МИНОБРНАУКИ РОССИИ

Государственное образовательное учреждение высшего образования

«Ярославский государственный технический университет»

Кафедра «Информационных систем и технологий»

Контрольная работа была защищена

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель доцент кафедры "Информационные системы и технологии", к.ф-м.н. Раухваргер А.Б. \_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_г.

**Разработка консольных приложений и графическое приложение Windows API**

Контрольная работа по дисциплине

“Информатика”

Выполнил

Студент, Гулиев С.А.

Группа ДСИТ-17

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_г.

2019

# **Разработать класс для хранения строки и программу для тестирования работы методов класса.**

**Требования к программе:**

Класс должен содержать:

* объект класса string с уровнем инкапсуляции private для хранения строки символов
* конструктор на основе заданной строки типа string;
* конструктор на основе заданной c-строки, копирующий конструктор;
* конструктор на основе имени файла, заданного c-строкой;
* метод получения хранимой строки в виде объекта string;
* метод получения хранимой строки в виде c-строки;
* метод установки хранимой строки из заданной c-строки;
* метод установки хранимой строки из объекта string;
* метод загрузки хранимой строки из файла;
* метод сохранения хранимой строки в файл.

## Описание используемых функций и алгоритмов

**Конструктор** — это особый тип метода класса, который автоматически вызывается при создании объекта этого класса. Конструкторы обычно используются для инициализации переменных-членов класса соответствующими значениями, которые предоставлены по умолчанию или пользователем, ну или для выполнения любых шагов настройки, необходимых для используемого класса (например, открыть определенный файл или базу данных).

В отличии от обычных методов, конструкторы имеют определенные правила, которые относятся к их названиям:

Конструкторы всегда должны иметь то же имя, что и класс (учитывается верхний и нижний регистры).

Конструкторы не имеют типа возврата (даже void-а).

**Классы -** в мире объектно-ориентированного программирования типы данных могут не только содержать данные, но и функции, которые будут работать с этими данными. Для определения такого типа данных в C++ используется ключевое слово **class**. Использование ключевого слова class определяет новый пользовательский тип данных — **класс**.

**Методы** **класса**

Помимо хранения данных, классы также могут содержать функции! Функции, определенные внутри класса, называются **функциями-членами** или **методами**. Методы могут быть определены внутри или вне класса.

**ofstream** - для вывода (записи) данных в файл;

**ifstream** - для ввода (чтения) данных из файла;

**Инструкция для пользователя:**

Запустить консольное приложение.

1. В окно приложения ввести текст.
2. По окончанию работы приложения текст, введенный пользователем сохранится в файл, и отобразит его на экране.
3. После повторного использования, ранее введенный текст выведется на экран (Рисунок 1)

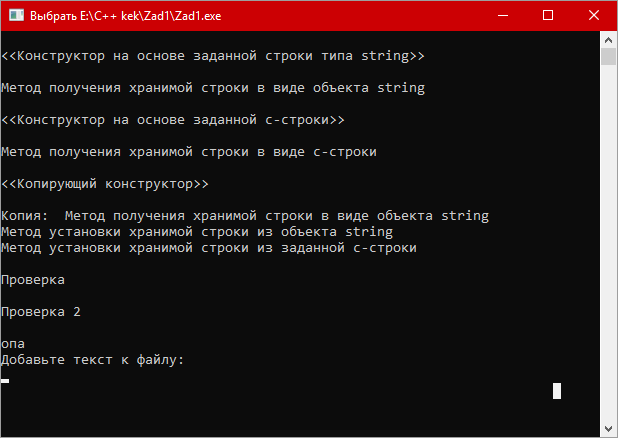


Рисунок 1.Пример использование конструкторов и методов

Код конструкторов и методы, их вызовы

#include <string>

#include <iostream>

#include <fstream>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

using namespace std;

class MyClass

{

private:

string str; ;

public:

char\* get\_char(){

return (char\* )str.c\_str();

};

MyClass(string qstr){

str=qstr;

cout<<"\n конструктор на основе заданной строки типа string\n"<<endl;

};

MyClass(char\* qch){

str=qch;

cout<<"\n конструктор на основе с-строки \n"<<endl;

};

MyClass(const MyClass &other){

str=other.str;

cout<<"\n копирующий конструктор \n"<<endl;

cout<<"Êîïèÿ: ";

};

MyClass(int null, char\* filename);{

string i;

char j;

ifstream f(filename);

while(!f.eof())

{

j=f.get();

if(f.eof())break;

i+=j;

}

f.close();

str=i;

}

string get\_string(){

return str;

};

void set\_string(string wstr){

str=wstr;

};

void set\_char(char\* wch){

str=wch;

};

void file\_load(int null, char\* filename) {

string i;

string j;

ifstream f(filename);

while(!f.eof())

{

j=f.get();

if(f.eof())break;

i+=j;

}

f.close();

str=i;

};

void file\_save(string Input, char\* filename) {

str+="\n"+Input;

ofstream f(filename);

int i = 0;

while(str[i])

{

f.put(str[i]);

i++;

}

f.close();

};

};

}

int main(int argc, char\*\* argv)

{

string a = "Метод получения хранимой строки в виде объекта string";

MyClass text1(a);

cout<<text1.get\_string()<<endl;

MyClass text2("Метод получения хранимой строки в виде c-строки");

cout<<text2.get\_char()<<endl;

MyClass text3(text1);

cout<<text3.get\_string()<<endl;

text1.set\_string("Метод установки хранимой строки из объекта string");

cout<<text1.get\_string()<<endl;

text2.set\_char("Метод установки хранимой строки из заданной c-строки\n");

cout<<text2.get\_char()<<endl;

MyClass file1(0, "Test.txt");

cout<<file1.get\_string()<<endl;

cout<<"Добавте текст к файлу: "<<endl;

string b;

getline(cin, b);// Чтение входного потока вместе с пробелом Функция **getline** предназначена для ввода данных из потока, например, для ввода данных из консольного окна

file1.file\_save(b, "Test.txt");

file1.file\_load(0, "Test.txt");

cout<<file1.get\_string()<<endl;

return 0;

**}**

## **Описание кода**

class MyClass **// определение класса с именем MyClass**

{

private:

string str**; //объект класса string с уровнем инкапсуляции private для хранения строки символов**

Тело класса содержит: **private** и **public**. Ключевой особенностью объектно-ориентированного программирования является возможность сокрытия данных. Этот термин понимается в том смысле, что данные заключены внутри класса и защищены от несанкционированного доступа функций, расположенных вне класса. Если необходимо защитить какие - либо данные, то их помещают внутрь класса с ключевым словом **private**. Такие данные доступны только внутри класса. Данные, описанные с ключевым словом **public**, напротив, доступны за пределами класса.

Для того чтобы получить доступ к методу класса, необходимо использовать операцию точки (.), связывающую метод с именем объекта

**text3.get\_string -** метод установки хранимой строки из заданной c-строки;

**f.eof** - Функция **feof** проверяет, достигнут ли конец файла, связанного с потоком, через параметр **filestream**. Возвращается значение, отличное от нуля, если конец файла был действительно достигнут.

Функция **fgets()** считывает до num-1 символов из файла и помещает их в массив символов, на который указывает **str**. Символы считываются до тех пор, пока не встретится символ «новая строка», EOF или до достижения указанного предела. По окончании считывания в массив **str** сразу после последнего считанного символа помещается нулевой символ. Символ «новая строка» при считывании будет сохранен и станет частью массива **str.**

## **Заключение**

В данной работе мы научились создавать класса, использовать конструкторы, методы загрузки и сохранения файлов, а также вызывать через основной файл приложения.

# **Разработать графическое приложение Windows API с функционалом простого текстового редактора.**

Рекомендуется использование в качестве промежуточного буфера между файлом и элементом управления “Edit” объекта класса, созданного в пункте 1.

## **Инструкция для пользователя**

Открыть текстовый редактор «Text.exe», Рисунок 2.

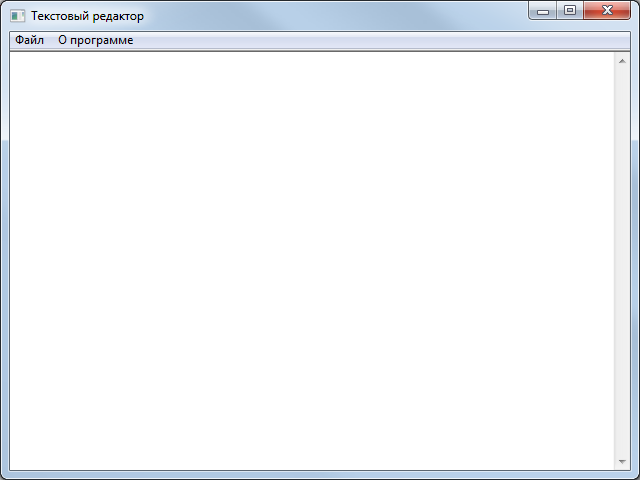
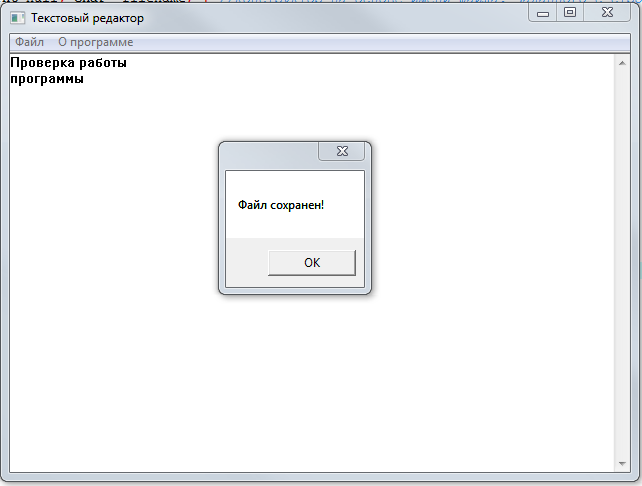


Рисунок 2.Текстовый редактор

Ввести нужный нам текст, далее выбираем сохранить и прописываем имя файла и путь сохранения, файл автоматически сохранится с расширением (\*.txt), Рисунок 3.





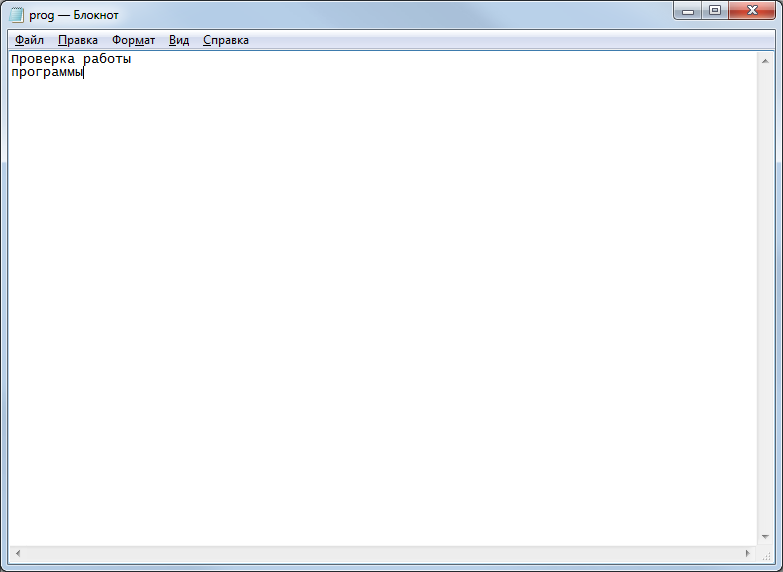


Рисунок 3.Создание и сохранение файла в txt

1. Также при нажатии «Файл» -> «Открыть», можно выбрать текстовый файл с расширением (\*.txt) для просмотра содержимого, Рисунок 4.

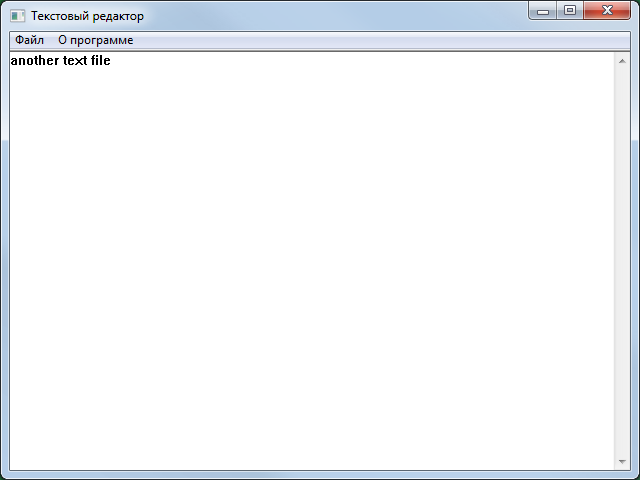


Рисунок 4.Открытие файла .txt

1. При нажатии в «Файл» -> «Выход» окно текстового редактора закроется, но при нажатии «Выход» приложение не предлагает пользователю произвести сохранение файла, поэтому перед выходом, нажмите «Файл» -> «Сохранить» и введите имя файла, путь и нажмите сохранить, чтобы не потерять свои записи.

## **Код программы**

#include <windows.h>

#include <iostream>

#include "bufstr.hpp"

/\*Глобальные переменные\*/

HWND hwndEdit; // Создание переменной для окна редактирования текста

OPENFILENAME ofn;

/\*указатель на структуру OPENFILENAME

в которой содержится информация для инциализации диалога.

Содержит информацию о выбранном файле для GetOpenFileName и GetSaveFileName \*/

char szFile[100000]= {0};

char szFile\_name[10000]= {0};

bool openfs = false;

MyClass files;

LRESULT CALLBACK WndProc(HWND hwnd, UINT Message, WPARAM wParam, LPARAM lParam) {

ofn.lpstrFile = szFile;

ofn.nMaxFile =sizeof(szFile

ofn.lpstrFileTitle = szFile\_name;

ofn.nMaxFileTitle = sizeof(szFile\_name);

ofn.lStructSize = sizeof(ofn);

ofn.Flags = OFN\_EXPLORER|OFN\_FILEMUSTEXIST|OFN\_PATHMUSTEXIST;

ofn.hwndOwner = NULL;

ofn.lpstrFilter = "Текстовый файл (\*.txt)\0\*.txt\0";

ofn.nFilterIndex = 1;

ofn.lpstrInitialDir = NULL;

ofn.lpstrDefExt = ("txt");

switch(Message) {

case WM\_SIZE:

{

MoveWindow(hwndEdit, 0, 0, LOWORD(lParam), HIWORD(lParam) , TRUE); //Изменяет положение и размеры указанного окна

break;

}

case WM\_COMMAND:{

switch (LOWORD(wParam)) {

case 101:{ //id которое задал в меню, на кнопку "ОТКРЫТЬ"

if(GetOpenFileName(&ofn)!=0)

{

files.file\_load(0, szFile);

SetWindowText(hwndEdit,files.get\_char());

openfs=true;//Для реализации кнопки Сохранить

}

break;

}

case 102:{ //id которое задал в меню, на кнопку "ОТКРЫТЬ"

if(openfs==false)

{

if(GetSaveFileName(&ofn)!=0)

{

char textEditt[10000];

GetWindowText(hwndEdit,textEditt,10000);

files.set\_char(textEditt);

files.file\_save(0, szFile);

openfs=true;

MessageBox(hwnd, TEXT("Файл сохранен"),TEXT(""), 0);

}

}

else

{

char textEditt[10000];

GetWindowText(hwndEdit,textEditt,10000 );

files.set\_char(textEditt);

files.file\_save(0, szFile);

MessageBox(hwnd, TEXT("Файл сохранен!" ),TEXT(""), 0);

}

break;

}

case 103:/\*Кнопка Сохранить как\*/

{

if(GetSaveFileName(&ofn)!=0)

{

char textEditt[10000];

GetWindowText(hwndEdit,textEditt,10000 );

files.set\_char(textEditt);

files.file\_save(0, szFile);

MessageBox(hwnd, TEXT("Файл сохранен!"),TEXT("szFile"), 0);

}

break;

}

case 104: /\*Нажатие на "Выход"\*/

{

SendMessage(hwnd, WM\_CLOSE, 0, 0);

break;

}

case 105:{ //id которое задал в меню

MessageBox(hwnd,"Текстовый редактор, студента группы ДСИТ-17а - Гулиев С.А.","О программе",MB\_OK);

break;

}

}

}

default: // если не сработал ниодин case то выполняется default

return DefWindowProc(hwnd, Message, wParam, lParam);

}

return 0;

switch(Message) { //обработка сообщений

/\* Upon destruction, tell the main thread to stop \*/

case WM\_DESTROY: { // сообщение выводится когда уничтожается окно первого уровня

PostQuitMessage(0);

break;

}

}

return 0;

}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

WNDCLASSEX wc;

HWND hwnd;

MSG msg;

memset(&wc,0,sizeof(wc));

wc.cbSize = sizeof(WNDCLASSEX);

wc.lpfnWndProc = WndProc;

wc.hInstance = hInstance;

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);

wc.hbrBackground = (HBRUSH)(COLOR\_WINDOW);

wc.lpszClassName = "WindowClass"; // имя класса

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wc.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

if(!RegisterClassEx(&wc)) {

MessageBox(NULL, "Window Registration Failed!","Error!",MB\_ICONEXCLAMATION|MB\_OK);

return 0;

}

hwnd = CreateWindowEx(WS\_EX\_CLIENTEDGE,"WindowClass","Текстовый редактор",WS\_VISIBLE|WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, /\* x \*/

CW\_USEDEFAULT, /\* y \*/

640, /\* width \*/

480, /\* height \*/

NULL,NULL,hInstance,NULL);

hwndEdit = CreateWindowEx(NULL,"Edit","", WS\_VISIBLE|WS\_CHILD|WS\_VSCROLL|ES\_MULTILINE,

NULL, /\* x \*/

NULL, /\* y \*/

600, /\* width \*/

420, /\* height \*/

hwnd,NULL,hInstance,NULL);

HMENU FirstMenu = CreateMenu();

HMENU SecondMenu = CreatePopupMenu();

AppendMenu(FirstMenu, MF\_STRING | MF\_POPUP, (UINT\_PTR)SecondMenu, "Файл");

AppendMenu(SecondMenu, MF\_STRING, 101, "Открыть");

AppendMenu(SecondMenu, MF\_STRING, 102, "Сохранить");

AppendMenu(SecondMenu, MF\_STRING, 103, "Сохранить как");

AppendMenu(SecondMenu, MF\_STRING, 104, "Выход");

AppendMenu(FirstMenu, MF\_STRING, 105, "О программе");

SetMenu(hwnd, FirstMenu);

SetMenu(hwnd, SecondMenu);

if(hwnd == NULL) {

MessageBox(NULL, "Window Creation Failed!","Error!",MB\_ICONEXCLAMATION|MB\_OK); // сообщение пользователю

return 0;

}

// цикл сообщений

while(GetMessage(&msg, NULL, 0, 0) > 0) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

## Описание кода

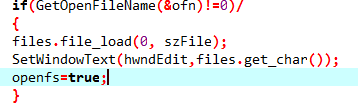


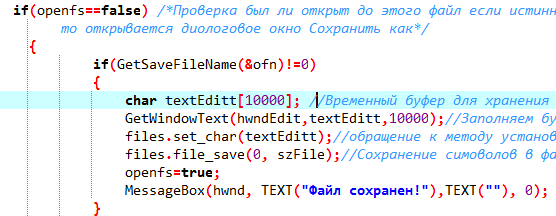
Рисунок 5.Открытие файла

if(GetOpenFileName(&ofn)!=0) - **GetOpenFileName** открывает стандартное общее диалоговое окно открытия файла. Функция не открывает файлы, а только открывает диалоговое окно и возвращает имя файла (файлов), **(&ofn)** указатель на переменную **ofn** (присвоили адрес переменной указателю)

files.file\_load(0, szFile); Считывание из файла последовательно всех символов, через метод загрузки хранимой строки из файла;

SetWindowText(hwndEdit,files.get\_char());Функция **SetWindowText** изменяет текст заголовка заданного окна (если таковой имеется). Если определяемое окно - орган управления, то изменяется его текст., передает его в окно редактора через метод получения хранимой строки в виде c-строки;

openfs=true;//Для реализации кнопки «Сохранить», что файл был открыт и просто сохранить текст в данном файле.



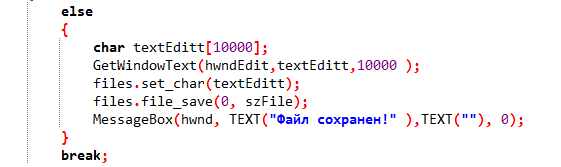


Рисунок 6.Сохранение открытого или нового файла

if(openfs==false) /\*Проверка был ли открыт до этого файл если истинно

то открывается диологовое окно Сохранить как\*/

char textEditt[10000]; //Временный буфер для хранения символов из Edit

GetWindowText(hwndEdit,textEditt,10000); Копирует текст строки заголовка указанного окна (если оно есть) в буфер. Заполняем буфер символами из Edit

files.set\_char(textEditt);//обращение к методу установки хранимой строки из объекта string;

files.file\_save(0, szFile);//Сохранение символов в файл методом метод сохранения хранимой строки в файл;

openfs=true;

MessageBox(hwnd, TEXT("Файл сохранен!"),TEXT(""), 0);

Если же документ был не открыт, а считается как новый, то файл будет сохранен «Сохранить как» и ввести имя файла и путь.

## Заключение

В данной работе мы научились создавать оконное приложение WinApi со свойствами простого текстового редактора.