

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

Дисциплина: «Лингвистическое и программное обеспечение

автоматизированных систем»

Отчёт по лабораторной работе № 1. Символьные строки

Задан текст. Определить слово с наибольшим вхождением заданного

символа.

**Выполнил студент группы №475:**

Овчинников Роман Сергеевич

**Проверили преподаватели:**

ст. преп., к.т.н. Корниенко И.Г.

ст. преп., Федин А.К.

Санкт-Петербург

2020

*Постановка задачи*

Задан текст. Определить слово с наибольшим вхождением заданного

символа.

*Исходные данные*

В качестве исходных данных программа принимает от пользователя массив и элемент для поиска. Пользователь может заполнить строку с клавиатуры или из файла.

*Особые ситуации*

Необходимо рассмотреть следующие особые ситуации.

* Везде, где программой ожидаются числа, если пользователь ввёл символ вместо числа, будет отображено уведомление с просьбой ввести число.
* Программа выводит сообщение, если выбран несуществующий пункт меню
* При вводе пользователем пути к несуществующему файлу, будет отображено уведомление о том, что не удалось найти файл и предложено повторить ввод.
* При сохранении строки в файл, если такой файл уже имеется – спрашивать у пользователя разрешение на перезапись. Путь к нему не должен содержать недопустимых символов.
* После выполнения программы пользователю будет предложено сохранить результат.
* Пользователю предоставляется выбор способа ввода: заполнить вручную, взять значения из файла.
* Перед выходом из программы будет предложено вернуться в главное меню.

*Алгоритмы решения задачи:*

Для выполнения задачи был выбран следующий алгоритм:

Разделяем введённую строку на слова, меняем все символы на пробелы, проверяем на наличие символ, введённого пользователем, и выводим слово, с наибольшим вхождением данного символа.

*Форматы представления данных*

Таблица 1 – Переменные, используемые в программе.

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| tmp | int, char | Временная переменная |
| fileName | string | Путь к файлу |
| text | string | Текст, который будет печататься другим цветом |
| characterArray | string | Тест вводимый пользователем |
| symbolArray | string | Символ вводимый пользователем |
| first | int | Переменная для смены выводимого сообщения |
| saveArray | string | Отсортированные слова |
| word | string | Слово из строки |
| element | int | Символ из файла (Проверка количества символов в файле) |
| finished | bool | Остановка цикла при верном вводе |
| checkStorage | bool | Проверка сохранения файла и выход из цикла |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| color | int | Номер цвета |
| сhoice | int | Выбор пункта меню пользователем |
| checkExists | bool | Проверка существования файла |
| buf | string | Буфер в котором лежит строка |

*Структура программы*

Программа разделена на 3 модулей:

* Main – модуль, осуществляющий запуск программы.
* Menu – модуль, осуществляющий пользовательский интерфейс.
* Source – модуль в котором выполняются основные алгоритмы программы.

Таблица 2 – Функции модуля Menu

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Greeting | Вывод на экран данных о авторе программы и текста задачи. |
| IntegerCheck | Проверка на целые числа |
| MainMenu | Отображение и выбор пункта меню с последующим переходом к нему |

Таблица 3 – Функции модуля Source

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| YesNo | Отображение вопроса на который можно ответить “Да” и “Нет”, и возвращение ответа |
| ColorChange | Изменение цвета вывода |
| SaveArray | Сохранение введённого теста |
| FileSave | Сохранение и слова в файл |
| AskSave | Скрашивает нужно ли пользователю сохранять |
| CheckSymbol | Проверка введённого символа |
| СhangeString | Замена всех символов на пробелы |
| FindWord | Нахождение слова с наибольшим вхождением заданного символа |
| SaveFile | Отображение вопроса о сохранении файла, считывание имени файла и передача его другим функциям |
| FileEnter | Заполнение из файла |
| FileCheck | Проверка на существование |
| UserChoice | Выбор пользователем способ ввода |
| MainProgram | Отображение меню выбора способа ввода |

Модульные тесты программы в модуле UnitTest1.cpp:

• TestMethodEnglishLetter;

• TestMethodRussianLetter;

• TestMethodEnter;

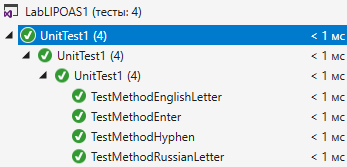
**• TestMethodHyphen.

Рисунок 1 – Пройденные тесты

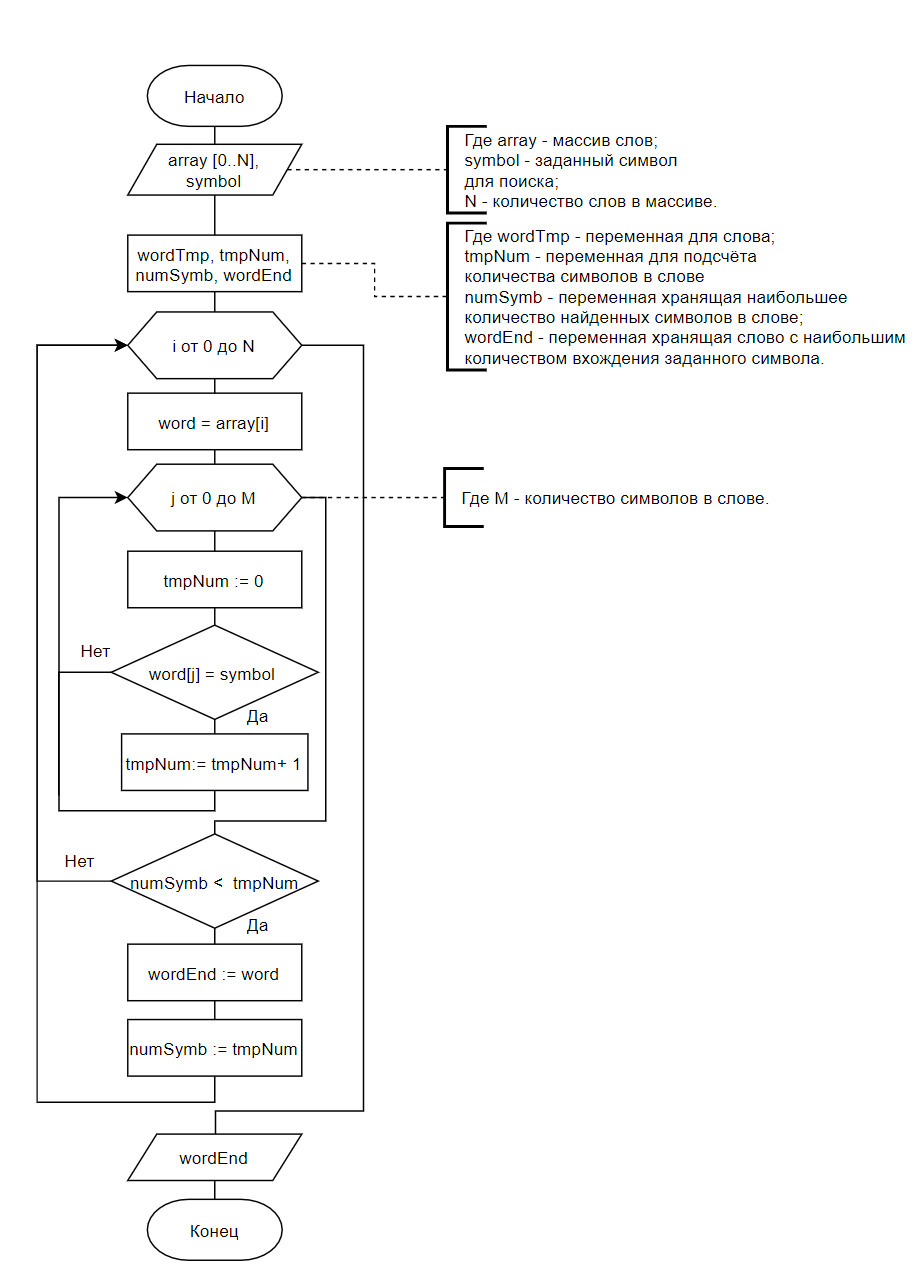
*Блок-схема алгоритма программы*

Рисунок 2 – Блок-схема алгоритма

*Описание хода выполнения лабораторной работы*

1. В ходе лабораторной работы было создано решение в интегрированной среде разработки Microsoft Visual Studio C++ 2017. В нём был создан проект.
2. После набора теста программы выяснилось, что вывод текста на экран консольного приложения работает не правильно из-за различия кодировок консольного приложения и среды разработки.  
   Для решения этой проблемы была использована функция

setlocale(LC\_ALL, "Russian");

которая обеспечивает работу приложению с символами языка системы, в моём случае кириллицы.

1. При сохранении в файл, при вводе имён файлов с символами “/”, “\”, “|”, “\*”, “>” и т.д., а также некоторых слов, например, “aux”, “lpt1”, “com1”, “con”, выводилось сообщение о том, что файл сохранён, но в папке проекта никаких файлов не создавалось.  
   Для решения этой проблемы, я воспользовался функцией

is\_regular\_file(); которая расположена в библиотеке filesystem и содержит в себе все запрещённые имена.

*Результаты работы программы*

В результате работы программа выводит в консоль слово с наибольшим вхождением заданного пользователем символа

*Исходный текст программы*

[Блок Main.cpp ---]

#include <iostream>

#include "Menu.h"

#include "Source.h"

#include <Windows.h>

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

Greeting();

do {

} while (MainProgram(MainMenu(true)));

return 0;

}

[--- Конец блока Main.cpp]

[Блок Menu.cpp ---]

#include "Source.h"

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

enum InternalMenu {

Start = 1,

Exit = 2,

};

// Приветствие

void Greeting() {

cout << "Студент группы № 475: Овчинников Роман" << endl << endl;

cout << "Задан текст. Определить слово с наибольшим вхождением заданного символа." << endl << endl;

}

// Проверка на целочисленность. Принимает минимальное и максимальное значение.

int IntegerCheck(int min, int max) {

int tmp;

cin >> tmp;

while (cin.fail()) {

cout << "Неверный ввод. Повторите ввод:";

cin.clear();

cin.ignore(INT\_MAX, '\n');

cin >> tmp;

}

if ((tmp >= min) && (tmp <= max)) {

cin.ignore(INT\_MAX, '\n');

return tmp;

}

else {

cin.ignore(INT\_MAX, '\n');

cout << "Неверный ввод. Повторите ввод:";

IntegerCheck(min, max);

}

}

// Главное меню

bool MainMenu(bool сhoice) {

if (сhoice) {

cout << "1. Начать работу." << endl;

}

else {

cout << endl << "Попробовать еще раз?" << endl;

cout << "1. Начать заново" << endl;

}

cout << "2. Выход." << endl << endl;

cout << "Ваш выбор: ";

int const NumberItemsMin = 1;

int const NumberItemsMax = 2;

switch (IntegerCheck(NumberItemsMin, NumberItemsMax)) {

case(Start):

cout << endl;

MainProgram(true);

break;

case(Exit):

return false;

break;

}

}

[--- Конец блока Menu.cpp]

[Блок Source.cpp ---]

#include <Windows.h>

#include "Menu.h"

#include <iostream>

#include <string>

#include <fstream>

#include <filesystem>

#include <sstream>

using namespace std;

HANDLE hOUTPUT = GetStdHandle(STD\_OUTPUT\_HANDLE);

// Цвета

enum ConsoleColor {

LightGray = 7,

LightGreen = 10,

LightRed = 12,

Yellow = 14,

};

// Пункты меню.

enum InternalMenu {

Yes = 1,

No,

Again,

};

enum Menu {

ManualInput = 1,

FileInput,

};

enum Numbers {

NumberItemsMin = 1,

NumberItemsMax,

};

int YesNo() {

cout << "1. Да." << endl;

cout << "2. Нет." << endl;

cout << "Ваш выбор: ";

return IntegerCheck(Yes, No);

}

// Изменение цвета

void ColorChange(string text, int color) {

if (color == LightRed) {

SetConsoleTextAttribute(hOUTPUT, LightRed);

cout << text << endl;

SetConsoleTextAttribute(hOUTPUT, LightGray);

}

if (color == LightGreen) {

SetConsoleTextAttribute(hOUTPUT, LightGreen);

cout << text << endl;

SetConsoleTextAttribute(hOUTPUT, LightGray);

}

if (color == Yellow) {

SetConsoleTextAttribute(hOUTPUT, Yellow);

cout << text << endl;

SetConsoleTextAttribute(hOUTPUT, LightGray);

}

}

// Спросить нужно ли сохранять

int AskSave() {

cout << "Хотите сохранить текст в файл?" << endl;

return YesNo();

}

//Проверка ввода символа

string CheckSymbol() {

string symbolArray;

bool finished = false;

cout << "Введите символ для поиска: ";

do {

cin >> symbolArray;

if (symbolArray.length() > 1) {

cout << "Cимвол должен быть только 1!";

cout << endl;

cin.clear();

while (cin.get() != '\n'); // чтобы после пробела не считывало ещё раз

continue;

}

unsigned char tmp = symbolArray[0]; // сделал это, потому что русские буквы уходили в отрицательность

// по таблице ASCII

// 65 - "A"; 90 - "Z"; 97 - "a"; 122 - "z"; 192 - "A"(рус); 255 - "я";

if (static\_cast<int>(tmp) < 65 ||

(static\_cast<int>(tmp) > 90 && static\_cast<int>(tmp) < 97) ||

((static\_cast<int>(tmp) > 122 && static\_cast<int>(tmp) < 192) ||

static\_cast<int>(tmp) > 255)) {

cout << "Символ должен быть буквой! ";

cout << endl;

continue;

}

finished = true;

} while (!finished);

return symbolArray;

}

// Сохранение в файл

void SaveArray(string buf) {

bool checkStorage = false;

string fileName;

int first = 1;

do {

if (first == 1) {

cout << endl << "Если вы введёте только имя файла, то он будет в папке с проектом." << endl;

cout << "Для сохранения файла в другом месте укажите полный путь к файлу." << endl;

cout << "(По умолчанию в конец добавляется .txt)" << endl;

cout << endl << "Введите путь к файлу: ";

}

else {

cout << endl << "Попробуйте ввести другой: ";

}

first++;

cin >> fileName;

fileName += ".txt";

bool checkExists = false;

if (ifstream(fileName)) // проверяем на сущ.

checkExists = true;

ofstream file;

file.open(fileName, ios::app);

if (file.good()) {

if (!std::experimental::filesystem::is\_regular\_file(fileName)) {

ColorChange("Недопустимое имя файла.", LightRed);

file.close();

continue;

}

if (checkExists) {

cout << "Данный файл существует! Перезаписать?" << endl;

if (YesNo() == No) {

file.close();

continue;

}

else {

file.close();

file.open(fileName, ios::trunc);

file << buf;

checkStorage = true;

ColorChange("Файл успешно записан!", LightGreen);

cout << endl;

}

}

else {

file.close();

file.open(fileName, ios::out);

file << buf;

checkStorage = true;

ColorChange("Файл успешно записан!", LightGreen);

cout << endl;

}

}

else {

ColorChange("Ошибка при открытии файла.", LightRed);

}

file.close();

} while (!checkStorage);

}

// Проверка файла

bool FileCheck(string fileName) {

ifstream file;

file.open(fileName);

long fileSize;

file.seekg(0, ios::end);

fileSize = file.tellg();

if (fileSize == 0) {

ColorChange("Файл пуст!", LightRed);

return false;

}

file.close();

return true;

}

// Заполнение из файла

string FileEnter() {

string fileName;

cout << "Введите полный путь к файлу: ";

getline(cin, fileName);

ifstream file;

file.open(fileName);

if (std::experimental::filesystem::is\_regular\_file(fileName) && file.good()) {

if (FileCheck(fileName) == true) {

ColorChange("Файл открыт!", LightGreen);

cout << endl;

if (!file.eof()) {

string buf;

if (!file.eof()) {

file >> buf.assign((std::istreambuf\_iterator<char>(file.rdbuf())), std::istreambuf\_iterator<char>()); // записать весь буфер в файл

}

cout << "Введённая строка:" << endl;

ColorChange(buf, LightGreen);

cout << endl;

return buf;

}

}

}

else {

ColorChange("Ошибка открытия файла!", LightRed);

cout << endl;

return "0";

}

file.close();

return "0";

}

// Убираю символы и заменяю на пробел

string СhangeString(string buf) {

string newBuf = "";

unsigned char tmp;

for (int i = 0; i < buf.length(); i++)

{

tmp = buf[i];

// по таблице ASCII

// 65 - "A"; 90 - "Z"; 97 - "a"; 122 - "z"; 192 - "A"(рус); 255 - "я"; 45 - "-"

if (static\_cast<int>(tmp) < 65 ||

(static\_cast<int>(tmp) > 90 && static\_cast<int>(tmp) < 97) ||

((static\_cast<int>(tmp) > 122 && static\_cast<int>(tmp) < 192) ||

static\_cast<int>(tmp) > 255)) {

if (static\_cast<int>(tmp) == 45) { // если это дефис

newBuf += tmp;

continue;

}

newBuf += " ";

}

else {

newBuf += tmp;

}

}

return newBuf;

}

string FindWord(string buf, string symbolArray) {

string word;

istringstream ist(buf);

string wordExit = "";

int numWords = 0;

while (ist >> word) {

for (auto symbol : symbolArray)

{

if (word.find(symbol) != string::npos)

{

int numWord = 0;

for (int i = 0; i < word.length(); i++)

{

if (word[i] == symbol) {

numWord++;

}

}

if (numWords < numWord) {

numWords = numWord;

wordExit = word;

}

break;

}

}

}

return wordExit;

}

// Пользовательский выбор

void UserСhoice(int choice) {

string buf, symbolArray, word;

if (choice == ManualInput) {

cout << "Введите текст (для завершения ввода поставьте символ ';' и нажмите Enter): ";

SetConsoleTextAttribute(hOUTPUT, LightGreen);

getline(cin, buf, ';');

SetConsoleTextAttribute(hOUTPUT, LightGray);

}

if (choice == FileInput) {

buf = FileEnter();

}

if ((buf != "0") && (buf.length() != 0)) {

if (choice == ManualInput) {

if (AskSave() == Yes) {

SaveArray(buf);

}

}

buf = СhangeString(buf); // очищаю от хлама "!@#$%^&\*"

cout << endl;

symbolArray = CheckSymbol(); // Ввод и проверка символа

cout << "Найдено слово: " << endl;

word = FindWord(buf, symbolArray);

ColorChange(word, Yellow);

string saveArray = word;

if (saveArray.size() != 0) {

if (AskSave() == Yes) {

SaveArray(saveArray);

}

}

else {

ColorChange("Ничего не найдено!!!", LightRed);

}

}

else {

if (choice == ManualInput) {

ColorChange("Строка пустая!", LightRed);

}

}

}

// Меню выбора

bool MainProgram(bool check) {

if (check == true) {

cout << "1. Ручной ввод." << endl;

cout << "2. Ввод из файла." << endl;

cout << endl << "Ваш выбор: ";

switch (IntegerCheck(NumberItemsMin, NumberItemsMax)) {

case(ManualInput):

UserСhoice(ManualInput);

break;

case(FileInput):

UserСhoice(FileInput);

break;

}

MainMenu(false);

}

return false;

}

[--- Конец блока Sourse.cpp]

[--- Начало блока Unittest1.cpp]

#include "stdafx.h"

#include "CppUnitTest.h"

#include "../LabLIPOAS1/Source.h"

#include "../LabLIPOAS1/Source.cpp"

#include "../LabLIPOAS1/Menu.cpp"

#include "../LabLIPOAS1/Menu.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1

{

TEST\_CLASS(UnitTest1)

{

public:

TEST\_METHOD(TestMethodEnglishLetter)

{

std::string text = "abb aba bba bbb aaa bbab bbbab";

std::string symbol = "a";

std::string word = "aaa";

Assert::AreEqual(FindWord(text, symbol), word);

}

TEST\_METHOD(TestMethodRussianLetter)

{

std::string text = "гтл ллг гггглл гыыл ггылыол";

std::string symbol = "г";

std::string word = "гггглл";

Assert::AreEqual(FindWord(text, symbol), word);

}

TEST\_METHOD(TestMethodEnter)

{

std::string text = "jhdf\n hhhjjj\n jjkkyyyyyyy\n ooo";

std::string symbol = "j";

std::string word = "hhhjjj";

Assert::AreEqual(FindWord(text, symbol), word);

}

TEST\_METHOD(TestMethodHyphen)

{

std::string text = "Здесь когда-то было море зелёное";

std::string symbol = "о";

std::string word = "когда-то";

Assert::AreEqual(FindWord(text, symbol), word);

}

};

} [--- Конец блока Unittest1.cpp]