Изображение выглядит как цепь, медальон

Автоматически созданное описание

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Системы автоматизированного проектирования | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Программирование | | |
| Курс | I | | | Группа | 414 |

Отчёт по контрольной работе № 4

Вариант № 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 414 |  |  |  | Кондиляброва  Вероника Данииловна |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Федин Алексей Константинович |

Содержание

[1 Постановка задачи 3](#_Toc114818740)

[2 Исходные данные 3](#_Toc114818741)

[3 Особые ситуации 3](#_Toc114818742)

[4 Математические методы и алгоритмы решения задач 3](#_Toc114818743)

[5 Форматы представления данных 4](#_Toc114818744)

[6 Структура программы 5](#_Toc114818745)

[7 Описание хода выполнения 5](#_Toc114818746)

[8 Блок-схема алгоритма решения задачи 7](#_Toc114818747)

[9 Результаты работы программы 12](#_Toc114818748)

[10 Исходный код полученного программного решения 17](#_Toc114818749)

## 1 Постановка задачи

Задание предназначено для приобретения практического опыта работы с классом std::string в языке программирования С++. Цель контрольной работы состоит в формировании знаний и умений: – по использованию различных способов описания и формирования символьных строк, – по использованию методов чтения и записи строк в текстовых файлах, – по использованию методов чтения и анализа потоковых данных, вводимых с клавиатуры. Во всех программах необходимо предусмотреть возможность многострочного ввода с клавиатуры.

Подсчитать в заданном тексте количество символов, слов, строк, абзацев. Подсчитать количество слов в предложениях и вывести статистическую таблицу, в которой длине предложения в словах будет соответствовать количество таких предложений в анализируемом тексте.

## 2 Исходные данные

В качестве исходных данных программа использует текст. Ввод с клавиатуры, из файла.

## 3 Особые ситуации

1 Отсутствие ожидаемых программой файлов устранено методом open. Если программа не находит файл с введенным пользователем именем, этот метод сам его создает;

2 Чтобы завершить ввод, введено условие: пользователь должен нажать символ \*

## 4 Математические методы и алгоритмы решения задач

Поиск количества символов в тексте:

1. Посчитать длины строк – это и будет количество строк;

Поиск количества слов в тексте:

* 1. Ищем указатель на пробел;
  2. Пока пробелы есть, увеличиваем количество слов;
  3. Удаляем строку до пробела включительно с помощью указателя;
  4. Ищем следующий указатель на пробел;
  5. Ищем указатель на отступ;
  6. Пока отступы есть, увеличиваем количество слов;
  7. Удаляем строку до отступа включительно с помощью указателя;
  8. Ищем следующий отступ;

Поиск количества строк в тексте:

1. Считывание текста идет по строкам, поэтому можно просто подсчитать количество элементов в векторе строк;

Поиск количества абзацев в тексте:

1. Ищем указатель на отступ;
2. Пока отступы есть, увеличиваем количество абзацев;
3. Удаляем строку до отступа включительно с помощью указателя;
4. Ищем следующий отступ;

Создание таблицы соотношения количеству слов количества предложений с таким количеством слов:

1. Ищем указатель на точку, вопросительный знак и восклицательный знак;
2. Пока какой-нибудь из данных знаков препинания найден, подсчитываем количество слов до него;
3. Найденное количество слов добавляем в вектор;
4. Находим максимальное количество слов;
5. Двойным циклом for (первый до максимального числа слов, второй до размера вектора) находим нужные соотношения и выводим на экран;

## 5 Форматы представления данных

Таблица 1 – Переменные

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| Count\_symbols | int | Подсчет количества символов |
| Count\_words | int | Подсчет количества слов |
| Count\_line | int | Подсчет количества строк |
| Count\_paragraph | int | Подсчет количества абзацев |
| text | vector<string> | Текст |
| table | vector<int> | Таблица соотношения |
| exit | bool | Выход из программы |

## 6 Структура программы

Таблица 2 – Структура программы

|  |  |  |
| --- | --- | --- |
| Модуль | Функция | Описание |
| Correct\_input | GetInt | Проверка ввода количества строк и столбцов |
| Function | InputString | Ввода текста |
|  | OfFileText | Считывание текста из файла |
|  | CountSymbols | Подсчет символов |
|  | CountWords | Подсчет слов |
|  | CountLine | Подсчет строк |
|  | CountParagraph | Подсчет абзацев |
|  | Table | Создание таблицы соотношения |
| Menu | Menu | Осуществляет работу пользовательского интерфейса |
| Main | main | Вызывает функцию Menu, с которой начинается выполнение программы |
| Test | test | Тестирует работу программы |

## 7 Описание хода выполнения

При выполнении контрольной работы я:

* Изучила работу с вектором типа string;
* Научилась считывать строки;
* Научилась осуществлять многострочный ввод с помощью метода getline;
* Изучила работа с методами класса string, такими как find, erase;
* Освоила работу с указателями на символы строки;

Решенные возникшие проблемы:

* Как осуществить многострочный ввод – сделала ограничение на ввод символа \*
* К посчитанному количеству слов приходилось прибавлять один, так как подсчет осуществлялся с помощью поиска пробелов – учла отступ, чтобы первое слово тоже считалось;

Непонятные моменты:

* Как осуществить многострочный ввод, не используя ограничения на символ;
* Как сделать поиск количества предложений способом, походим на Текст.find(‘!’, ‘?’, ‘.’), а не перебирая большое количество вариантов нахождения этих знаков препинания относительно друг друга. Или как сделать поиск количества предложений не методом find;
* Как отчистить вектор от повторяющихся чисел, которые не идут подряд;

## 8 Блок-схема алгоритма решения задачи

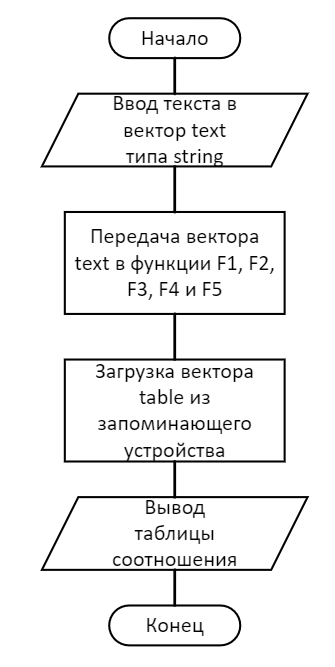


Рисунок 1 – Блок-схема алгоритма решения

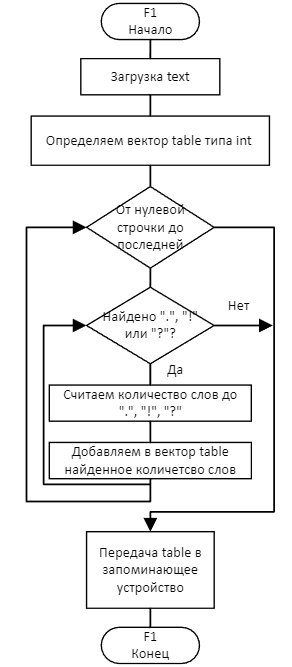


Рисунок 2 – Блок-схема функции создания таблицы соотношения

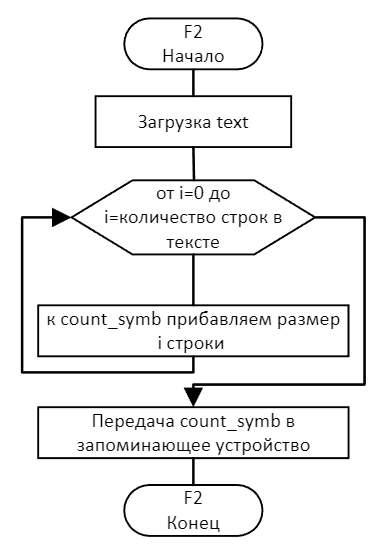


Рисунок 3 – Блок-схема функции подсчета символов

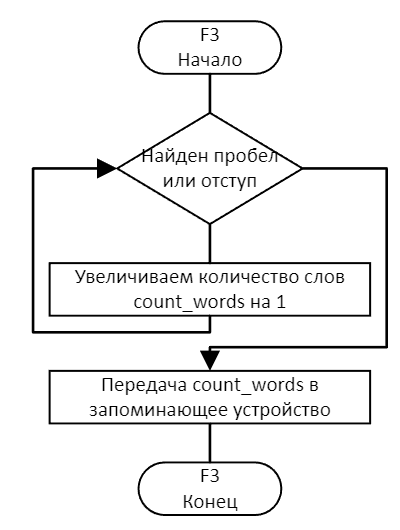


Рисунок 4 – Блок-схема функции подсчета слов

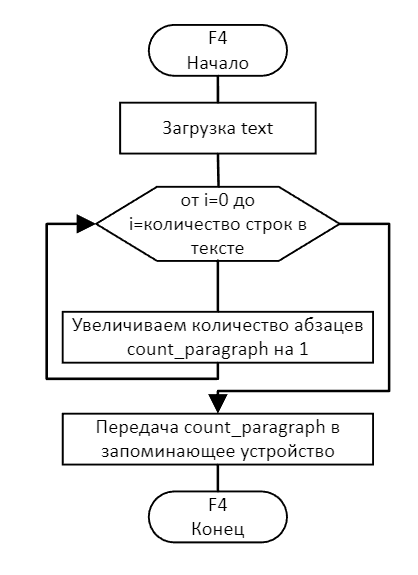


Рисунок 5 – Блок-схема функции подсчета абзацев

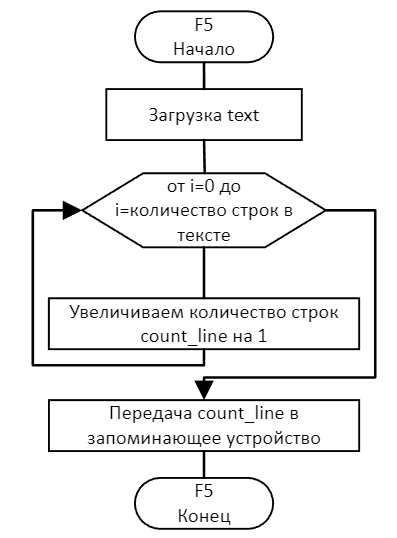


Рисунок 6 – Блок-схема функции подсчета строк

## 9 Результаты работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Скриншот непройденного теста

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Скриншот пройденного теста

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Скриншот проверки на ввод

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Скриншот ввода текста с клавиатуры и подсчет символов, слов, строк, абзацев

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 – Скриншот чтения текста с файла и подсчет символов, слов, строк, абзацев Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 – Скриншот таблицы

## 10 Исходный код полученного программного решения

//Main.cpp

#include<iostream>

#include"Menu.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "ru");

cout << "Kondilyabrova Veronika Daniilovna\nGroup 414" << endl;

cout << "Control work 4\nVariant 10\n\n" << endl;

cout<<"Count the number of characters, words, lines, paragraphs in a given text.\nCount the number of words in sentences and output a statistical\ntable in which the length of the sentence in words will correspond\nto the number of such sentences in the analyzed text.\n\n";

Menu();

return 0;

}

//Menu.h

#pragma once

#include<iostream>

#include<vector>

#include <algorithm>

#include"Test.h"

#include"Function.h"

using namespace std;

enum {

Test=1,

Manual\_input\_text,

Text\_from\_file,

Look\_text,

Look\_table,

Exit,

};

void Menu();

//Menu.cpp

#include "Menu.h"

void Menu() {

int Count\_symbols{};

int Count\_words{};

int Count\_line{};

int Count\_paragraph{};

vector<string> text;

bool exit = true;

while (exit == true) {

cout << "Look test - 1\nEnter the text from keyboard - 2\nAdd the text from file - 3\nView text - 4\nView table - 5\nExit - 6\nChoose: ";

int user\_choice{};

user\_choice = GetInt();

switch (user\_choice) {

case(Test): {

test();

}break;

case(Manual\_input\_text): {

cout << "Text must end '\*'" << endl;

text=InputString(text, &Count\_symbols, &Count\_words, &Count\_line, &Count\_paragraph);

cout << "Symbols: " << Count\_symbols << "\n" << "Words: " << Count\_words<< "\n" << "Line: " << Count\_line << "\n" << "Paragraph: " << Count\_paragraph << endl;

}break;

case(Text\_from\_file): {

cout << "Text must end '\*'" << endl;

text=OfFileText(text, &Count\_symbols, &Count\_words, &Count\_line, &Count\_paragraph);

cout << "Symbols: " << Count\_symbols << "\n" << "Words: " << Count\_words<< "\n" << "Line: " << Count\_line << "\n" << "Paragraph: " << Count\_paragraph << endl;

}break;

case(Look\_text): {

for (int i = 0; i< static\_cast<int>(text.size()); i++) {

cout << text[i] << endl;

}

}break;

case(Look\_table): {

vector<int> table;

table=Table(text);

int max\_count\_words = \*max\_element(table.begin(), table.end());

cout << "Count words in sentences\t\t Count sentences" << endl;

for (int j = 1; j <= max\_count\_words; j++) {

for (int i = 0; i< static\_cast<int>(table.size()); i++) {

if (table[i] == j) {

cout << j << "\t\t\t\t\t\t" << count(begin(table), end(table), j) << endl << endl;

break;

}

}

}

}break;

case(Exit): {

exit = false;

}break;

default: {

cout << "Error, try again\n";

}break;

}

}

}

//Function.h

#pragma once

#include"Correct\_input.h"

#include<string>

#include<vector>

#include<fstream>

vector<string> InputString(vector<string> Text, int\* count\_symb, int\* count\_words, int\* count\_line, int\* count\_paragraph);

vector<string> OfFileText(vector<string> Text, int\* count\_symb, int\* count\_words, int\* count\_line, int\* count\_paragraph);

int CountSymbols(string line, int\* count\_symb);

int CountWords(string line, int\* count\_words);

int CountLine(vector<string> Text, int\* count\_line);

int CountParagraph(string line, int\* count\_paragraph);

vector<int> Table(vector<string> Text);

//Function.cpp

#include"Function.h"

#include<algorithm>

vector<string> InputString(vector<string> Text, int\* count\_symb, int\* count\_words, int\* count\_line, int\* count\_paragraph) {

string str;

while (getline(cin, str, '\n') && str!="\*")

{

Text.push\_back(str);

CountSymbols(str, count\_symb);

CountWords(str, count\_words);

CountParagraph(str, count\_paragraph);

}

CountLine(Text, count\_line);

return Text;

}

vector<string> OfFileText(vector<string> Text, int\* count\_symb, int\* count\_words, int\* count\_line, int\* count\_paragraph) {

cout << "Enter the name of file(without file permission!): ";

string name\_file = "";

cin >> name\_file;

ifstream file;

file.open(name\_file);

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

cout << "File opening error" << endl;

}

else {

while (!file.eof()) {

string str{};

while (getline(file, str, '\n')&&str != "\*")

{

Text.push\_back(str);

CountSymbols(str, count\_symb);

CountWords(str, count\_words);

CountParagraph(str, count\_paragraph);

//Table(str);

}

CountLine(Text, count\_line);

}

}

file.close();

return Text;

}

int CountSymbols(string line, int\* count\_symb) {

\*count\_symb += static\_cast<int>(line.size());

return \*count\_symb;

}

int CountWords(string line, int\* count\_words) {

int pos = line.find(' ');

int posit = line.find('\t');

while (posit + 1) {

++\*count\_words;

line.erase(0, posit + 1);

posit = line.find('\t');

}

while (pos+1) {

++\*count\_words;

line.erase(0, pos + 1);

pos = line.find(' ');

}

return \*count\_words;

}

int CountLine(vector<string> Text, int\* count\_line) {

\*count\_line = static\_cast<int>(Text.size());

return \*count\_line;

}

int CountParagraph(string line, int\* count\_paragraph) {

int pos = line.find('\t');

while (pos + 1) {

++\*count\_paragraph;

line.erase(0, pos + 1);

pos = line.find('\t');

}

return \*count\_paragraph;

}

vector<int> Table(vector<string> Text) {

int Words{};

vector<int> table;

string line;

for (int j = 0; j< static\_cast<int>(Text.size()); j++) {

int pos\_point = Text[j].find('.');

int pos\_exclamation\_mark = Text[j].find('!');

int pos\_question\_mark = Text[j].find('?');

while ((pos\_point + 1 || pos\_exclamation\_mark + 1 || pos\_question\_mark + 1)) {

pos\_point = Text[j].find('.');

pos\_exclamation\_mark = Text[j].find('!');

pos\_question\_mark = Text[j].find('?');

if (pos\_exclamation\_mark + 1) {

if (((pos\_exclamation\_mark + 1) && (pos\_point == -1) && (pos\_question\_mark == -1)) || ((pos\_question\_mark + 1) && (pos\_point == -1) && (pos\_exclamation\_mark + 1 < pos\_question\_mark + 1)) || ((pos\_point + 1) && (pos\_question\_mark == -1) && (pos\_exclamation\_mark + 1 < pos\_point + 1)) || ((pos\_question\_mark + 1) && (pos\_point + 1) && (pos\_exclamation\_mark + 1 < pos\_question\_mark + 1) && (pos\_exclamation\_mark + 1 < pos\_point + 1))) {

line.append(Text[j], 0, pos\_exclamation\_mark + 1);

Words = CountWords(line, &Words);

table.push\_back(Words);

Words = 0;

Text[j].erase(0, pos\_exclamation\_mark + 1);

pos\_exclamation\_mark = Text[j].find('!');

line.erase(0);

}

}

pos\_point = Text[j].find('.');

pos\_exclamation\_mark = Text[j].find('!');

pos\_question\_mark = Text[j].find('?');

if (pos\_question\_mark + 1) {

if (((pos\_point + 1) && (pos\_exclamation\_mark + 1) && (pos\_question\_mark + 1 < pos\_point + 1) && (pos\_question\_mark + 1 < pos\_exclamation\_mark + 1)) || ((pos\_question\_mark + 1) && (pos\_point == -1) && (pos\_exclamation\_mark == -1)) || ((pos\_exclamation\_mark + 1) && (pos\_question\_mark + 1 < pos\_exclamation\_mark + 1) && (pos\_point == -1)) || ((pos\_point + 1) && (pos\_question\_mark + 1 < pos\_point + 1) && (pos\_exclamation\_mark == -1))) {

line.append(Text[j], 0, pos\_question\_mark + 1);

Words = CountWords(line, &Words);

table.push\_back(Words);

Words = 0;

Text[j].erase(0, pos\_question\_mark + 1);

pos\_question\_mark = Text[j].find('?');

line.erase(0);

}

}

pos\_point = Text[j].find('.');

pos\_exclamation\_mark = Text[j].find('!');

pos\_question\_mark = Text[j].find('?');

if (pos\_point + 1) {

if (((pos\_exclamation\_mark + 1) && (pos\_question\_mark + 1) && (pos\_point + 1 < pos\_question\_mark + 1) && (pos\_point + 1 < pos\_exclamation\_mark + 1)) || ((pos\_point + 1) && (pos\_question\_mark == -1) && (pos\_exclamation\_mark == -1)) || (pos\_exclamation\_mark + 1 && pos\_question\_mark == -1 && pos\_point + 1 < pos\_exclamation\_mark + 1) || (pos\_question\_mark + 1 && pos\_exclamation\_mark == -1 && pos\_point + 1 < pos\_question\_mark + 1)) {

line.append(Text[j], 0, pos\_point + 1);

Words = CountWords(line, &Words);

table.push\_back(Words);

Words = 0;

Text[j].erase(0, pos\_point + 1);

pos\_point = Text[j].find('.');

line.erase(0);

}

}

pos\_point = Text[j].find('.');

pos\_exclamation\_mark = Text[j].find('!');

pos\_question\_mark = Text[j].find('?');

}

}

return table;

}

//Correct\_input.h

#pragma once

#include<iostream>

using namespace std;

int GetInt();

//Correct\_input.cpp

#include"Correct\_input.h"

int GetInt() {

int input{};

cin >> input;

while (!(input >= 1)) {

cout << "Error, try again" << endl;

cin.clear();

cin.sync();

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

cin >> input;

}

return input;

}

//Test.h

#pragma once

#include "Menu.h"

void test();

//Test.cpp

#include "Test.h"

void test() {

vector<string> TextTest;

vector<int> TestTable;

vector<int> ResultTable = { 3, 5, 3, 3, 3, 4, 3, 2, 3, 3};

int count\_symb\_result = 169;

int count\_words\_result = 32;

int count\_paragraph\_result = 3;

int count\_line\_result = 3;

int count\_symb{};

int count\_words{};

int count\_paragraph{};

int count\_line{};

int count\_symb\_test{};

int count\_words\_test{};

int count\_paragraph\_test{};

int count\_line\_test{};

string name\_file = "";

name\_file= "text";

ifstream file;

file.open(name\_file);

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

cout << "File opening error" << endl;

}

else {

while (!file.eof()) {

string str{};

while (getline(file, str, '\n') && str != "\*")

{

TextTest.push\_back(str);

count\_symb\_test = CountSymbols(str, &count\_symb);

count\_words\_test = CountWords(str, &count\_words);

count\_paragraph\_test = CountParagraph(str, &count\_paragraph);

}

count\_line\_test = CountLine(TextTest, &count\_line);

}

}

file.close();

TestTable = Table(TextTest);

if (TestTable == ResultTable && (count\_symb\_test == count\_symb\_result) && (count\_words\_test== count\_words\_result) && (count\_paragraph\_test== count\_paragraph\_result) && (count\_line== count\_line\_result) ) {

cout << endl << "Test passed" << endl << endl;

}

else {

cout << "Test is not passed" << endl;

cout << "The result of the work:" << endl;

int max\_count\_words = \*max\_element(TestTable.begin(), TestTable.end());

cout << "Count words in sentences\t\t Count sentences" << endl;

for (int j = 1; j <= max\_count\_words; j++) {

for (int i = 0; i< static\_cast<int>(TestTable.size()); i++) {

if (TestTable[i] == j) {

cout << j << "\t\t\t\t\t\t" << count(begin(TestTable), end(TestTable), j) << endl << endl;

break;

}

}

}

cout << endl << "Expected result:" << endl;

max\_count\_words = \*max\_element(ResultTable.begin(), ResultTable.end());

cout << "Count words in sentences\t\t Count sentences" << endl;

for (int j = 1; j <= max\_count\_words; j++) {

for (int i = 0; i< static\_cast<int>(ResultTable.size()); i++) {

if (ResultTable[i] == j) {

cout << j << "\t\t\t\t\t\t" << count(begin(ResultTable), end(ResultTable), j) << endl << endl;

break;

}

}

}

}

}

// uedsf sdiiuf dfiu.spdfjo ajkfb sdkf sakfj vksd.podsf kasj fskdjv.

// IOSdf kdjf k!KJds fkjs d ? SLdf sdlkf lsf blf!

// jKd sdjk fsjd.Sdjf skd.KSDjf ksdj f.KSDjf kblkd v.

//\*