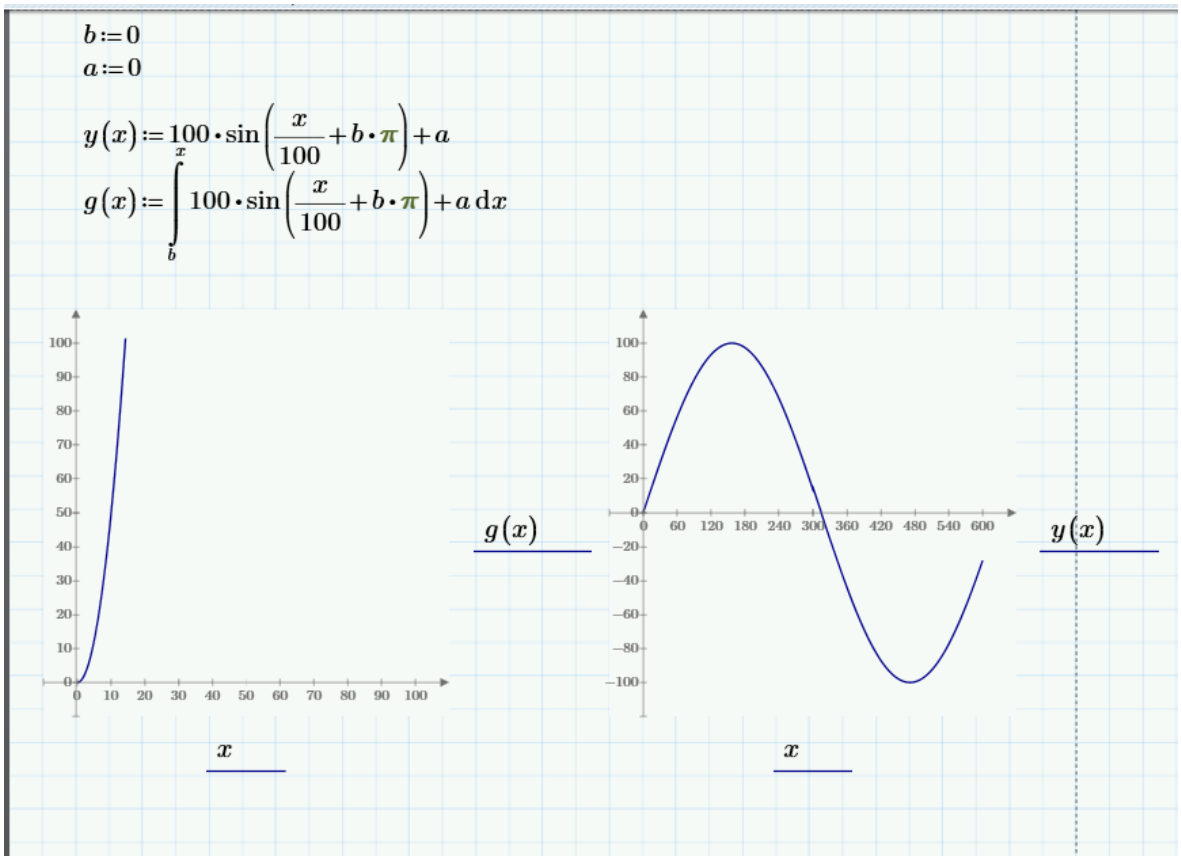
1. Введите параметры функции в поля ввода «a= » и «b= ». Затем укажите ограничение по оси X в соответствующем поле ввода
2. Нажмите кнопку «Построить» и дождитесь окончания отрисовки графика
3. Если нужно отобразить еще один график, измените параметры функции и нажмите кнопку «Построить»
4. Если вам нужно сохранить изображение, нажмите соответствующую кнопку
5. Для очистки окна нажмите кнопку «Очистить»

Ведомость соответствия программы и спецификации

ТРЕБОВАНИЕ	ТРЕБОВАНИЕ ВЫПОЛНЕНО
Проект выполняется в среде Microsoft Visual Studio с использованием компонентов библиотеки Microsoft Foundation Classes (MFC) на языке C++	да
Вычислить определенный интеграл в границах Xmin…Xmax от заданной функции методом прямоугольника.	да
Построить график функции и график зависимости интеграла от параметров функции	да
Обеспечить интерактивный режим изменений параметра функции и сохранения выбранного графика в файле в виде изображения	да
Разработано оконное приложение для системы X86 выполняющая расчеты и графическую визуализацию результатов с поддержкой универсальной кодировки символов Unicode.	да
Разработан графический пользовательский интерфейс.	да
Кнопки управления приложением	да
Анимация графика	да
Язык интерфейса – русский	да
Разработана диаграмма классов	да
Поля ввода данных для расчета	да
Разработана схема потока данных	да
Справка о программе	да
Сброс отображения графиков	да
Результат должен быть подтвержден расчетами и графиками в системе Mathcad	да
Выполнен отчёт по ГОСТу	да

Контрольный пример, сравнение результатов с эталоном MathCad





Вывод

Было разработано оконное приложение Windows в среде Microsoft Visual Studio с использованием библиотеки MFC на языке C++. Программа вычисляет определённый интеграл функции и рисует график самой функции и график зависимости интеграла от параметров функции. Реализовано интерактивное изменение параметров функции.
Приложение занимает на диске 338 КБ, работает в ОС Windows

Недостатки приложения:

Программа вычисляет определенный интеграл только конкретной функции, что ограничивает область применения приложения

Ограничения ввода:

В поля ввода можно водить только целые положительные числа.

Код с комментарием

CourseWorkDlg.h

```
// CourseWorkDlg.h : файл заголовка
//

#pragma once
#include "MyGraph.h"
#include "MyCalc.h"


// диалоговое окно CCourseWorkDlg
class CCourseWorkDlg : public CDialogEx
{
	// Создание
public:
	CCourseWorkDlg(CWnd* pParent = NULL);	// стандартный конструктор

// Данные диалогового окна
#ifdef AFX_DESIGN_TIME
	enum { IDD = IDD_COURSEWORK_DIALOG };
#endif

protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// поддержка DDX/DDV


// Реализация
protected:
	HICON m_hIcon;

	// Созданные функции схемы сообщений
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	DECLARE_MESSAGE_MAP()
public:
	//functions
	afx_msg void OnTimer(UINT_PTR nIDEvent);
	afx_msg void OnBnClickedButton2();
	afx_msg void OnBnClickedButton3();
	afx_msg void OnBnClickedOk();
	afx_msg void OnBnClickedButton4();
	afx_msg void OnBnClickedButton5();
	afx_msg void OnBnClickedButton1();

	// class variable
	CMyGraph m_Paint;
	CMyCalc m_Calc;

	// function parameters
	int m_Count;
	int m_a;
	int m_b;


};
```

CourseWorkDlg.cpp

```
// CourseWorkDlg.cpp : файл реализации
//

#include "stdafx.h"
#include "CourseWork.h"
#include "CourseWorkDlg.h"
#include "afxdialogex.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif
#define IDC_TIMER 9181

#include <initguid.h>
DEFINE_GUID(ImageFormatBMP, 0xb96b3cab, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8,
0x1e, 0xf3, 0x2e);
DEFINE_GUID(ImageFormatEMF, 0xb96b3cac, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8,
0x1e, 0xf3, 0x2e);
DEFINE_GUID(ImageFormatWMF, 0xb96b3cad, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8,
0x1e, 0xf3, 0x2e);
DEFINE_GUID(ImageFormatJPEG, 0xb96b3cae, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8,
0x1e, 0xf3, 0x2e);
DEFINE_GUID(ImageFormatPNG, 0xb96b3caf, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8,
0x1e, 0xf3, 0x2e);
DEFINE_GUID(ImageFormatGIF, 0xb96b3cb0, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8,
0x1e, 0xf3, 0x2e);
DEFINE_GUID(ImageFormatTIFF, 0xb96b3cb1, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8,
0x1e, 0xf3, 0x2e);
// Диалоговое окно CAboutDlg используется для описания сведений о приложении

class CAboutDlg : public CDialogEx
{
public:
	CAboutDlg();

// Данные диалогового окна
#ifdef AFX_DESIGN_TIME
	enum { IDD = IDD_ABOUTBOX };
#endif

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // поддержка DDX/DDV

// Реализация
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialogEx(IDD_ABOUTBOX)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialogEx::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
END_MESSAGE_MAP()


// диалоговое окно CCourseWorkDlg
```

```
CCourseWorkDlg::CCourseWorkDlg(CWnd* pParent /*=NULL*/)
	: CDialogEx(IDD_COURSEWORK_DIALOG, pParent)

	, m_Count(0)
	, m_a(0)
	, m_b(0)
{
	m_hIcon = AfxGetApp()->LoadIcon(IDI_ICON1);
}

void CCourseWorkDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialogEx::DoDataExchange(pDX);

	DDX_Text(pDX, IDC_EDIT1, m_Count);
	DDX_Text(pDX, IDC_EDIT2, m_a);
	DDX_Text(pDX, IDC_EDIT3, m_b);
}

BEGIN_MESSAGE_MAP(CCourseWorkDlg, CDialogEx)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_WM_TIMER()
	ON_BN_CLICKED(IDC_BUTTON1, &CCourseWorkDlg::OnBnClickedButton1)
	ON_BN_CLICKED(IDC_BUTTON2, &CCourseWorkDlg::OnBnClickedButton2)
	ON_BN_CLICKED(IDC_BUTTON3, &CCourseWorkDlg::OnBnClickedButton3)
	ON_BN_CLICKED(IDOK, &CCourseWorkDlg::OnBnClickedOk)
	ON_BN_CLICKED(IDC_BUTTON4, &CCourseWorkDlg::OnBnClickedButton4)
	ON_BN_CLICKED(IDC_BUTTON5, &CCourseWorkDlg::OnBnClickedButton5)
END_MESSAGE_MAP()


// обработчики сообщений CCourseWorkDlg

BOOL CCourseWorkDlg::OnInitDialog()
{
	CDialogEx::OnInitDialog();

	// Добавление пункта "О программе..." в системное меню.

	// IDM_ABOUTBOX должен быть в пределах системной команды.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		BOOL bNameValid;
		CString strAboutMenu;
		bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
		ASSERT(bNameValid);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Задает значок для этого диалогового окна.  Среда делает это автоматически,
	//  если главное окно приложения не является диалоговым
	SetIcon(m_hIcon, TRUE);			// Крупный значок
```

```
	SetIcon(m_hIcon, FALSE);		// Мелкий значок

	// TODO: добавьте дополнительную инициализацию
	m_Paint.SubclassDlgItem(IDC_STATIC_GRAPH,this);
	// Pen and brush consciousness
	m_Paint.MainPen.CreatePen(PS_SOLID, 2, RGB(0, 0, 255));
	m_Paint.AxisPen.CreatePen(PS_DASH, 1, RGB(0, 0, 0));
	m_Paint.InvisPen.CreatePen(PS_NULL, 1, RGB(255, 255, 255));
	m_Paint.ThinPen.CreatePen(PS_SOLID, 1, RGB(180, 180, 180));
	m_Paint.RedPen.CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
	m_Paint.WhiteBrush.CreateSolidBrush(NULL_BRUSH);
	// Axis drawing
	m_Paint.m_Axis = true;
	return TRUE;  // возврат значения TRUE, если фокус не передан элементу управления
}

void CCourseWorkDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialogEx::OnSysCommand(nID, lParam);
	}
}

// При добавлении кнопки свертывания в диалоговое окно нужно воспользоваться приведенным
ниже кодом,
//  чтобы нарисовать значок.  Для приложений MFC, использующих модель документов или
представлений,
//  это автоматически выполняется рабочей областью.

void CCourseWorkDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // контекст устройства для рисования

		SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()),
0);

		// Выравнивание значка по центру клиентского прямоугольника
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Нарисуйте значок
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialogEx::OnPaint();
	}

}

// Система вызывает эту функцию для получения отображения курсора при перемещении
//  свернутого окна.
HCURSOR CCourseWorkDlg::OnQueryDragIcon()
```

```
{
	return static_cast<HCURSOR>(m_hIcon);
}



void CCourseWorkDlg::OnTimer(UINT_PTR nIDEvent)
{
	// TODO: добавьте свой код обработчика сообщений или вызов стандартного
	// Redrawing the function graph window
	m_Paint.Invalidate();

	CDialogEx::OnTimer(nIDEvent);
}


void CCourseWorkDlg::OnBnClickedButton1()
{
	// Reading variables
	UpdateData(TRUE);
	m_Paint.m_Start = true;
	m_Paint.m_tmp = m_Count;
	m_Paint.m_aPaint = m_a;
	m_Paint.m_bPaint = m_b;
	// Start timer
	SetTimer(9181, 1, NULL);
	// TODO: добавьте свой код обработчика уведомлений
}


void CCourseWorkDlg::OnBnClickedButton2()
{
	// Clear the graph window
	m_Paint.m_Clear = true;

	// TODO: добавьте свой код обработчика уведомлений
}


void CCourseWorkDlg::OnBnClickedButton3()
{
	CWnd *pWn = GetDlgItem(IDC_STATIC_GRAPH);
	if (!pWn)
		return;
	CWindowDC winDC(pWn);
	CRect rc;
	pWn->GetClientRect(&rc);
	CDC memDC;
	memDC.CreateCompatibleDC(&winDC);
	CBitmap bitMap;
	bitMap.CreateCompatibleBitmap(&winDC, rc.Width(), rc.Height());
	HGDIOBJ pOld = memDC.SelectObject(&bitMap);
	memDC.FillSolidRect(&rc, RGB(0, 255, 0));
	memDC.BitBlt(0, 0, rc.Width(), rc.Height(), &winDC, 0, 0, NOTSRCCOPY);
	memDC.SelectObject(pOld);

	static TCHAR szFilter[] = _T("BMP Files (*.bmp)|*.bmp|")
		_T("PNG Files (*.png)|*.png|GIF Files (*.gif)|*.gif|")
		_T("JPG Files (*.jpg)|*.jpg|All Files (*.*)|*.*||");
	CFileDialog dlg(FALSE, _T(".bmp"), NULL, 6UL, szFilter);
	if (IDOK == dlg.DoModal())
	{
		CImage image;
		image.Attach(HBITMAP(bitMap));
		CString strFull = dlg.GetOFN().lpstrFile;
```

```
                HRESULT hr;
                if (-1 != strFull.Find(_T(".png")))
                        hr = image.Save(strFull, ImageFormatPNG);
                else if (-1 != strFull.Find(_T(".jpg")))
                        hr = image.Save(strFull, ImageFormatJPEG);
                else if (-1 != strFull.Find(_T(".gif")))
                        hr = image.Save(strFull, ImageFormatGIF);
                else if (-1 != strFull.Find(_T(".bmp")))
                        hr = image.Save(strFull, ImageFormatBMP);
                else
                {
                        strFull += _T(".bmp");
                        hr = image.Save(strFull, ImageFormatBMP);
                }

                if (FAILED(hr))
                {
                        CString strErr;
                        strErr.Format(L" Couldn't Save File: %s, %x ", (LPCTSTR)strFull, hr);
                        AfxMessageBox(strErr, MB_OK | MB_ICONERROR);
                }
        }
        // TODO: добавьте свой код обработчика уведомлений
}


void CCourseWorkDlg::OnBnClickedOk()
{
        // TODO: добавьте свой код обработчика уведомлений
        CDialogEx::OnOK();
}


void CCourseWorkDlg::OnBnClickedButton4()
{
        MessageBox(L"Приложение реализует построение графика функции y(x) = 100*sin
(0.01*x+b) +a и графика зависимости интеграла этой функции от параметров a и
b.\nПриложение работает в ОС Windows(7, 8, 8.1, 10)\n\n1.Введите параметры функции в поля
ввода «a = » и «b = ».Затем укажите ограничение по оси X в соответствующем поле
ввода\n2.Нажмите кнопку «Построить» и дождитесь окончания отрисовки графика\n3.Если нужно
отобразить еще один график, измените параметры функции и нажмите кнопку
«Построить»\n4.Если вам нужно сохранить изображение, нажмите соответствующую
кнопку\n5.Для очистки окна нажмите кнопку «Очистить»\n\nПриятного пользования!",
L"Информация");
        // TODO: добавьте свой код обработчика уведомлений
}


void CCourseWorkDlg::OnBnClickedButton5()
{

        // TODO: добавьте свой код обработчика уведомлений
}
```

MyGraph.h

```
#pragma once
#include "MyCalc.h"

// CMyGraph

class CMyGraph : public CStatic
{
	DECLARE_DYNAMIC(CMyGraph)

public:
	CMyGraph();
	virtual ~CMyGraph();

protected:
	DECLARE_MESSAGE_MAP()
public:
	afx_msg void OnPaint();

	// function variables
	int x = 0, x1 = 0;
	int i = 0, i2 = 0;
	int m_tmp = 0;
	int m_aPaint = 0;
	int m_bPaint = 0;

	// function variables
	double m_Centre = 0;
	double y = 0, y1 = 0;

	// logical variable
	bool m_Start = false;
	bool m_Clear = false;
	bool m_Axis = false;

	// class variable
	CMyCalc m_Result;

	// pen and brush
	CPen MainPen, AxisPen, InvisPen, ThinPen,RedPen;
	CBrush WhiteBrush;

};
```

MyGraph.cpp

```
// MyGraph.cpp: файл реализации
//

#include "stdafx.h"
#include "CourseWork.h"
#include "MyGraph.h"


// CMyGraph

IMPLEMENT_DYNAMIC(CMyGraph, CStatic)

CMyGraph::CMyGraph()
{

}

CMyGraph::~CMyGraph()
{
}


BEGIN_MESSAGE_MAP(CMyGraph, CStatic)
	ON_WM_PAINT()
END_MESSAGE_MAP()



// обработчики сообщений CMyGraph




void CMyGraph::OnPaint()
{
	// device context for painting
	CPaintDC dc(this);
	CRect rc;
	GetClientRect(&rc);
	CRgn rgn;
	rgn.CreateRectRgn(rc.left, rc.top, rc.right, rc.bottom);
	dc.SelectClipRgn(&rgn);

	if (m_Start)
	{
		// reset variables
		x = 0;
		x1 = 0;
		y = 0;
		y1 = 0;
		i = 0;
		i2 = 0;
		m_Result.x = 0;
		m_Result.x2 = 0;
		m_Result.j = 0;
		m_Result.j2 = 0;
		m_Result.i = 0;
		m_Start = false;
		///
	}
	if (m_Clear)
	{
		// clearing the window and the construction of the axes
```

```
        dc.Rectangle(rc.left, rc.top, rc.right, rc.bottom);
        dc.SelectObject(AxisPen);
        dc.MoveTo(rc.left, m_Centre);
        dc.LineTo(rc.right, m_Centre);
        dc.SelectObject(ThinPen);
        dc.MoveTo(100, rc.top);
        dc.LineTo(100, rc.bottom);
        dc.MoveTo(200, rc.top);
        dc.LineTo(200, rc.bottom);
        dc.MoveTo(300, rc.top);
        dc.LineTo(300, rc.bottom);
        dc.MoveTo(400, rc.top);
        dc.LineTo(400, rc.bottom);
        dc.MoveTo(500, rc.top);
        dc.LineTo(500, rc.bottom);
        dc.MoveTo(600, rc.top);
        dc.LineTo(600, rc.bottom);
        dc.SelectObject(InvisPen);
        m_Clear = false;
        ///
}
// sending data to the calculation class
m_Result.i = m_tmp;
m_Result.m_aCalc = m_aPaint;
m_Result.m_bCalc = m_bPaint;
m_Result.CalcFunction();
m_Result.CalcDependence();
///
m_Centre = (rc.bottom - rc.top) / 2;
if (m_Axis)
{
        // clearing the window and the construction of the axes
        dc.Rectangle(rc.left, rc.top, rc.right, rc.bottom);
        dc.SelectObject(AxisPen);
        dc.MoveTo(rc.left, m_Centre);
        dc.LineTo(rc.right, m_Centre);
        dc.SelectObject(ThinPen);
        dc.MoveTo(100, rc.top);
        dc.LineTo(100, rc.bottom);
        dc.MoveTo(200, rc.top);
        dc.LineTo(200, rc.bottom);
        dc.MoveTo(300, rc.top);
        dc.LineTo(300, rc.bottom);
        dc.MoveTo(400, rc.top);
        dc.LineTo(400, rc.bottom);
        dc.MoveTo(500, rc.top);
        dc.LineTo(500, rc.bottom);
        dc.MoveTo(600, rc.top);
        dc.LineTo(600, rc.bottom);
        dc.SelectObject(InvisPen);
        m_Axis = false;
        ///
}
if (i < m_Result.i)
{
        // plotting a function
        dc.SelectObject(MainPen);
        if (i == 0)
        {
                dc.SelectObject(InvisPen);
        }
        dc.MoveTo(x, y + m_Centre);
        y = m_Result.m_Point[i];
        x += 1;
        dc.LineTo(x, y + m_Centre);
```

```
			i++;
			///
		}
		if (i2 < m_Result.i)
		{
			// building the dependence of the integral on the parameter
			dc.SelectObject(RedPen);
			if (i2 == 0)
			{
				dc.SelectObject(InvisPen);
			}
			dc.MoveTo(x1, y1 + m_Centre);
			y1 = m_Result.m_Point_Int[i2] ;
			x1 += 3;
			dc.LineTo(x1, y1 + m_Centre);
			i2++;
			///
		}

		// TODO: добавьте свой код обработчика сообщений
		// Не вызывать CStatic::OnPaint() для сообщений рисования
}
```

MyCalc.h

```
#pragma once
#include <vector>
class CMyCalc
{
public:
	CMyCalc();
	~CMyCalc();
// functions:
	void CalcFunction();
	void CalcDependence();
	double CalcIntegral(double a, double b, int n);
	double InFunction(double x, double a);
// variables:
	int i = 0;
	int j = 0, j2 = 0;
	int m_aCalc = 0;
	int m_bCalc = 0;
	double x = 0, x2 = 0;
// arrays of points
	std::vector<double> m_Point;
	std::vector<double> m_Point_Int;


};
```

MyCalc.cpp

```
#include "stdafx.h"
#include "MyCalc.h"
#include <vector>

using namespace std;

CMyCalc::CMyCalc()
{
}


CMyCalc::~CMyCalc()
{
}
// calculating points of the original function
void CMyCalc::CalcFunction()
{

	m_Point.resize(i);

	while(j < i)
	{
		x++;
		m_Point[j] = -100*sin(x/100+m_bCalc*3.1415)+m_aCalc;
		j++;
	}
	x = 0;
	j = 0;

}

// the method of rectangles
double CMyCalc::CalcIntegral(double a, double b, int n)
{
	double h = (b-a)/n;
	double result = 0;
	for (int i = 0; i < n;i++)
	{

		result += InFunction(a + h*(i + 0.5), a);

	}
	result *= h;
	return result;
}

// the original function
double CMyCalc::InFunction(double x, double a)
{
	return  100 * sin(x / 100 + m_bCalc*3.1415) + a;

}

// calculating the points of dependence of the integral on the parameter
void CMyCalc::CalcDependence()
{
	m_Point_Int.resize(i);
	for (int a = 0; a < i; a++)
	{
		m_Point_Int[a] = CalcIntegral(a, m_bCalc,100);
	}
}
```