

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 | | 09.03.03 | Прикладная информатика | | |
| Направленность (профиль) | |  | Прикладная информатика в химии | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Программирование | | |
| Курс | I | | | Группа | 415 |

Отчёт по контрольной работе № 4

Вариант №4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 415 |  |  |  | Никита Леванович Шарашидзе |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Федин Алексей Константинович |
|  |  | (дата, подпись) |  | Корниенко Иван Григорьевич |

Содержание

[1 Постановка задачи 3](#_Toc117424662)

[2 Исходные данные 3](#_Toc117424663)

[3 Особые ситуации 3](#_Toc117424664)

[4 Математические методы и алгоритмы решения задач 3](#_Toc117424665)

[5 Форматы представления данных 3](#_Toc117424666)

[6 Структура программы 5](#_Toc117424667)

[7 Блок-схемы алгоритмов 7](#_Toc117424668)

[8 Результаты работы программы 8](#_Toc117424669)

[9 Описание хода выполнения контрольной работы 10](#_Toc117424670)

[10 Исходный текст программы 10](#_Toc117424671)

## 1 Постановка задачи

Выделить в заданном тексте все диалоги (начинающиеся с новой строки и символа «тире»). Сохранить диалоги в отдельных текстовых файлах. Использовать правила построения диалогов, применяющиеся в русскоязычных текстах.

## 2 Исходные данные

В качестве исходных данных программа использует текст, введённый пользователем.

## 3 Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

– если текст, введённый пользователем, не содержит букв, то есть невозможно выделить диалоги

– если текст, введённый пользователем, содержит пустые пропущенные строки

## 4 Математические методы и алгоритмы решения задач

Согласно постановке задачи для составления программы будет использован алгоритм, реализующий:

-разделение текст на диалоги

## 5 Форматы представления данных

Таблица 1- Переменные, используемые в программе

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| input | enum class Type | Выбор пользователя для ввода |
| output | enum class Type | Выбор пользователя для вывода |
| end\_program | enum class Answer | Выбор пользователя ДА/НЕТ |
| count | int | Счётчик диалогов |
| user\_choice | int | Пользовательский ввод |
| isDialog | bool | Проверка является ли строка диалогом |

Таблица 2- Структуры, используемые в программе

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| dialogs | Vector | Vector контейнер структур типа string |
| text | string | Исходный текст |

Для выбора вариаций алгоритма и пунктов меню используются константные выражения

Таблица 3- Константные значения, используемые в программе

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Тип | Значение | Описание |
| this\_file | const int | 2 | Пункт меню, ответственный за выбор пункта “да” |

Данные могут быть предоставлены двумя видами, с помощью консольного ввода и в виде файла с разрешением “.txt”, программа поддерживает многострочный консольный ввод.

## 6 Структура программы

Для оптимизации работы с кодом было принято решение разделить программу на модули.

|  |  |
| --- | --- |
| Имя | Описание |
| Main.cpp | Основной модуль |
| Menu | Модуль, включающий в себя модульное меню |
| DialogFinder | Модуль, включающий в себя алгоритмы выполнения задания |
| FileInput | Модуль, ответственный за файловый ввод |
| Check | Модуль, ответственный за обработку исключений |
| ConsoleOutput | Модуль, ответственный за консольный вывод |
| SaveInitialData | Модуль, ответственный за файловый вывод исходных данных |
| FileOutput | Модуль, ответственный за файловый вывод |
| GetFileName | Модуль, ответственный за проверку существования файла |
| TextInput | Модуль, ответственный за консольный ввод |

Таблица 4-Таблица модулей, составляющих программу

Каждый модуль разделён на функции, выполняющие какие-либо задачи.

Таблица №5-Таблица функций и методов, составляющих модуль “ DialogFinder.cpp”.

|  |  |
| --- | --- |
| Имя класса | Описание |
| DialogFinder | Класс хранящий диалоги |
| Имя функции | писание |
| FindDialogs | Функция ответственная за разделения текста на диалоги |
| GetDialog | Функция, возвращающая текст диалога |
| GetCountOfDialogs | Функция, возвращающая количество диалогов |

Таблица №6-Таблица функций, составляющих модуль “ Menu.сpp”.

|  |  |
| --- | --- |
| Имя | Описание |
| AscUserInputType | Функция, выбора типа ввода |
| AscOutputType | Функция, выбора типа вывода |
| AscInitialDataSaveMenu | Функция, выбора сохранения данных |

Таблица №7- Таблица функций и переменных, составляющих модуль “Check.h”.

|  |  |
| --- | --- |
| Имя функции | Описание |
| CheckMenu | Функция проверки ввода пунктов меню |

Таблица №8- Таблица функций и переменных, составляющих модуль “ FileInput.h”.

|  |  |
| --- | --- |
| Имя | Описание |
| FileInput | Функция ввода из файла |

Таблица №9- Таблица функций и переменных, составляющих модуль “ FileOutput.h”.

|  |  |
| --- | --- |
| Имя | Описание |
| FileOutputResult | Функция вывода результатов в файл |

Таблица №10- Таблица функций и переменных, составляющих модуль “ ConsoleOutput.h”.

|  |  |
| --- | --- |
| Имя | Описание |
| ConsoleOutput | Функция вывода результатов на консоль |

Таблица №11- Таблица функций и переменных, составляющих модуль “ SaveInitialData.h”.

|  |  |
| --- | --- |
| Имя | Описание |
| SaveInitialData | Функция сохранения исходных данных |

Таблица №12- Таблица функций и переменных, составляющих модуль “ GetFileName.h”.

|  |  |
| --- | --- |
| Имя | Описание |
| IsValidName | Функция проверки файла на существования |
| IsCanOpenFile | Функция проверки файла на возможность записи |
| GetFileName | Функция проверки имени файла |

Таблица №13- Таблица функций и переменных, составляющих модуль “ TextInput.h”.

|  |  |
| --- | --- |
| Имя | Описание |
| InputText | Функция ввода текста с консоли |

## 7 Блок-схемы алгоритмов

Рисунок 1-Блок-схема алгоритма разделения текста на слова

## 8 Результаты работы программы

В результате работы программы создаётся контейнер vector наполненный встречающимися диалогами и их число, данные которого могут быть выведены на консоль либо сохранены в файл. На экранных копиях работы программы показаны как корректный ввод данных, так и не корректный.

Экранные копии работы программы:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2-Экранная копия результатов работы программы

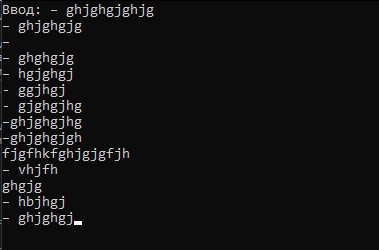


Рисунок 3-Экранная копия результатов работы программы

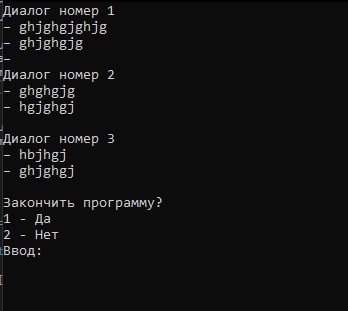


Рисунок 4-Экранная копия результатов работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5-Экранная копия результатов работы программы

## 9 Описание хода выполнения контрольной работы

1) В ходе контрольной работы было создано решение (Solution) в

интегрированной среде разработки Microsoft Visual Studio C++ 2017. В нём был

создан проект.

2) Для того, чтобы создать функцию многострочного ввода, было принято решение работать с методами класса «string».

3) При разработке функции DialogFinder было некорректно указан принцип разбиения текста на диалоге. Позже это было исправлено изменением блока условия.

## 10 Исходный текст программы

1)Main.cpp

// main.cpp

// Контрольная работа No 4.

//Выделить в заданном тексте все диалоги (начинающиеся //с новой строки и символа «тире»). Сохранить диалоги в //отдельных текстовых файлах. Использовать правила //построения диалогов, применяющиеся в русскоязычных //текстах.

// Студент группы 415, Шарашидзе Никита Леванович. 2022 год

#include <windows.h>

#include <iostream>

#include "Menu.h"

#include "DialogFinder.h"

#include "FileInput.h"

#include "FileOutput.h"

#include "TextInput.h"

#include "SaveInitialData.h"

#include "ConsoleOutput.h"

#include "Check.h"

enum class Type

{

console = 1,

file

};

enum class Answer

{

Yes = 1,

No

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_CTYPE, "RU");

std::cout << "Контрольная работа №2, вариант 5, сделано студентом группы 415 Шарашидзе Н.Л." << std::endl;

std::cout << "Выделить в заданном тексте все диалоги (начинающиеся с новой строки и символа «тире»)." << std::endl;

std::cout << "Сохранить диалоги в отдельных текстовых файлах." << std::endl;

std::cout << "Использовать правила построения диалогов, применяющиеся в русскоязычных текстах." << std::endl;

std::string text;

while (true)

{

Type input = static\_cast<Type>(AscUserInputType());

switch (input)

{

case Type::console:

text = InputText();

break;

case Type::file:

text = FileInput();

break;

}

DialogFinder finder(text);

finder.FindDialogs();

if (static\_cast<Answer>(AscInitialDataSaveMenu()) == Answer::Yes)

SaveInitialData(text);

Type output = static\_cast<Type>(AscOutputType());

switch (output)

{

case Type::console:

ConsoleOutput(finder);

break;

case Type::file:

FileOutputResult(finder);

}

std::cout << "Закончить программу?\n1 - Да\n2 - Нет" << std::endl;

Answer end\_program = static\_cast<Answer>(CheckMenu(2));

if (end\_program == Answer::Yes)

return 0;

system("cls");

}

}2) DialogFinder.h

#pragma once

#include <iostream>

#include <vector>

#include <string>

class DialogFinder

{

private:

std::string text;

std::vector <std::string> dialogs;

public:

DialogFinder(std::string textValue)

: text("\n" + textValue)

{}

void FindDialogs();

std::string GetDialog(int number);

std::vector <std::string> GetVector();

int GetCountOfDialogs();

};

2.1) DialogFinder.cpp

#include "DialogFinder.h"

void DialogFinder::FindDialogs()

{

std::string tempDialog;

bool isDialog = false;

int count = 0;

for (int i = 1; i < static\_cast<int>(text.size()); i++)

{

if (text[i - 1] == '\n' && text[i] == '–' && text [i + 1] == ' ')

{

isDialog = true;

count++;

}

if (isDialog)

tempDialog += text[i];

if (text[i] == '\n' && text[i + 1] != '–')

{

isDialog = false;

if (count > 1)

{

count = 0;

dialogs.push\_back(tempDialog);

tempDialog.clear();

}

else

{

count = 0;

tempDialog.clear();

}

}

}

}

std::string DialogFinder::GetDialog(int number)

{

if (number > static\_cast<int>(dialogs.size()))

return 0;

return dialogs[number];

}

std::vector<std::string> DialogFinder::GetVector()

{

return dialogs;

}

int DialogFinder::GetCountOfDialogs()

{

return static\_cast<int>(dialogs.size());

}

3)Check.h

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include <cstdlib>

int CheckMenu(int items);

3.1)Check.cpp

#include "Check.h"

int CheckMenu(int items)

{

std::string value\_input;

int value;

while (true)

{

std::cout << "Ввод: ";

std::getline(std::cin, value\_input);

std::cout << std::endl;

std::cin.sync();

try

{

value = stoi(value\_input);

if (value <= items && value >= 1)

return value;

else

throw value;

}

catch (std::exception&)

{

std::cout << "Не существующий пункт меню. Повторите ввод." << std::endl;

}

catch (int ex\_value)

{

std::cout << ex\_value << "Не существующий пункт меню. Повторите ввод." << std::endl;

}

}

}

4) ConsoleOutput.h

#pragma once

#include "DialogFinder.h"

void ConsoleOutput(DialogFinder finder);

4.1) ConsoleOutput.cpp

#include "ConsoleOutput.h"

void ConsoleOutput(DialogFinder finder)

{

system("cls");

if (finder.GetCountOfDialogs() == 0)

{

std::cout << "Не было найдено ни одного диалога." << std::endl;

std::cout << std::endl;

return;

}

for (int i = 0; i < finder.GetCountOfDialogs();i++)

{

std::cout << "Диалог номер " << i + 1 << std::endl;

std::cout << finder.GetDialog(i) << std::endl;

}

}

5) FileInput.h

#pragma once

#include <fstream>

#include <iostream>

#include <string>

#include "GetFileName.h"

std::string FileInput();

5)Fileoutput.cpp

#include "FileInput.h"

std::string FileInput()

{

system("cls");

std::string text = "";

std::ifstream file;

std::string file\_name;

file.exceptions(std::ifstream::badbit | std::ifstream::failbit);

while (true)

{

try

{

file\_name = GetFileName(input);

file.open(file\_name);

std::cout << "Файл успешно отктрыт." << std::endl;

break;

}

catch (std::exception&)

{

std::cout << "Ошибка доступа.\n" << "Хотите попробовать снова? 1 – Нет, 2 – Да" << std::endl;

std::cout << "Ввод: ";

int user\_choice = CheckMenu(2);

if (user\_choice != restart)

return text;

}

}

std::string buff;

while (true)

{

try

{

std::getline(file, buff);

text += buff + "\n";

}

catch (const std::exception&)

{

break;

}

}

//std::cin.clear();

//std::cin.ignore();

return text;

}

6.1) GetFileName.h

#pragma once

#include<iostream>

#include <fstream>

#include <string>

#include <algorithm>

#include "Check.h"

#define WIN\_NAME\_SIZE 22

enum

{

output = 0,

input = 1

};

const int restart = 2;

const std::string win\_name[WIN\_NAME\_SIZE] = { "CON", "PRN", "AUX", "NUL",

"COM1", "COM2", "COM3", "COM4", "COM5",

"COM6", "COM7", "COM8", "COM9", "LPT1",

"LPT2", "LPT3", "LPT4", "LPT5", "LPT6",

"LPT7", "LPT8", "LPT9" };

bool IsValidName(std::string name);

bool IsCanOpenFile(const std::string file\_name, const int stream\_type);

std::string GetFileName(int stream\_type);

6) GetFileName.cpp

#include "GetFileName.h"

bool IsValidName(std::string name)

{

std::cout << std::endl;

try

{

if (!(name.find(".txt") == std::string::npos))

{

std::size\_t temp\_sz = name.find(".txt");

std::string temp\_st = name.substr(temp\_sz);

if (temp\_st.size() != 4)

{

std::cout << "Неверный тип файла. Повторите ввод." << std::endl;

return false;

}

}

std::string temp1 = "", temp2 = name;

for (int i = 0; i < WIN\_NAME\_SIZE; i++)

{

temp1 = win\_name[i] + ".TXT";

if (temp1.size() == temp2.size())

{

std::transform(temp2.begin(), temp2.end(), temp2.begin(), ::toupper);

if (temp1 == temp2)

{

std::cout << "Введено зарезервированное операционной системой Windows слово. Повторите ввод." << std::endl;

return false;

}

}

}

return true;

}

catch (std::exception&)

{

std::cout << "Ошибка." << std::endl;

return false;

}

}

bool IsCanOpenFile(const std::string file\_name, const int stream\_type)

{

const int this\_file = 2;

try {

if (std::ifstream(file\_name))

{

if (stream\_type == output)

{

std::cout << "Файл уже существует. Вы хотите пересоздать его?" << std::endl

<< "1 - Нет." << std::endl

<< "2 - Да." << std::endl;

int user\_choice = CheckMenu(2);

if (user\_choice != this\_file)

return false;

}

}

else if (stream\_type == input)

{

std::cout << "Файла с таким названием не существует. Потворите ввод." << std::endl;

return false;

}

return true;

}

catch (std::exception&)

{

std::cout << "Ошибка." << std::endl;

return false;

}

}

std::string GetFileName(int stream\_type)

{

std::string name; //Название файла

while (true)

{

std::cout << "Введите название файла или полный путь до него." << std::endl << "Ввод: ";

std::getline(std::cin, name); //Ввод

if (!IsValidName(name)) //Если введённое название недоступно

{

std::cin.sync(); //Очистить поток cin

continue; //Уйти на новый цикл

}

try

{

if (IsCanOpenFile(name, stream\_type)) //Если можно открывать файл

return name; //Вернуть имя

}

catch (const std::exception&)

{

std::cout << "Ошибка." << std::endl;

}

}

}

7) Menu.h

#pragma once

#include <iostream>

#include "Check.h"

int AscUserInputType();

int AscOutputType();

int AscInitialDataSaveMenu();

7.1) Menu.cpp

#include "Menu.h"

int AscUserInputType()

{

int variant;

std::cout << "Введите 1 для консольного ввода" << std::endl

<< "Введите 2 для ввода из файла" << std::endl;

variant = CheckMenu(2);

return variant;

}

int AscOutputType()

{

int variant;

std::cout << "Введите 1 для консольного вывода" << std::endl

<< "Введите 2 для вывода в файла" << std::endl;

variant = CheckMenu(2);

return variant;

}

int AscInitialDataSaveMenu()

{

int variant;

std::cout << "1 - сохранить исходные данные" << std::endl

<< "2 - не сохранять исходные данные" << std::endl;

variant = CheckMenu(2);

return variant;

}

8) SaveInitialData.h

#pragma once

#include <fstream>

#include <iostream>

#include "GetFileName.h"

void SaveInitialData(std::string text);

8.1) SaveInitialData.cpp

#include "SaveInitialData.h"

void SaveInitialData(std::string text)

{

std::string file\_name = "";

std::ofstream file;

file.exceptions(std::ofstream::badbit | std::ofstream::failbit);

try

{

file\_name = GetFileName(output);

file.open(file\_name);

file << text;

}

catch (std::exception&)

{

std::cout << "Ошибка доступа." << std::endl;

}

file.close();

}

9) TextInput.h

#pragma once

#include <vector>

#include <iostream>

#include <string>

std::string InputText();

9.1) TextInput.cpp

#include "TextInput.h"

std::string InputText()

{

system("cls");

std::cout << "Ввод: ";

std::string text;

while (true)

{

std::string buffer = "";

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

if (buffer.empty())

{

if (text.empty())

{

std::cout << "Сначала введите текст: ";

continue;

}

break;

}

text += buffer + "\n";

}

return text;

}