Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа**

**по курсу «ООП»**

**Тема:**

**Проектирование структуры классов.**

|  |  |
| --- | --- |
| Студент: | Игитова А.А, |
| Группа: | М80-208Б-18 |
| Преподаватель: | Журавлев А.А. |
| Вариант: | 9 |
| Оценка: |  |
| Дата: |  |

Москва

2019

**1. Постановка задачи**

Спроектировать простейший текстовый редактор.

Требование к функционалу редактора:

* создание нового документа
* импорт документа из файла
* экспорт документа в файл
* добавление в документ
* удаление из документа
* отображение документа на экране
* реализовать операцию undo, отменяющую последнее сделанное действие. Должно действовать для операций добавления/удаления.

**2. Код программы на языке C++**

**text.cpp:**

#include <iostream>

#include <string>

#include <cctype>

#include "command.h"

#include "document.h"

#include "editor.h"

int main() {

editor editor\_;

std::string str;

while (std::cin >> str) {

if (islower(str[0])) {

editor\_.InsertInDocument(str);

} else if (str == "Right" || str == "R") {

editor\_.CursorRightInDocument();

} else if (str == "Left" || str == "L") {

editor\_.CursorLeftInDocument();

} else if (str == "Print" || str == "P") {

editor\_.PrintDocument();

} else if (str == "Create" || str == "C") {

std::cout << "Input name of file: ";

std::string fileName;

std::cin >> fileName;

editor\_.CreateDocument(fileName);

} else if (str == "Backspace" || str == "B") {

editor\_.DeleteInDocument();

} else if (str == "Undo" || str == "U") {

try {

editor\_.Undo();

} catch (std::logic\_error &e) {

std::cout << e.what();

}

} else if (str == "Load") {

std::cout << "Input name of file: ";

std::string fileName;

std::cin >> fileName;

editor\_.LoadDocument(fileName);

} else if (str == "Save" || str == "S") {

editor\_.SaveDocument();

} else if (str == "Exit" || str == "Quit" || str == "E" || str == "Q") {

break;

}

}

return 0;

}

**editor.h:**

#ifndef EDITOR\_H

#define EDITOR\_H 1

#include <stack>

#include <string>

#include <memory>

#include <iostream>

#include "command.h"

#include "document.h"

struct editor {

editor() : doc\_{nullptr}, history\_{} {}

void InsertInDocument(std::string &str);

void DeleteInDocument();

void CursorLeftInDocument();

void CursorRightInDocument();

void CreateDocument(std::string &name);

bool DocumentExist();

void SaveDocument();

void LoadDocument(std::string &name);

void PrintDocument();

void Undo();

private:

std::shared\_ptr<document> doc\_;

std::stack<std::shared\_ptr<command>> history\_;

};

#endif //EDITOR\_H

**editor.cpp:**

#include "editor.h"

#include "document.h"

#include "command.h"

void editor::InsertInDocument(std::string &str) {

if (doc\_ == nullptr) {

std::cout << "No document\n";

return;

}

std::shared\_ptr<command> cmd = std::shared\_ptr<command>(new InsertCmd(doc\_, str.size(), doc\_->GetCursor()));

doc\_->Insert(str);

history\_.push(cmd);

}

void editor::DeleteInDocument() {

if (doc\_ == nullptr) {

std::cout << "No document\n";

return;

}

size\_t index = doc\_->GetCursor();

if (index == 0) {

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

return;

}

std::shared\_ptr<command> cmd = std::shared\_ptr<command>(new DeleteCmd(doc\_->GetElem(index - 1), index - 1, doc\_));

doc\_->Delete();

history\_.push(cmd);

}

void editor::CursorLeftInDocument() {

if (doc\_ == nullptr) {

std::cout << "No document\n";

return;

}

doc\_->CursorLeft();

}

void editor::CursorRightInDocument() {

if (doc\_ == nullptr) {

std::cout << "No document\n";

return;

}

doc\_->CursorRight();

}

void editor::CreateDocument(std::string &name) {

doc\_ = std::make\_shared<document>(name);

}

bool editor::DocumentExist() {

return doc\_ != nullptr;

}

void editor::SaveDocument() {

if (doc\_ == nullptr) {

std::cout << "No document\n";

return;

}

std::string name = doc\_->GetName();

doc\_->Save(name);

}

void editor::LoadDocument(std::string &name) {

try {

doc\_ = std::make\_shared<document>(name);

doc\_->Load(name);

while (!history\_.empty()) {

history\_.pop();

}

} catch (std::runtime\_error &err) {

std::cout << err.what();

}

}

void editor::PrintDocument() {

doc\_->Print();

}

void editor::Undo() {

if (history\_.empty()) {

throw std::logic\_error("History is empty\n");

}

std::shared\_ptr<command> last = history\_.top();

last->UnExecute();

history\_.pop();

}

**command.h:**

#ifndef COMMAND\_H

#define COMMAND\_H 1

#include "document.h"

struct command {

virtual void UnExecute() = 0;

virtual ~command() = default;

protected:

std::shared\_ptr<document> doc\_;

};

struct InsertCmd : command {

InsertCmd(std::shared\_ptr<document> &doc, size\_t len, size\_t cursor);

void UnExecute() override;

private:

size\_t len\_;

size\_t cursor\_;

};

struct DeleteCmd : command {

DeleteCmd(std::string str, size\_t index, std::shared\_ptr<document> &doc);

void UnExecute() override;

private:

std::string str\_;

size\_t index\_;

};

#endif //COMMAND\_H

**command.cpp:**

#include "command.h"

InsertCmd::InsertCmd(std::shared\_ptr<document> &doc, size\_t len, size\_t cursor) {

doc\_ = doc;

len\_ = len;

cursor\_ = cursor;

}

void InsertCmd::UnExecute() {

doc\_->RemoveLast(len\_, cursor\_);

}

DeleteCmd::DeleteCmd(std::string str, size\_t index, std::shared\_ptr<document> &doc) {

str\_ = str;

index\_ = index;

doc\_ = doc;

}

void DeleteCmd::UnExecute() {

doc\_->InsertIndex(str\_, index\_);

}

**document.h:**

#ifndef DOCUMENT\_H

#define DOCUMENT\_H 1

#include <memory>

#include <vector>

#include <string>

#include <fstream>

#include <iostream>

struct document {

document(std::string &name) : name\_(name), buffer\_{}, cursor\_{0} {}

void Save(const std::string &name) const;

void Load(const std::string &name);

void Insert(std::string &str);

void InsertIndex(std::string &str, size\_t index);

void Delete();

bool CursorLeft();

bool CursorRight();

void Print();

void RemoveLast(size\_t len, size\_t cursor);

size\_t GetCursor();

std::string GetElem(size\_t index);

std::string GetName();

private:

std::string name\_;

std::string buffer\_;

size\_t cursor\_;

};

#endif //DOCUMENT\_H

**document.cpp:**

#include "document.h"

void document::Save(const std::string &name) const {

std::ofstream file{name};

if (!file) {

throw std::runtime\_error("File isn't opened\n");

}

file << cursor\_ << " ";

file << buffer\_;

}

void document::Load(const std::string &name) {

std::ifstream file{name};

if (!file) {

throw std::runtime\_error("File isn't opened\n");

}

file >> cursor\_;

buffer\_.clear();

file >> buffer\_;

name\_ = name;

}

void document::Insert(std::string &str) {

buffer\_.insert(cursor\_, str);

cursor\_ += str.size();

}

void document::InsertIndex(std::string &str, size\_t index) {

buffer\_.insert(index, str);

cursor\_++;

}

void document::Delete() {

buffer\_.erase(cursor\_ - 1, 1);

cursor\_--;

}

bool document::CursorLeft() {

if (cursor\_ == 0) {

return false;

}

cursor\_--;

return true;

}

bool document::CursorRight() {

if (cursor\_ == buffer\_.size()) {

return false;

}

cursor\_++;

return true;

}

void document::Print() {

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

}

void document::RemoveLast(size\_t len, size\_t cursor) {

buffer\_.erase(cursor, len);

cursor\_ = cursor;

}

size\_t document::GetCursor() {

return cursor\_;

}

std::string document::GetElem(size\_t index) {

std::string str;

str += buffer\_[index];

return str;

}

std::string document::GetName() {

return name\_;

}

**3. Ссылка на репозиторий на GitHub.**

https://github.com/m4rkovka/oop\_exercise\_07/tree/master/task

**4. Набор testcases.**

**test\_01.txt:**

C

new\_file

abcdef L L L xyz R p P U P U P uuu P Q

**test\_02.txt:**

C

file\_name

abcdef L R R xyz L p P U P U P uuu P Q

**5. Результаты выполнения тестов.**

**test\_01.test:**

Input name of file: abcxyzdpef

abcxyzdef

abcdef

abcuuudef

**test\_02.txt:**

Input name of file: abcdefxypz

abcdefxyz

abcdef

abcdefuuu

**6. Объяснение результатов работы программы.**

В проекте есть 6 файлов. Файл document.h, в котором реализован класс Document, содержащий следующие методы-члены:

* Конструкторы
* Сохранение документа в файл
* Загрузка документа из файла
* Вставка в положение курсора
* Вставка по индексу
* Удаление
* Сдвиг курсора влево
* Сдвиг курсора вправо
* Вывод в консоль
* Удаление последнего
* Получение курсора
* Получение элемента по индексу
* Получение имени документа

И следующие переменные:

* Cursor Положение курсора.
* Name. Имя документа.
* Buffer. Буфер для хранения строки.

Файл editor.h содержит основной функционал редактора.

Файл command.h содержит команды добавления и удаления.

Файл text.cpp основной файл, в котором находится функция main.

**7. Вывод.**

Выполняя данную лабораторную, я получила практические навыки в проектировании структуры классов приложения. На мой взгляд умение правильно проектировать классы приложения — это очень нужный навык, т. к. правильно структурированные классы, на мой взгляд, добавляют гибкости программе, её гораздо легче будет исправлять.