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.