Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

# Лабораторная работа
# по курсу «Объектно-ориентированное программирование»
# III Семестр

# Задание 7
# Вариант 27
# Основы метапрограммирования



| | |
|---|---|
| Студент: | Чурсина Н. А. |
| Группа: | М8О-208Б-18 |
| Преподаватель: | Журавлев А.А. |
| | |
| Оценка: | |
| Дата: | |



Москва 2019

# 1. Код программы на языке C++

**Main.cpp**

```
#include <iostream>
#include <string>
#include "redactor.h" //

int main() {
   std::string str;
   std::string str1;

          editor r;

   std::cout << "Input M or m for print menu: " << std::endl;
          while (std::cin >> str) {
                    if ( str == "I" or str =="i" ){
         std::cin >> str1;
         r.insert(str1);
      }else if (str == "M" or str =="m" ){
         std::cout << "! not case-sensitive program ! \n"
                << "M - Print menu\n"
                << "C - Create document\n"
                << "Load - Load document\n"
                << "S - Save document\n"
                << "I - Insert text\n"
                << "L - Cursor left\n"
                << "R - Cursor right\n"
                << "Backspace or B - Remove\n"
                << "Undo or U - Undo\n"
                << "Print or P - Print document\n"
                << "Q - Exit\n";
                   } else if (str == "Create" or str == "C" or str == "c" or str == "create") {
                            std::cout << "Input name of file: ";
                            std::string fileName;
                            std::cin >> fileName;
                            r.create(fileName);
                   } else if (str == "Backspace" or str == "B" or str == "b" or str == "backspace") {
                            r.delete_();
                   } else if (str == "Undo" or str == "U" or str == "u" or str == "undo") {
                            try {
                                     r.undo();
                            } catch (std::logic_error &e) {
                                     std::cout << e.what();
                            }
                   } else if (str == "Load"  or str == "load") {
                            std::cout << "Input name of file: ";
                            std::string fileName;
                            std::cin >> fileName;
                            r.load(fileName);
                   } else if (str == "Save" or str == "S" or str == "s" or str == "save") {
                            r.save();
                   } else if ( str == "Q" or str == "q") {
                            break;
```

```
    } else if (str == "Right" or str == "R" or str == "r" or str == "right") {
        r.cursorRight();
    } else if (str == "Left" or str == "L" or str == "l" or str == "left") {
        r.cursorLeft();
    } else if (str == "Print" or str == "P" or str == "p" or str == "print") {
        r.print();
                } else {
                    std::cout << "Incorrect input\n"; }
        }

        return 0;
}
```

## redactor.h

```
#ifndef REDACTOR_H
#define REDACTOR_H 1

#include <stack>
#include <string>
#include <memory>
#include <iostream>

#include "cmd.h"
#include "doc.h"

struct editor {

        editor() : doc{nullptr}, memory{} {}

        void insert(std::string &str);
        void delete_();
        void cursorLeft();
        void cursorRight();
        void create(std::string &name);
        void save();
        void load(std::string &name);
        void print();
        void undo();

private:
        std::shared_ptr<document> doc;
        std::stack<std::shared_ptr<command>> memory;
};

#endif //REDACTOR_H
```

## redactor.cpp

```
#include "redactor.h"
#include "doc.h"
#include "cmd.h"

void editor::insert(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_doc(str);
        memory.push(cmd);
        std::cout << "Insert - OK\n";
}

void editor::delete_() {
        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_doc();
        memory.push(cmd);
   std::cout << "Delete - OK\n";
}

void editor::cursorLeft() {
        if (doc == nullptr) {
                std::cout << "No document\n";
                return;
        }
        doc->cursorLeft_doc();
   std::cout << "Left - OK\n";
}

void editor::cursorRight() {
        if (doc == nullptr) {
                std::cout << "No document\n";
                return;
        }
        doc->cursorRight_doc();
   std::cout << "Right - OK\n";
}

void editor::create(std::string &name) {
        doc = std::make_shared<document>(name);
   std::cout << "Create - OK\n";
}


void editor::save() {
        if (doc == nullptr) {
                std::cout << "No document\n";
                return;
        }
        std::string name = doc->getName();
        doc->save_doc(name);
   std::cout << "Save - OK\n";
}
```

```
void editor::load(std::string &name) {
        try {
                doc = std::make_shared<document>(name);
                doc->load_doc(name);
                while (!memory.empty()) {
                        memory.pop();
                }
    std::cout << "Load - OK\n";
        } catch (std::runtime_error &err) {
                std::cout << err.what();
        }
}

void editor::print() {
   if (doc == nullptr) {
      std::cout << "No document\n";
      return;
   }
        doc->print_doc();
}

void editor::undo() {
        if (memory.empty()) {
                throw std::logic_error("History is empty\n");
        }
        std::shared_ptr<command> last = memory.top();
        last->undoThis();
        memory.pop();
   std::cout << "Undo - OK\n";
}
```

# doc.h

```
#ifndef DOC_H
#define DOC_H 1

#include <memory>
#include <vector>
#include <string>
#include <fstream>
#include <iostream>

struct document {
        document(std::string &name) : name_(name), strbuff{}, position{0} {}

        void save_doc(const std::string &name) const;
        void load_doc(const std::string &name);
        void insert_doc(std::string &str);
        void insertIndex_doc(std::string &str, size_t index);
        void delete_doc();
        bool cursorLeft_doc();
        bool cursorRight_doc();
        void print_doc();
        void removeEnd(size_t len, size_t cursor);

        size_t getCursor();
```

```
        std::string getElem(size_t index);
        std::string getName();

private:
        std::string name_;
        std::string strbuff;
        size_t position;
};

#endif //DOC_H
```

## doc.cpp

```
#include "doc.h"

void document::save_doc(const std::string &name) const {
        std::ofstream file{name};
        if (!file) {
                throw std::runtime_error("ERROR : File isn't opened\n");
        }
        file << position << " ";
        file << strbuff;
}

void document::load_doc(const std::string &name) {
        std::ifstream file{name};
        if (!file) {
                throw std::runtime_error("ERROR : File isn't opened\n");
        }
        file >> position;
        strbuff.clear();
        file >> strbuff;
        name_ = name;
}

void document::insert_doc(std::string &str) {
        strbuff.insert(position, str);
        position += str.size();
}

void document::insertIndex_doc(std::string &str, size_t index) {
        strbuff.insert(index, str);
        position++;
}

void document::delete_doc() {
        strbuff.erase(position - 1, 1);
        position--;
}

bool document::cursorLeft_doc() {
        if (position == 0) {
                return false;
        }
        position--;
```

```
	return true;
}

bool document::cursorRight_doc() {
	if (position == strbuff.size()) {
		return false;
	}
	position++;
	return true;
}

void document::print_doc() {
	std::cout << strbuff << "\n";
}

void document::removeEnd(size_t len, size_t cursor) {
	strbuff.erase(cursor, len);
	position = cursor;
}

size_t document::getCursor() {
	return position;
}

std::string document::getElem(size_t index) {
	std::string str;
	str += strbuff[index];
	return str;
}

std::string document::getName() {
	return name_;
}
```

## cmd.h

```
#ifndef CMD_H
#define CMD_H 1

#include "doc.h"

struct command {
	virtual void undoThis() = 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 undoThis() override;

private:
```

```
        size_t len_;
        size_t cursor_;
};

struct DeleteCmd : command {
        DeleteCmd(std::string str, size_t index, std::shared_ptr<document> &doc);

        void undoThis() override;

private:
        std::string str_;
        size_t index_;
};

#endif //CMD_H
```

**cmd.cpp**

```
#include "cmd.h"

InsertCmd::InsertCmd(std::shared_ptr<document> &doc, size_t len, size_t cursor) {
        doc_ = doc;
        len_ = len;
        cursor_ = cursor;
}

void InsertCmd::undoThis() {
        doc_->removeEnd(len_, cursor_);
}

DeleteCmd::DeleteCmd(std::string str, size_t index, std::shared_ptr<document> &doc) {
        str_ = str;
        index_ = index;
        doc_ = doc;
}

void DeleteCmd::undoThis() {
        doc_->insertIndex_doc(str_, index_);
}
```

# 2. Ссылка на репозиторий на Github

https://github.com/kwk18/oop_exercise_07

# 3.Набор testcases

1. **Входные данные:**

```
Input M for print menu:
M
M - Print menu
C - Create document
Load - Load document
S - Save document
I - Insert text
L - Cursor left
R - Cursor right
Backspace or B - Remove
Undo or U - Undo
Print or P - Print document
Q - Exit
C
Input name of file: t
Create - OK
I
abcD1
Insert - OK
S
Save - OK
L
Left - OK
B
Delete - OK
P
abc1
U
Undo - OK
S
Save - OK
P
abcD1
```

# 4. Объяснение результатов программы

doc.h главный файл, в котором реализован класс Document, содержащий следующие методы-члены:

*void save_doc*- сохранение документа в файл
*void load_doc* - загрузка документа из файла
*void insert_doc* - вставка в положение курсора
*void insertIndex_doc* - вставка по индексу (нужна для команд добавления и удаления)
*void delete_doc* - удаление
*bool cursorLeft_doc* - сдвиг курсора влево
*bool cursorRight_doc* - сдвиг курсора вправо
*void print_doc* - вывод в консоль
*void removeEnd* - удаление последнего

Получение курсора
Получение элемента по индексу
Получение имени документа
Переменные:
Position – отвечает за положение курсора в файле;
Name - имя документа;
Strbuff - буфер для хранения строки;

redactor.h содержит основной функционал редактора;
cmd.h содержит команды добавления и удаления;
main.cpp - основной файл, в котором находится функция main и минимальный пользовательский интерфейс;

Вывод: Проделав данную работу я получила практические навыки проектирования классов.