

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГСН | | 09.00.00 | Информатика и вычислительная техника | | |
| Уровень образования | |  | Высшее образование – бакалавриат | | |
| Форма обучения | |  | Очная | | |
| Факультет | |  | Информационных технологий  и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Информационные технологии  и программирование | | |
| Курс | I | | | Группа | 4303 |

Отчёт по контрольной работе № 2

Вариант № 13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 4303 |  |  |  | Иванов Глеб Игоревич |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверил: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Федин Алексей Константинович |
|  |  |  |  |  |

# СОДЕРЖАНИЕ

[1 Задание №1 3](#_Toc161745186)

[1.1 Постановка задачи 3](#_Toc161745187)

[1.2 Исходные данные 3](#_Toc161745188)

[1.3 Особых ситуаций 3](#_Toc161745189)

[1.4 Математические методы и алгоритмы решения задачи 4](#_Toc161745190)

[1.5 Блок-схемы алгоритмов программы 5](#_Toc161745191)

[1.6 Форматы представления данных 6](#_Toc161745192)

[1.7 Структуры (модулей, функций) программы 9](#_Toc161745193)

[1.8 Ход выполнения работы 10](#_Toc161745194)

[1.9 Полученные результаты 10](#_Toc161745195)

[1.10 Исходный код программы 13](#_Toc161745196)

# 1 Задание №1

## Постановка задачи

В контрольной работе задается базовый и производный класс. Необходимо разработать поля и методы, наследуемые из базового класса, а также собственные компоненты производных классов. Базовый класс может быть абстрактным. Реализовать возможность получения списка объектов в контейнере. Предусмотреть необходимые проверки исходных данных.

Базовый класс – документ предприятия. Производный класс – письмо.. Реализовать возможность:

– получения списка документов,

– получения списка документ по имени,

– удаление по имени.

## Исходные данные

Назначение данных:

Программа получает на вход: тип, имя, адрес, адрасата, адресанта.

Представление в программе:

Имя - name\_,

тип - *type\_,*

адрес - *document\_filename\_,*

адрасата - *addressee\_,*

адресанта - sender\_

## 1.3 Особые ситуации

При выполнении задания были встречены ситуации, в которых программа может вести себя некорректно и непредсказуемо. Для решения этой проблемы было принято решение включить в исходный код следующие функции: *Num\_input()* – функция, которая проверяет введённое число на корректность.

Также были обработаны ошибки с некорректно введённым путём к файлу. Программа выдаёт ошибку “*Failed to open the file*” и просит ввести новый путь до тех пор, пока не получит существующий файл.

## Математические методы и алгоритмы решения задачи

В программе используется структура данных std::unordered\_map, которая представляет собой хеш-таблицу для хранения пар ключ-значение.

При начале работы с std::unordered\_map программа предоставляет выбор операции, которую нужно выполнить: добавление элемента, удаление элемента или поиск элемента.

В зависимости от выбранной операции, программа переходит к соответствующему этапу:

Добавление элемента (Add Element).

Удаление элемента (Remove Element).

Поиск элемента (Find Element).

Перед выполнением операции с элементом производится проверка наличия ключа в хеш-таблице. Если ключ существует, программа переходит к соответствующей операции, иначе операция завершается.

После выполнения операции может потребоваться перехеширование хеш-таблицы, например, при добавлении или удалении элемента.

Если перехеширование необходимо, происходит обновление хеш-таблицы с новыми хешами и распределением элементов.

## Блок-схемы алгоритмов программы

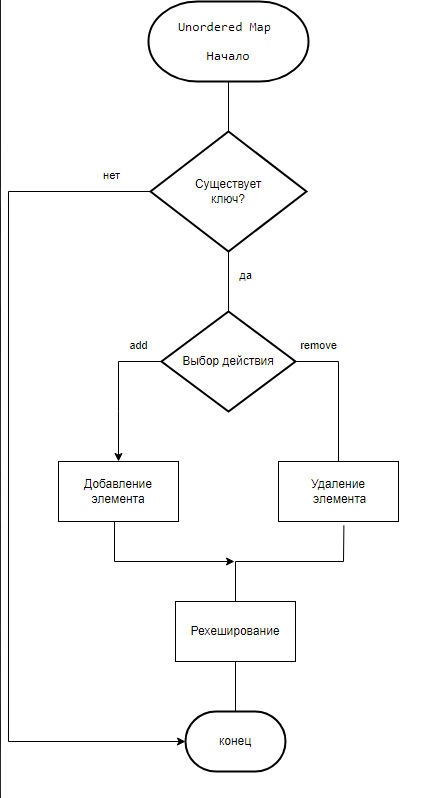


Рисунок 1 – Блок-схема алгоритма unordered\_map

## 1.6 Форматы представления данных

Таблица 1 – Переменные класса Document

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| type\_ | std::string | Тип документа |
| document\_filename\_ | std::string | Имя файла документа |
| name\_ | std::string | Имя документа |

Таблица 2 – Переменные класса Letter

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| addressee\_ | std::string | Адресат письма |
| sender\_ | std::string | Отправитель письма |

Таблица 3 – Основные переменные, используемые в программе

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| end\_of\_file | char | Символ конца файла, используемый для проверки, пуст ли файл. |
| filename | std::string | Хранит имя файла, с которым выполняется работа. |
| file | std::ofstream/std::ifstream | Объект файлового потока, используемый для работы с файлом. |
| is\_empty | bool | Флаг, указывающий, пуст ли файл. Используется в функции FileFilling для определения пустоты файла перед чтением данных. |
| file\_is\_open | uint16\_t | Код ошибки при открытии файла. Используется в функции FileFilling для определения успешности открытия |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| document\_list | std::unordered\_map<std::string, Document\*> | Хранит список документов в виде неупорядоченного ассоциативного контейнера, где ключ - строковое имя документа, а значение - указатель на объект документа. |
| size | size\_t | Хранит размер списка документов. |
| option\_of\_function | int | Хранит выбранный пользователем номер опции функции в главном меню. |
| filename\_for\_test | std::string | Хранит имя файла, используемого для проведения тестов. |
| name | std::string | Хранит имя документа, введенное пользователем при добавлении или удалении документа. |
| type | std::string | Хранит тип документа, введенный пользователем при добавлении документа. |
| filename | std::string | Хранит имя файла документа, введенное пользователем при добавлении документа. |
| addressee | std::string | Хранит адресата письма, введенного пользователем при добавлении письма. |
| sender | std::string | Хранит отправителя письма, введенного пользователем при добавлении письма. |
| exit | bool | Флаг, указывающий, нужно ли завершить работу программы. |
| option | uint16\_t | Хранит выбранный пользователем режим работы программы: start (режим работы с документами) или test (режим проведения тестов). |
| document | Document\* | Указатель на объект документа, найденного по имени. |

## 1.7 Структуры (модулей, функций) программы

Таблица 4 – Функции, составляющие программу

|  |  |
| --- | --- |
| Имя | Описание |
| Num\_input | Функция для ввода беззнакового целого числа с защитой от некорректного ввода |
| FileIsEmpty | Функция для проверки, пуст ли файл |
| OpenFile | Функция для открытия файла для чтения или записи |
| FileInput | Функция для записи данных пользователей в файл |
| FileFilling | Функция для чтения данных пользователей из файла |
| FindDocumentByName | Находит документ по имени. |
| DeleteDocument | Удаляет документ из списка. |
| AddDocument | Добавляет новый документ в список. |
| DisplayAll | Выводит на экран информацию о документах. |
| Test1 | Проверяет открытие файла. |
| Test2 | Проверяет открытие пустого файла. |
| Test3 | Проверяет открытие непустого файла. |
| Test4 | Проверяет загрузку данных из файла. |
| Test5 | Проверяет поиск документа по имени. |
| Function | Основная функция, выполняющая основной сценарий программы |

## 1.8 Ход выполнения работы

Во время выполнения программы в Visual Studio 2022 были освоены базовые навыки работы, такие как: настройка, запуск проекта, отладка процесса выполнения с помощью точек останова и просмотра значений переменных. Также были обнаружены проблемы в нехватке знаний необходимых инструментов. Были решены проблемы с созданием класса и добавлением новых клиентов вектор путем чтения многих форумов. Проблемы ввода и вывода данных при помощи файлов также были решены благодаря многочисленным форумам в интернете. При помощи YouTube были решены проблемы сортировки и вывода данных.

## 1.9 Полученные результаты

На рисунках 2 – 4 показана работа функций, обрабатывающих ошибки, на рисунках 5 – 6 показана работа функций сортировок и тестов.

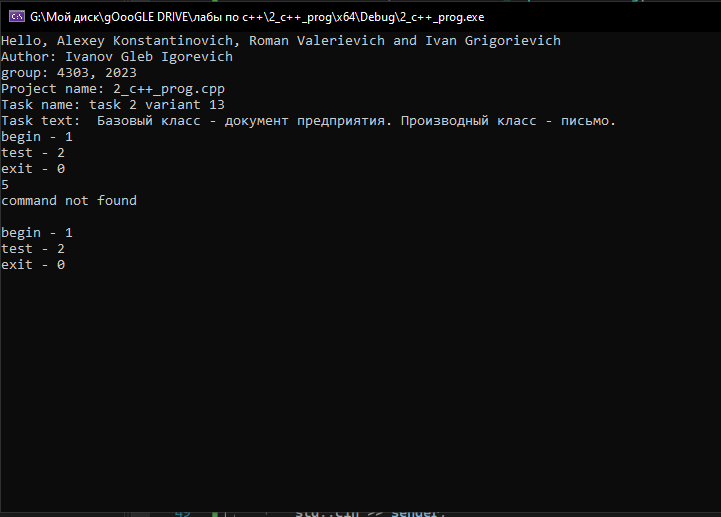


Рисунок 2 – Экранная копия результата работы разработанной программы задания №2

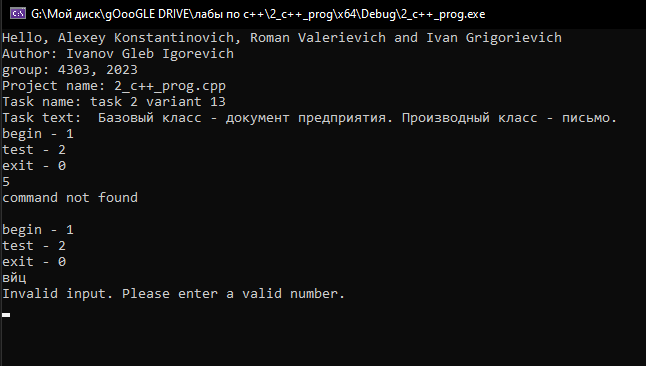


Рисунок 3 – Экранная копия результата работы разработанной программы задания №2

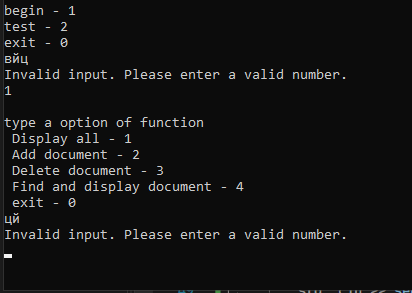


Рисунок 4 – Экранная копия результата работы разработанной программы задания №2

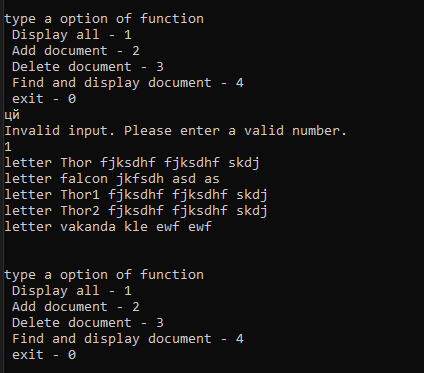


Рисунок 5 – Экранная копия результата работы разработанной программы задания №2

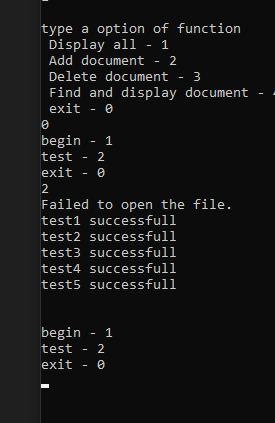


Рисунок 6 – Экранная копия результата работы разработанной программы задания №2

## 1.10 Исходный код программы

#include <iostream>

#include <algorithm>

#include <fstream>

#include <vector>

#include <string>

using std::string;

class Letter :public Document {

public:

void SetAddressee(std::string addressee) {

addressee\_ = addressee;

}

void SetSender(std::string sender) {

sender\_ = sender;

}

auto GetAddressee() {

return addressee\_;

}

auto GetSender() {

return sender\_;

}

void Display() override {

std::cout << type\_ << " " << name\_ << " " << document\_filename\_ << " " << addressee\_ << " " << sender\_ << " ";

}

private:

std::string addressee\_; //получатель

std::string sender\_; //отправитель

};

Document::Document() {

}

uint32\_t Document::GetID() {

return ID\_;

}

std::string Document::GetDocumentFilename() {

return document\_filename\_;

}

void Document::SetDocumentFilename(std::string filename) {

document\_filename\_ = filename;

}

std::string Document::GetName() {

return name\_;

}

void Document::SetName(std::string name) {

name\_ = name;

}

std::string Document::GetType() {

return type\_;

}

void Document::SetType(std::string type) {

type\_ = type;

}

//fielname:

// ./text.txt

// name

// text

void Document::SetNameFromFilename() {

for (size\_t i = 0; i < document\_filename\_.size(); i++)

{

if (document\_filename\_[i] == '/') {

for (size\_t j = i+1; j < document\_filename\_.size(); j++)

{

if (document\_filename\_[j] == '.') return;

name\_.push\_back(document\_filename\_[j]);

}

}

}

}

bool FileIsEmpty(std::string filename) {

std::ifstream file;

file.open(filename);

char end\_of\_file;

if (file.is\_open()) {

file >> end\_of\_file;

file.close();

if (end\_of\_file < 0) return true;

else return false;

}

return false;

}

bool OpenFile(std::string filename, std::ofstream& file) { // запись

// Попытаться открыть файл

file.open(filename, std::ios::out);// для старые данные удаляются

if (file.is\_open()) {

//std::cout << "File successfully opened" << "\n";

return true;

}

else {

std::cout << "Failed to open the file." << "\n";

return false;

}

}

uint16\_t OpenFile(std::string filename, std::ifstream& file) { //чтение

bool is\_empty = FileIsEmpty(filename);

// Попытаться открыть файл

file.open(filename);

if (file.is\_open() && is\_empty) {

// std::cout << "File is not empty, failed to open the file. " << "\n";

return file\_open\_empty;

}

if (file.is\_open()) {

//std::cout << "File successfully opened" << "\n";

return file\_open;

}

else {

std::cout << "Failed to open the file." << "\n";

return false;

}

}

void FileInput(std::unordered\_map <std::string, Document\*>& document\_list) {

std::ofstream file;

while (true) {

//std::cout << "please tape any filenameway" << "\n";

std::string filename = "./test.txt";

//std::cin >> filename;

//открытие файла:

if (OpenFile(filename, file)) {

break;

}

}

// Запись данных в файл

file << document\_list.size() << "\n";

for (auto& [name,document] : document\_list) {

file << document->GetType() << " " << document->GetName() << " " << document->GetDocumentFilename() << " " << dynamic\_cast<Letter\*>(document)->GetAddressee() << " " << dynamic\_cast<Letter\*>(document)->GetSender() << " \n";

}

//std::cout << "Data successfully written to the file." << std::endl;

// Закрытие файла

file.close();

}

void FileFilling(std::unordered\_map <std::string,Document\*> &document\_list, size\_t& size\_of\_book) {

std::ifstream file;

uint16\_t file\_is\_open =0;

while (true) {

std::string filename = "./test.txt";

//std::cout << "please tape any filenameway" << "\n";

//std::cin >> filename;

//открытие файла:

file\_is\_open = OpenFile(filename, file);

if (file\_is\_open) {

break;

}

}

if (file\_is\_open == file\_open\_empty) {

size\_of\_book = 0;

}

else file >> size\_of\_book;

// Чтение данных из файла

for (int i = 0; i < size\_of\_book; i++) { // заполнение

std::string type;

std::string name;

std::string document\_filename;

file >> type;

file >> name;

file >> document\_filename;

file.ignore(); // clear input buffer

//сделать выбор для разных типов

if (type == "letter") {

auto\* document = new Letter;

std::string addressee; //получатель

std::string sender;

file >> addressee;

file >> sender;

document->SetSender(sender);

document->SetAddressee(addressee);

document->SetDocumentFilename(document\_filename);

document->SetName(name);

document->SetType(type);

document\_list.insert({ name, document });

}

}

// Закрытие файла

file.close();

}

void FileFilling(std::unordered\_map <std::string, Document\*>& document\_list, size\_t& size\_of\_book, std::string current\_filename) { // для тестов

std::ifstream file;

uint16\_t file\_is\_open = 0;

while (true) {

std::string filename = current\_filename;

//std::cout << "please tape any filenameway" << "\n";

//std::cin >> filename;

//открытие файла:

file\_is\_open = OpenFile(filename, file);

if (file\_is\_open) {

break;

}

}

if (file\_is\_open == file\_open\_empty) {

size\_of\_book = 0;

}

else file >> size\_of\_book;

// Чтение данных из файла

// Чтение данных из файла

for (int i = 0; i < size\_of\_book; i++) { // заполнение

std::string type;

std::string name;

std::string document\_filename;

file >> type;

file >> name;

file >> document\_filename;

file.ignore(); // clear input buffer

//сделать выбор для разных типов

if (type == "letter") {

auto\* document = new Letter;

std::string addressee; //получатель

std::string sender;

file >> addressee;

file >> sender;

document->SetSender(sender);

document->SetAddressee(addressee);

document->SetDocumentFilename(document\_filename);

document->SetName(name);

document->SetType(type);

document\_list.insert({ name, document });

}

}

// Закрытие файла

file.close();

}

Document\* FindDocumentByName(std::unordered\_map <std::string, Document\*>& document\_list, std::string name) {

if (document\_list.empty()) return nullptr;

else if (!document\_list.contains(name)) return nullptr;

else if (document\_list.contains(name)) return document\_list[name];

else return nullptr;

}

void DeleteDocument(std::unordered\_map <std::string, Document\*>& document\_list, std::string name) {

document\_list.erase(name);

}

void AddDocument(std::unordered\_map <std::string, Document\*>& document\_list) {

std::cout << "example: type (letter) \"name\" filename ....." << "\n";

std::string type;

std::string name;

std::string filename;

while (true) {

std::cin >> type;

if (type == "letter" /\* || type == dontSave\*/) {

break;

}

std::cout << "Invalid input. Please enter a valid type." << "\n";

std::cin.ignore();

}

std::cout << "\n";

std::cin >> name;

std::cin >> filename;

//сделать выбор для разных типов

if (type == "letter") {

auto\* document = new Letter;

std::cout << "example continue: addressee, sender\n ";

std::string addressee; //получатель

std::string sender;

std::cin >> addressee;

std::cin >> sender;

document->SetSender(sender);

document->SetAddressee(addressee);

document->SetDocumentFilename(filename);

document->SetName(name);

document->SetType(type);

document\_list.insert({ name, document });

}

std::cout << "\n" << "\n";

}

void DisplayAll(std::unordered\_map <std::string, Document\*>& document\_list) {

for (const auto& [name, document] : document\_list) {

document->Display();

std::cout << "\n";

}

}

void Test1() {

std::string filename\_for\_test = "./test1";

std::ifstream file;

if (OpenFile(filename\_for\_test, file) == 0) { // н открываеится

std::cout << "test1 successfull\n";

file.close();

}

}

void Test2() {

std::string filename\_for\_test = "./test2.txt"; //пустой

std::ifstream file;

if (OpenFile(filename\_for\_test, file) == file\_open\_empty) {

std::cout << "test2 successfull\n";

file.close();

}

}

void Test3() {

std::string filename\_for\_test = "./test3.txt";

std::ifstream file;

if (OpenFile(filename\_for\_test, file) == file\_open) {

std::cout << "test3 successfull\n";

file.close();

}

}

void Test4() {

std::unordered\_map <std::string, Document\*> document\_list;

size\_t size;

std::string filename\_for\_test = "./test4.txt";

// написать filefilllingh с filename

FileFilling(document\_list, size, filename\_for\_test);

Document\* document = FindDocumentByName(document\_list, "Nos");//добавить имя

if (document == nullptr) {

std::cout << "test4 successfull\n";

}

}

void Test5() {

std::unordered\_map <std::string, Document\*> document\_list;

size\_t size;

std::string filename\_for\_test = "./test5.txt";

// написать filefilllingh с filename

FileFilling(document\_list, size, filename\_for\_test);

Document\* document = FindDocumentByName(document\_list, "KIngdomCome");

if (document != nullptr) {

std::cout << "test5 successfull\n";

}

}

void Function(uint16\_t option) {

// запускается программа

// открывается база данных

// можно вывести базу или добавить элемент или удалить

// или поиск по имени

std::unordered\_map <std::string, Document\*> document\_list;

size\_t size;

FileFilling(document\_list, size); // база данных записалась

bool exit = false;

if (option == start) { //обычное

while (true) {

int option\_of\_function = 0;

std::cout << "\n";

std::cout << "type a option of function\n Display all - 1\n Add document - 2\n Delete document - 3\n Find and display document - 4\n exit - 0" << "\n";

while (true) {

option\_of\_function = static\_cast<int>(Num\_input());

if (option\_of\_function == display) {//Display all

break;

}

else if (option\_of\_function == Add\_document) {// Add document

break;

}

else if (option\_of\_function == Delete\_document) {// Delete document

break;

}

else if (option\_of\_function == Findand\_display\_document) {// Find and display document

break;

}

else if (option\_of\_function == Exit) {// exit

exit = true;

break;

}

else {

std::cout << "Invalid input. Please enter a valid command." << "\n";

}

}

if (exit == true) {

break;

}

switch (option\_of\_function) {

case display:

DisplayAll(document\_list);

break;

case Add\_document:

AddDocument(document\_list);

break;

case Delete\_document:

{

std::cout << "type a name to delete" << "\n";

std::string name;

std::cin >> name;

if (FindDocumentByName(document\_list, name) == nullptr) {

std::cout << "there is not document with this name\n";

break;

}

DeleteDocument(document\_list, name);

}

break;

case Findand\_display\_document:

{

std::cout << "type a name to find" << "\n";

std::string name;

std::cin >> name;

if (FindDocumentByName(document\_list, name) == nullptr) {

std::cout << "there is not document with this name\n";

break;

}

Document\* document = FindDocumentByName(document\_list, name);

document->Display();

}

break;

}

std::cout << "\n";

}

}

else if (option == test) {

Test1();

Test2();

Test3();

Test4();

Test5();

}

FileInput(document\_list);

}

int main()

{

setlocale(LC\_CTYPE, "Russian");

int command = 0;

std::cout << "Hello, Alexey Konstantinovich, Roman Valerievich and Ivan Grigorievich\n"

<< "Author: Ivanov Gleb Igorevich\n"

<< "group: 4303, 2023\n"

<< "Project name: 2\_c++\_prog.cpp\n"

<< "Task name: task 2 variant 13\n"

<< "Task text: Базовый класс – документ предприятия. Производный класс – письмо.\n";

while (true) {

std::cout << "begin - 1\ntest - 2 \nexit - 0\n";

command = static\_cast<int>(Num\_input());

//std::cout << "\n\n";

if (command == start) {

Function(start);

}

else if (command == test) {

Function(test);

std::cout << "\n\n";

}

else if (command == Exit) {

std::cout << "prog has been finished";

std::cout << "\n\n";

break;

}

else {

std::cout << "command not found" << "\n" << "\n";

}

}

}