**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

Образовательная программа «Информационные системы и технологии»

Дисциплина «Теория и технология программирования»

Отчет к лабораторной работе № 10

«Класс для работы со строками CString»

Выполнил:

студент группы 3530902/20001

Сафонов А. И.

Принял:

Доцент к.т.н Хлопин. С. В.

Санкт-Петербург, 2023 год

**Задание лабораторной № 10:**

Написать текстовый редактор с параметрами (индивидуальным заданием), представленными в системной матрице в Таблице 1. В текстовом редакторе должны быть функции открытия файла, проверки расширения файла и его сохранения.

4.Отбор (с N до M символа).

7.Замена всех первых строчных букв слов прописными.

8.Поиск в строке.

**Текст программы:**

Файл main.cpp:

#include <iostream>

#include <string>

#include <fstream>

#include <vector>

#include <Windows.h>

#include <sstream>

#include <algorithm>

#include "ConsoleRead.h"

#include "Position.h"

#include "ConsoleColor.h"

using namespace std;

namespace File {

void Write(string path, const vector<string>& text) {

ofstream fout;

fout.open(path, 'w');

for (const string& line : text) {

fout << line << endl;

}

fout.close();

}

void Write(string path, const string& text) {

ofstream fout;

fout.open(path, 'w');

fout << text << endl;

fout.close();

}

class InputFile {

public:

InputFile(string& path) {

while (true) {

if (path.empty()) {

cout << "Введите путь к файлу: ";

getline(cin, path);

if (!checkPath(path.c\_str())) {

std::cout << "Некоректный путь до файла" << endl;

continue;

}

}

file.open(path);

if (!file.is\_open()) {

std::cout << "Файл не открылся" << endl;

path.clear();

continue;

}

break;

}

}

vector<string> ReadFile() {

vector<string> result;

string str;

while (getline(file, str))

{

result.push\_back(str);

}

return result;

}

~InputFile() {

file.close();

}

private:

ifstream file;

bool checkPath(const char\* path)

{

const char\* reservedWords[] = { "CON", "CONSOLE", "PRN", "AUX", "COM1", "COM2", "COM3", "COM4", "COM5", "COM6", "COM7", "COM8", "COM9",

"LPT1", "LPT2", "LPT3", "LPT4", "LPT5", "LPT6", "LPT7", "LPT8", "LPT9" };

char name[\_MAX\_FNAME];

char ext[\_MAX\_EXT];

\_splitpath\_s(path, nullptr, 0, nullptr, 0, name, \_MAX\_FNAME, ext, \_MAX\_EXT);

std::transform(name, name + strlen(name), name, ::toupper);

for (int i = 0; i < 22; i++)

{

if (strcmp(name, reservedWords[i]) == 0)

{

return false;

}

}

return true;

}

};

}

class Print {

public:

static void PrintText(vector<string> text, string message= "Текущее состояние файла:", Color color = Color::primary) {

ConsoleColor Console;

cout << endl << message << endl;

Console.SetColor(color);

for (size\_t line\_count = 0; line\_count < text.size(); ++line\_count) {

PrintLine(line\_count, text[line\_count]);

}

}

private:

static void PrintLine(size\_t line, string str) {

cout << line + 1 << ". " << str << endl;

}

};

namespace IndividualTasks {

void Capitalize(vector<string>& text) {

for (string& line : text) {

bool IsCap = true;

for (char& ch : line) {

if (ch == ' ' || ch == '(' || ch == '"' || ch == '.' || ch == ',') {

IsCap = true;

}

else if (IsCap) {

ch = toupper(ch);

IsCap = false;

}

}

}

}

pair<Position, Position> GetSelectionPosition(vector<string>& text) {

Position pos\_begin;

Position pos\_end;

do

{

pos\_begin = Read::GetPosition("Введите номер строки и символ, откуда начать копирование: ");

pos\_end = Read::GetPosition("Введите номер строки и символ, конца копирования: ");

if (pos\_end == pos\_begin) {

cout << "Введите хотя бы какой-то интервал пожалуйста :(" << endl;

}

else if (pos\_begin.row < text.size() + 1 && pos\_begin.column <= text[pos\_begin.row - 1].size() + 1

&& pos\_end.row < text.size() + 1 && pos\_end.column <= text[pos\_end.row - 1].size() + 1) {

return { pos\_begin, pos\_end };

}

else {

cout << "Ошибка индексации, попробуйте еще раз" << endl;

}

} while (true);

}

string Copy(vector<string>& text, Position pos\_begin, Position pos\_end) {

string result;

if (pos\_begin > pos\_end) {

std::swap(pos\_begin, pos\_end);

}

for (size\_t column = pos\_begin.column - 1; column < (pos\_end.row == pos\_begin.row ?

pos\_end.column - 1

: text[pos\_begin.row - 1].size()); column++) {

result += text[pos\_begin.row - 1][column];

}

for (size\_t row = pos\_begin.row; row < pos\_end.row - 1; row++) {

result += '\n';

for (size\_t column = 0; column < text[row].size(); column++) {

result += text[row][column];

}

}

if (pos\_begin.row != pos\_end.row) {

result += '\n';

for (size\_t column = 0; column < pos\_end.column - 1; column++) {

result += text[pos\_end.row - 1][column];

}

}

return result;

}

void Selection(vector<string>& text) {

auto [pos\_begin, pos\_end] = GetSelectionPosition(text);

string copy = Copy(text, pos\_begin, pos\_end);

File::Write("Selected.txt", copy);

cout << "Отобранное значение\n";

ConsoleColor color;

color.SetColor(second);

cout<< copy << endl;

}

string ToLower(const string& str) {

string result;

for (char c : str) {

result += tolower(c);

}

return result;

}

void Search(vector<string> text) {

cout << "Введите запрос:"s;

string query = Read::ReadLine();

ConsoleColor color(second);

bool find = false;

for (size\_t i = 0; i < text.size(); i++) {

string to\_search = ToLower(text[i]);

size\_t ind = to\_search.find(ToLower(query));

while (ind < text[i].size()) {

find = true;

cout <<'"' << query << "\" найденно в \"" << text[i] << "\". На " << i + 1

<< " строке, первый символ " << ind + 1 << endl;

ind = to\_search.find(ToLower(query), ind + 1);

}

}

if (!find) {

cout << "По вашему запросу ничего не найдено"s << endl;

}

}

}

namespace Client {

void Begin(string& path) {

File::InputFile fin(path); // Возвращает путь

vector<string> text = fin.ReadFile();

Print::PrintText(text, "Начальный файл:", Color::second);

}

void Current(string& path) {

File::InputFile fin(path); // Возвращает путь

vector<string> text = fin.ReadFile();

Print::PrintText(text);

cout << endl << "Меню:" << endl;

cout << "1.Отбор (с N до M символа)." << endl

<< "2.Замена всех первых строчных букв слов прописными" << endl

<< "3.Поиск в строке." << endl

<< "-Любую другую цифру для выхода" << endl;

int a = Read::GetIntNum("Введите число:");

if (a == 1) {

IndividualTasks::Selection(text);

}

else if (a == 2) {

IndividualTasks::Capitalize(text);

}

else if (a == 3) {

IndividualTasks::Search(text);

}

else {

exit(1);

}

File::Write("output.txt", text);

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

locale loc("");

string path;

Client::Begin(path);

while (true) {

Client::Current(path);

}

return 0;

}

Файл Position.h:

#pragma once

#include <utility>

struct Position {

size\_t row = 0;

size\_t column = 0;

void operator=(std::pair<size\_t, size\_t> date);

};

bool operator>(Position rhs, Position lhs);

bool operator==(Position rhs, Position lhs);

Файл Position.cpp:

#include "Position.h"

void Position::operator=(std::pair<size\_t, size\_t> date)

{

row = date.first;

column = date.second;

}

bool operator>(Position rhs, Position lhs) {

return rhs.row > lhs.row || rhs.row == lhs.row && rhs.column > lhs.column;

}

bool operator==(Position rhs, Position lhs)

{

return rhs.column == lhs.column && rhs.row == lhs.row;

}

Файл ConsoleColor.h:

#pragma once

#include <Windows.h>

enum Color {

primary,

second

};

class ConsoleColor {

public:

ConsoleColor() {

out = GetStdHandle(STD\_OUTPUT\_HANDLE);

GetConsoleScreenBufferInfo(out, &start\_attribute);

}

ConsoleColor(Color color) {

out = GetStdHandle(STD\_OUTPUT\_HANDLE);

GetConsoleScreenBufferInfo(out, &start\_attribute);

SetColor(color);

}

void SetColor(Color color) {

if (color == Color::primary) {

SetConsoleTextAttribute(out, 3);

}

else if (color == Color::second) {

SetConsoleTextAttribute(out, 10);

}

}

~ConsoleColor() {

SetDefaultColor();

}

void SetDefaultColor() {

SetConsoleTextAttribute(out, start\_attribute.wAttributes);

}

private:

HANDLE out;

CONSOLE\_SCREEN\_BUFFER\_INFO start\_attribute;

};

Файл ConoleRead.h:

#pragma once

#include <string>

#include <vector>

#include <iostream>

using namespace std::literals;

class Read {

public:

static std::string ReadLine(std::string message = "Неккоректное значение"s)

{

while (true) {

std::string s;

std::getline(std::cin, s);

if (!s.empty()) {

return s;

}

std::cout << message;

}

}

static double GetNum(std::string message, std::string error\_message = "Неккоректное значение"s)

{

while (true) {

std::cout << message;

std::string str = ReadLine(message);

std::vector<std::string> words = SplitIntoWords(str);

if (words.size() == 1 && CheckIsFloat(words[0]))

{

return std::stod(str);

}

std::cout << error\_message << std::endl;

}

}

static double GetPositivNum(std::string message, std::string error\_message = "Неккоректное значение"s)

{

while (true) {

double epsilon = GetNum(message, error\_message);

if (epsilon > 0) {

return epsilon;

}

std::cout << error\_message << std::endl;

}

}

static int GetIntNum(std::string message, std::string error\_message = "Неккоректное значение"s)

{

while (true) {

std::cout << message;

std::string s = ReadLine(message);

std::vector<std::string> split\_word = SplitIntoWords(s);

if (split\_word.size() == 1 && CheckIsNum(split\_word[0])) {

return std::stoi(split\_word[0]);

}

std::cout << error\_message << std::endl;

}

}

static std::pair<size\_t, size\_t> GetPosition(std::string message = "Введите номер строки и символ:",

std::string error\_message = "Неккоректное значение"s) {

while (true) {

std::cout << message;

std::string s = ReadLine(message);

size\_t column = 0, row = 0;

std::vector<std::string> split\_word = SplitIntoWords(s);

if (split\_word.size() == 2 && CheckIsNum(split\_word[0]) && CheckIsNum(split\_word[1]) &&

std::stoi(split\_word[0]) > 0 && std::stoi(split\_word[1]) > 0 ) {

return { std::stoi(split\_word[0]), std::stoi(split\_word[1]) };

}

std::cout << error\_message << std::endl;

}

}

private:

static std::vector<std::string> SplitIntoWords(const std::string& textSaf)

{

std::vector<std::string> words;

std::string word;

for (const char c : textSaf) {

if (c == ' ') {

if (!word.empty()) {

words.push\_back(word);

word.clear();

}

}

else {

word += c;

}

}

if (!word.empty()) {

words.push\_back(word);

}

return words;

}

static bool CheckIsNum(std::string str)

{

if (str[0] == '+' || str[0] == '-')

str = str.substr(1, str.length());

bool have\_dots = false;

for (char ch : str) {

if (ch == '.' || ch == ',') {

have\_dots = true;

}

else if (have\_dots && ch != '0' || ch > '9' || ch < '0') {

return false;

}

}

return true;

}

static bool CheckIsFloat(std::string str)

{

if (str[0] == '+' || str[0] == '-')

str = str.substr(1, str.length());

bool have\_dots = false;

for (char ch : str) {

if (ch == '.' || ch == ',') {

if (have\_dots) {

return false;

}

have\_dots = true;

}

else if (ch > '9' || ch < '0') {

return false;

}

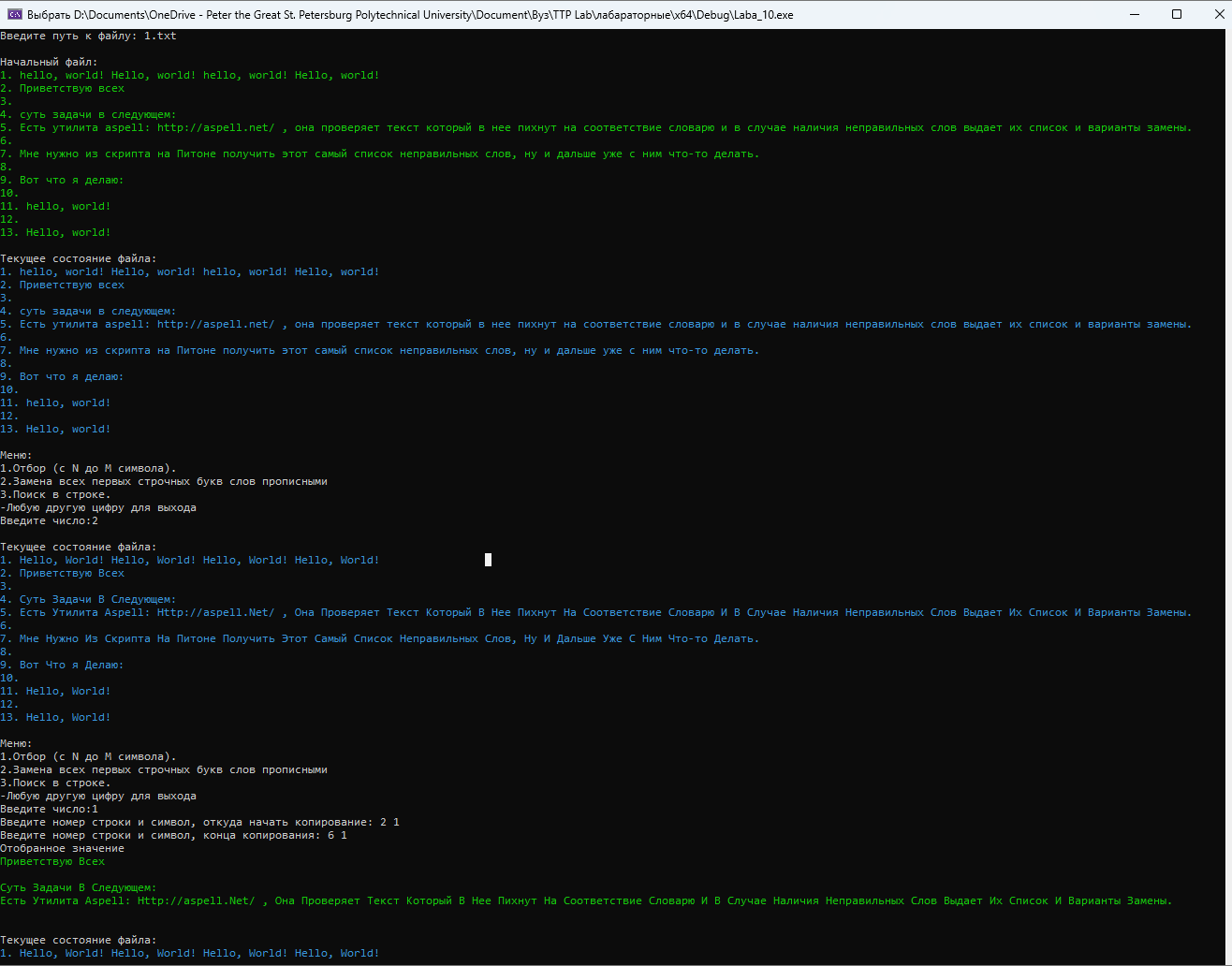
}

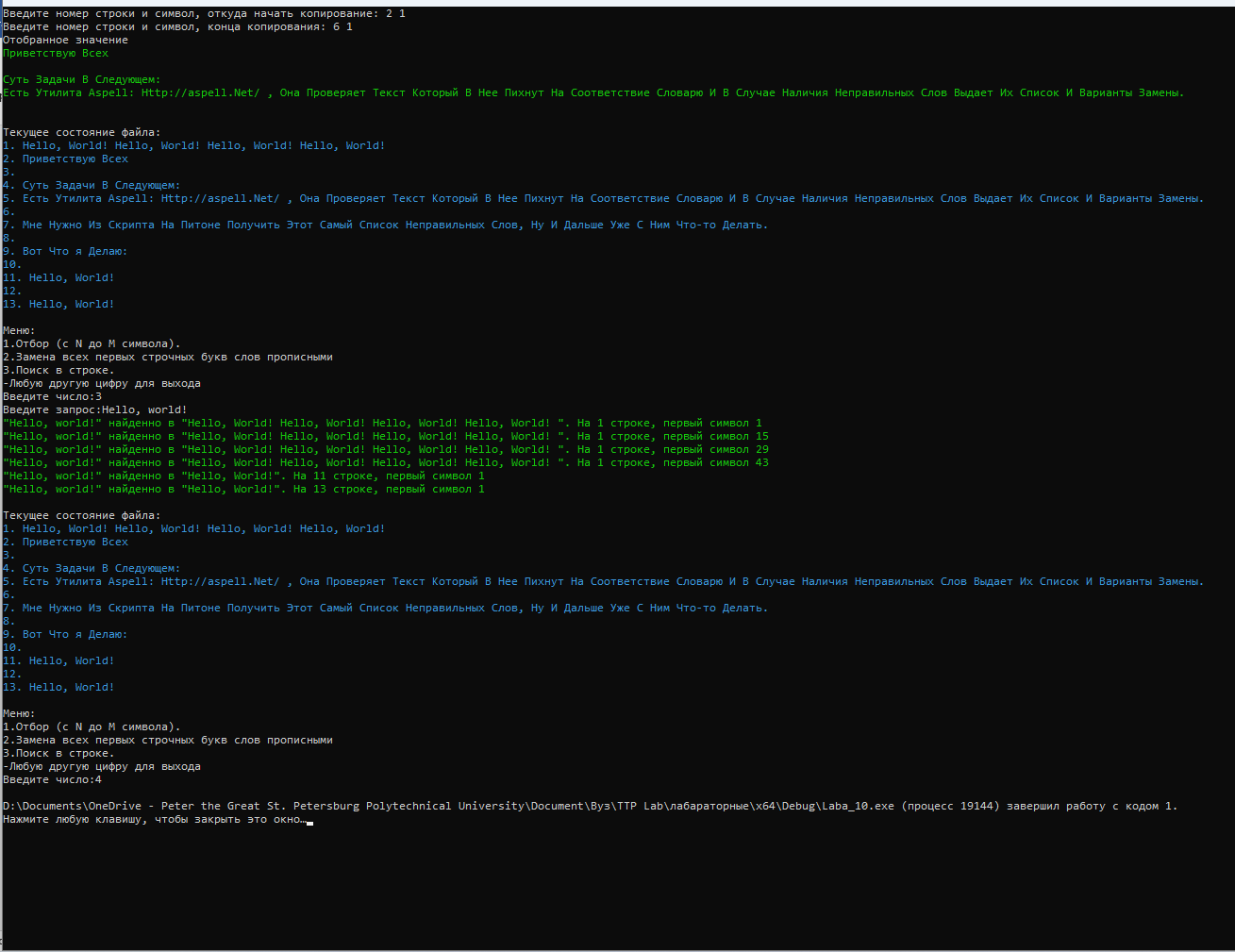
return true;

}

};

**Пример работы:**

****

****

**Выводы:** в ходе выполнения лабораторной работы я познакомился с классом string. Познакомился с различными методами этого класса, и с их помощью реализовал небольшой функционал примитивного текстового редактора. Научился отбирать текст по заданным строчкам и символам относительной этой строки.