Министерство науки и высшего образования РФ

ФГАОУ ВО ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИУ)

Институт естественных и точных наук

Факультет математики, механики и компьютерных технологий

Кафедра прикладной математики и программирования

«Текстовый редактор»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ   
по дисциплине «Объектно-ориентированное программирование»

ЮУрГУ–01.03.02.2020.108.ПЗ КР

|  |  |  |
| --- | --- | --- |
|  |  | *Руководитель,*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Демидов А.К.*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022г.* |
|  |  | *Автор работы:*  *Студента группы: ЕТ – 212*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Постаушкин Ф.Ю.*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022г.* |
|  |  | *Работа защищена с оценкой*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.* |

Челябинск – 2022

Федеральное государственное автономное образовательное учреждение высшего образования

«Южно-Уральский государственный университет   
(национальный исследовательский университет)»

Институт естественных и точных наук

Кафедра «Прикладная математика и программирование»

Направление \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

УТВЕРЖДАЮ

Заведующий кафедрой ПМиП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А.Замышляева

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**ЗАДАНИЕ**

**на курсовую работу студента**

\_\_\_\_\_\_\_\_*Постаушкин Ф.Ю.*\_\_\_\_\_\_\_\_\_

Группа \_\_*ЕТ-212*\_\_\_\_

1. Дисциплина  *Объектно-ориентированное программирование*

2. Тема работы \_\_\_\_*Текстовый редактор*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Срок сдачи студентом законченной работы *20 января 2022 г.*

4. Перечень вопросов, подлежащих разработке

1. разработка иерархии и интерфейса классов;
2. реализация программы (библиотеки классов) на языке С++
3. оформление программной документации (описание программы (библиотеки классов), руководство пользователя, листинг кода) и отчета по курсовой работе
4. презентация проектных решений для защиты КР (иерархия и интерфейсы классов, особенности реализации)

5. Календарный план

|  |  |  |
| --- | --- | --- |
| **Наименование разделов**  **(этапов) курсовой работы** | **Срок выполнения**  **разделов (этапов) работы** | **Отметка**  **о выполнении**  **руководителя** |
| анализ предметной области | 01.09.2021-10.10.2021 |  |
| разработка иерархии и интерфейса классов | 20.09.2021-07.11.2021 |  |
| реализация основных классов, функций | 01.10.2021-20.11.2021 |  |
| тестирование программы и/или классов, улучшение и исправление ошибок | 20.10.2021-10.12.2021 |  |
| оформление программной документации и отчета по курсовой работе | 30.10.2021-20.12.2021 |  |
| защита курсовой работы | 20.12.2021-28.12.2021 |  |

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (расшифровка)

АННОТАЦИЯ

Постаушкин Ф.Ю. Текстовый редактор. – Челябинск: ЮУрГУ, ЕТ-212, 2022. – 23с., библиографический список – 2 наим., 1 прил.

В курсовой работе описывается разработка библиотеки класса-редактора текста с возможностями Nano (Linux) / редактора в Far manager (Windows) с помощью объектно-ориентированного подхода. Работа содержит результаты объектно-ориентированного анализа и проектирования, инструкции по установке и использованию библиотеки.

В результате работы был разработан класс-редактор для редактирования текста, код которой приводится в приложении.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 5](#_Toc93183544)

[1 Постановка задачи 6](#_Toc93183545)

[2 Описание программы 6](#_Toc93183546)

[3 Инструкция по установке и требования к системе 8](#_Toc93183547)

[4 Руководство пользователя 8](#_Toc93183548)

[ЗАКЛЮЧЕНИЕ 10](#_Toc93183549)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 11](#_Toc93183550)

[ПРИЛОЖЕНИЕ А 12](#_Toc93183551)

# ВВЕДЕНИЕ

**Актуальность темы**. Объектно-ориентированный подход является наиболее прогрессивной технологией разработки программных систем, позволяет разрабатывать более сложные системы.

**Цель работы** – разработать библиотеку класса редактора для редактирования текста.

**Задачи работы**:

– изучить приемы объектно-ориентированного анализа;

– научиться разрабатывать программы в объектно-ориентированном стиле;

– овладеть технологиями объектно-ориентированного анализа и проектирования [1];

– изучить концепции объектно-ориентированного программирования; изучить особенности объектной модели языка программирования C++ [2]

– научиться самостоятельно и творчески использовать знания и полученные практические навыки;

– овладеть навыками самостоятельного получения новых знаний по теории и практике объектного подхода в программировании.

**Объект работы** – текстовый файл формата TXT

**Предмет работы** – применение объектно-ориентированного подхода для разработки класса.

**Результаты работы** можно использовать в процессе последующего обучения в соответствии с учебным планом подготовки бакалавров по направлению «Прикладная математика и информатика»

## 1 Постановка задачи

Необходимо разработать класс-редактор текста с возможностями Nano (Linux) / редактора в Far manager (Windows) (без использования меню, все управление с клавиатуры).

Сам редактор текста представляет собой упрощенный консольный файловый редактор. Файл выводиться на экран и может быть открыт для чтения / для редактирования. Редактирование и перемещение по файлу производиться через клавиатуру.

Необходимые операции:

* перемещение по тексту;
* режим вставки и замены;
* удаление и вставка символов и строк;
* выделение строк;
* добавление обработчика дополнительный клавиш (функциональные клавиши, Ctrl-, Alt- и т.д.).

В процессе анализа предметной области я выявил два объекта этого класса: объект для редактирования текста из файла и объект для просмотра текста «помощи». К первому объекту можно применять все действия, но только если разрешено редактирования, иначе только перемещение по файлу. А по второму объекту можно только перемещаться, так как он всегда открыт в режиме просмотра.

## 2 Описание программы

2.1 Для разработки программы были использованы:

– компилятор MinGW GNU C/C++ 7.2

2.2 Класс состоит из 1 модуля:

Модуль **text\_editor** (интерфейсная часть в файле .hpp, реализация в файле text\_editor.cpp) содержит следующие классы:

class Editor {

vector <string> text; // вектор строк хранящий текст

vector <string> highlighted\_lines; // буфер для выделеных строк

vector <int> fill\_text; // вектор какие строки должны быть выделены на экране

int begin; // номер верхней строки экрана

int start; // номер крайнего символа строки экрана

int line; // текущая строка - начиная с 0

int symbol; // текущий символ - начиная с 0

int x1, y1, x2, y2, color\_text, color\_fill; // параметры окна

bool view = true; // режим просмотра/редактирования

void (\*fun)(int) = nullptr; //указатель на функцию

public:

Editor(int x\_1, int y\_1, int x\_2, int y\_2, int ct = 0x7, int cf = 0x70); // конструктор x1, y1, x2, y2, цвет текста = 0x7, цвет выделения=0x70

void edit(); // выполнять редактирование

bool isView() const; // проверка на редактирование

void setView(bool t); // получение разрешения на редактирование

int count\_line(); // количество строк

int number\_line() const; // номер текущей строки - нумерация с 0

int number\_symbol() const; // номер текущего символа в строке

void move(int r, int c = 0); // перейти на строку, символ (r, c=0)

string print\_line(int r = -1) const; // получить строку (r=-1) - string, r - номер строки, если -1, то текущую

string print\_highlighted\_line(int r) const; // получить выделенную строку из буфера

void replace\_line(string s, int r = -1); // заменить строку (string, r=-1)

void remove\_line(int r = -1); // удалить строку (r=-1)

void add\_line(string s, int r = -1); // добавить строку (string, r=-1)

void add\_symbol(string s); // добавить символы (string) в текущее местоположение курсора

void remove\_symbol(int c = 1); // удалить символы (c=1) начиная с текущего

void fill\_lines(int n1, int n2); // выделить строки (n1,n2) c n1-й по n2-ю, если оба аргумента = -1, убрать выделение

void choose\_button(void (\*f)(int)); // установить обработчик нераспознанных клавиш (void (\*f)(int))

void update\_screen(); // обновление экрана

};

2.3 Пояснения по алгоритму и особенностям реализации

Класс считывает текст из файла во внутренний вектор строк. Определены методы класса тря редактирования текста.

Метод edit – основной метод класса, в нем происходит редактирование, считывание клавиш, перемещение по тексту. Методы isView и setView проверяют и изменяют режим просмотра/редактирования текста. Метод update\_screen выводит на экран текст и обновляет экран после нажатия клавиш. choose\_button используется для обработки сторонних клавиш, не участвующих в вводе текста. Остальные методы используются для взаимодействия со строками текста.

Из интересных реализаций я выделяю универсальный метод вывода на экран для перемещения по тексту любой длины и ширины. Для это используется номер верхней строки и номер крайнего левого символа.

2.4 Используемые внешние файлы

Класс может в процессе работы сохранять и загружать файлы в формате TXT.

## 3 Инструкция по установке и требования к системе

Для сборки библиотеки необходим компилятор, поддерживающий  
С++14.

Установка и настройки не требуются. Для работы необходимо добавить файлы text\_editor.hpp, text\_editor.cpp в проект.

## 4 Руководство пользователя

После установки нужно создать экземпляр класса, название у него может быть любое. Экземпляр класса может быть открыт в двух режимах: просмотра или редактирования.

Пример:

Editor text(2, 2, 119, 26, 0x1F, 0xF0);

Чтобы считать текст из файла нужно воспользоваться функцией open\_file. Она принимает на вход два аргумента: экземпляр класса и название файла.

Пример:

open\_file(&text, “text.txt”);

После этого вызывается функция edit, в которой и происходит редактирование файла.

isView и setView работаю в паре. Первый возвращает статус просмотра, а второй устанавливает. Если файл открывается в режиме просмотра, то по нему можно перемещаться при помощи стрелок и других клавиш перемещения.

Пример:

text.setView(false);

Для просмотра внутренних полей класса используется count\_line – возвращает количество строк текста, number\_line – возвращает номер текущей строки, number\_symbol – возвращает текущий символ, print\_line – возвращает строку по номеру, print\_highlighted\_line – возвращает выделенную строку по номеру.

Метод move перемещает курсор в определенную строку и символ текста.

Пример:

text.move(10, 10);

Для замены строчки в тексте используется метод replace\_line. Первый аргумент строка, второй аргумент номер строки, которую хотим заменить. Если второй аргумент не указывать, то он заменит строку, в которой находиться курсор.

Пример:

text.replace\_line(“Text”, 10);

Для удаления строчки в тексте используется метод remove\_line. Если второй аргумент не указывать, то он заменит строку, в которой находиться курсор.

Пример:

text.remove\_line(10);

Для добавление строки в текст используется метод add\_line. Первый аргумент строка, второй аргумент номер строки, которую хотим заменить. Если второй аргумент не указывать, то он заменит строку, в которой находиться курсор.

Пример:

text.add\_line(“Text”, 10);

Чтобы добавить символ или символы в текущую строку используется метод add\_symbol.

Пример:

text.add\_symbol(“a”);

А для удаления символов в тексте используется метод remove\_symbol. Удаление происходит начиная с текущего символа.

Пример:

text.remove\_symbol(5);

fill\_lines – выделяет строки и добавляет их в отдельный буфер для копирования. Получает два аргумента – с какой по какую строку выделить.

Пример:

text.fill\_lines(5, 10);

Чтобы подключить отдельный обработчик клавиш имеется метод choose\_button, который получает указатель на отдельную функцию.

update\_screen – обновляет экран, выводя на экран текст в заданной области экрана.

Для редактирования текста нужно вести в командной строке название файла, потом edit или view для нужного режима отображения текста и в конце название файла.

Пример:

\_\_all\_\_.exe edit text.txt

После открытия программы текст будет отображаться на экране. Перемещение по тексту происходит при помощи стрелок. Ввод символов осуществляется с клавиатуры. Можно использовать клавиши ENTER, SPACE, BACKSPACE и т.д.

Клавиши F1-F10 определяются вне класса и на них можно назначить любые действия, но предусмотрены стандартные операции. На пример F1 вызывает помощь, F2 – сохраняет файл, F10 закрыть программу.

Пример:

void special\_key(int k) {

if(k == KEY\_F2 + KEY\_SPECIAL)

save\_file(text, file);

}

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были выявлены объекты предметной области и определена система классов для них. После объектно-ориентированного проектирования классы были реализованы на языке С++. Разработанный код был проверен на контрольных тестах и в код были внесены необходимые исправления. Для библиотеки была разработана документация, описывающая её установку и использование. Таким образом, цель работы была достигнута, задачи – решены.

Результаты работы можно использовать в процессе последующего обучения в форме навыков практического применения объектно-ориентированного подхода для разработки сложных программных систем, понимания порядка этапов разработки программного обеспечения и достигаемых на каждом этапе результатов.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Гамма, Э. Приемы объектно ориентированного проектирования. Паттерны проектирования. [Электронный ресурс] / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. — Электрон. дан. — М. : ДМК Пресс, 2007. — 368 с. — Режим доступа: http://e.lanbook.com/book/1220

2 Липман, С. Язык программирования С++. Полное руководство. [Электронный ресурс] / С. Липман, Ж. Лажойе. — Электрон. дан. — М. : ДМК Пресс, 2006. — 1105 с. — Режим доступа: http://e.lanbook.com/book/1216

# ПРИЛОЖЕНИЕ А

### А.1 Файл text\_editor.hpp

#ifndef TEXT\_EDITOR\_H

#define TEXT\_EDITOR\_H

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <coniow.h>

using namespace std;

class Editor {

vector <string> text; // вектор строк хранящий текст

vector <string> highlighted\_lines; // буфер для выделеных строк

vector <int> fill\_text; // вектор какие строки должны быть выделены на экране

int begin; // номер верхней строки экрана

int start; // номер крайнего символа строки экрана

int line; // текущая строка - начиная с 0

int symbol; // текущий символ - начиная с 0

int x1, y1, x2, y2, color\_text, color\_fill; // параметры окна

bool view = true; // режим просмотра/редактирования

void (\*fun)(int) = nullptr; //указатель на функцию

public:

Editor(int x\_1, int y\_1, int x\_2, int y\_2, int ct = 0x7, int cf = 0x70); // конструктор x1, y1, x2, y2, цвет текста = 0x7, цвет выделения=0x70

void edit(); // выполнять редактирование

bool isView() const; // проверка на редактирование

void setView(bool t); // получение разрешения на редактирование

int count\_line(); // количество строк

int number\_line() const; // номер текущей строки - нумерация с 0

int number\_symbol() const; // номер текущего символа в строке

void move(int r, int c = 0); // перейти на строку, символ (r, c=0)

string print\_line(int r = -1) const; // получить строку (r=-1) - string, r - номер строки, если -1, то текущую

string print\_highlighted\_line(int r) const; // получить выделенную строку из буфера

void replace\_line(string s, int r = -1); // заменить строку (string, r=-1)

void remove\_line(int r = -1); // удалить строку (r=-1)

void add\_line(string s, int r = -1); // добавить строку (string, r=-1)

void add\_symbol(string s); // добавить символы (string) в текущее местоположение курсора

void remove\_symbol(int c = 1); // удалить символы (c=1) начиная с текущего

void fill\_lines(int n1, int n2); // выделить строки (n1,n2) c n1-й по n2-ю, если оба аргумента = -1, убрать выделение

void choose\_button(void (\*f)(int)); // установить обработчик нераспознанных клавиш (void (\*f)(int))

void update\_screen(); // обновление экрана

};

void draw\_frame(string s); // рисование рамки окна

void special\_key(int k); // обработка специльных клавиш

void open\_file(Editor \*t, string f); // считывание из файла

void save\_file(Editor t, string f); // сохранение в файл

#endif

### А.2 Файл text\_editor.cpp

#include "text\_editor.hpp"

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <coniow.h>

using namespace std;

Editor::Editor(int x\_1, int y\_1, int x\_2, int y\_2, int ct, int cf) { // конструктор x1, y1, x2, y2, цвет текста = 0x7, цвет выделения=0x70

begin = 1; // верхняя строка экрана

start = 1; // крайний символ строки

line = 0; // текущая строка

symbol = 1; // текущий символ

x1 = x\_1; // координаты отриски окна

y1 = y\_1;

x2 = x\_2;

y2 = y\_2;

color\_text = ct; // цвет текста

color\_fill = cf; // цвет выделения

}

void Editor::edit() { // выполнять редактирование

int k; // номер клавиши

string s; // техническая трока

char c; // технический символ

if (text.size() > y2 - y1 + 1) // расчитывание области отриски экрана

begin = text.size() - y2 + y1;

line = count\_line(); // устанавилавет курсор в конец текста

if (text[line - 1].size() > x2 - x1 + 1) // расчитывание области отриски экрана

start = text[line - 1].size() - x2 + x1 + 1;

symbol = text[line - 1].size() + 1; // устанавливает курсор в конец строки

while (1) { // основной цикл редактирования

if (symbol > text[line - 1].size() + 1) // ограничения для курсора на экране

symbol = text[line - 1].size() + 1;

if (symbol < start)

start = text[line - 1].size() - x2 + x1 + 1;

if (start < 1)

start = 1;

update\_screen(); // обновление экрана

gotoxy(symbol - start + 1, line - begin + 1); // перемещение курсора

s = "";

k = getkbm();

c = (char) k;

s += c;

if (32 <= k && k <= 255) { // клавиши текста

if (!isView()) {

if (symbol == start + x2 - x1)

start++;

add\_symbol(s);

}

continue;

}

switch (k) {

case KEY\_LEFT + KEY\_SPECIAL: // клавиша "ВЛЕВО"

if (symbol > start) // перемещение курсора влево

symbol--;

else if (start > 1) {

start--;

symbol--;

}

break;

case KEY\_UP + KEY\_SPECIAL: // клавиша "ВВЕРХ"

if (line != begin) // перемещение курсора вверх

line--;

else if (begin > 1) {

begin--;

line--;

}

break;

case KEY\_RIGHT + KEY\_SPECIAL: // клавиша "ВПРАВО"

if (symbol == start + x2 - x1) // перемещение курсора вправо

start++;

symbol++;

break;

case KEY\_DOWN + KEY\_SPECIAL: // клавиша "ВНИЗ"

if (line != begin + y2 - y1 && line != count\_line()) // перемещение курсора вниз

line++;

else if (line == begin + y2 - y1 && line != count\_line()) {

begin++;

line++;

}

break;

case KEY\_BACKSPACE: // клавиша "BACKSPACE"

if (!isView()) { // флаг для редактирования

if (symbol > 1) { // если символы в строке есть удаляет символ

symbol--;

remove\_symbol();

}

else if (line != 1) { // если символы закончились - копирует строку и вставляет ее в строку ввыше

string temp = "";

if (text[line - 1].size() > 0) {

for (int i = symbol - 1; i < text[line - 1].size(); i++)

temp += text[line - 1][i];

}

if (line == begin)

begin--;

remove\_line(line - 1); // удаление строки

symbol = text[line - 1].size() + 1;

add\_symbol(temp); // добавление строки ниже

symbol = text[line - 1].size() + 1;

}

}

break;

case KEY\_DELETE + KEY\_SPECIAL: // клавиша "DELETE"

if (!isView())

if (text[line - 1].size() != 0)

remove\_symbol();

break;

case KEY\_PGUP + KEY\_SPECIAL: // клавиша "PAGEUP"

begin = 1;

start = 1;

line = 1;

symbol = 1;

break;

case KEY\_PGDN + KEY\_SPECIAL: // клавиша "PAGEDWN"

begin = text.size() - y2 + y1; // обратока границ экрана

if (begin < 1)

begin = 1;

line = text.size();

symbol = text[line - 1].size() + 1;

if (text[line - 1].size() > x2 - x1 + 1)

start = text[line - 1].size() - x2 + x1 + 1;

break;

case KEY\_HOME + KEY\_SPECIAL: // клавиша "HOME"

start = 1;

symbol = 1;

break;

case KEY\_END + KEY\_SPECIAL: // клавиша "END"

symbol = text[line - 1].size();

break;

case KEY\_ENTER: // клавиша "ENTER"

if (!isView()) {

if (symbol <= text[line - 1].size()) { // если курсор находиться где-то в строке переносяться символы правее курсора

string temp = "";

for (int i = symbol - 1; i < text[line - 1].size(); i++) {

temp += text[line - 1][i];

}

text[line - 1].erase(symbol - 1, text[line - 1].size() - symbol + 1);

if (line == begin + y2 - y1)

begin++;

add\_line(temp, line);

}

else { // если символ находиться в конце строки создается пустая строка

if (line == begin + y2 - y1)

begin++;

add\_line("");

symbol = 1;

}

}

break;

default:

if (fun != nullptr) // обработка специальных клавиш

fun(k);

}

};

}

bool Editor::isView() const { // проверка на редактирование

return view;

}

void Editor::setView(bool t) { // получение разрешения на редактирование

view = t;

}

int Editor::count\_line() { // количество строк

return text.size();

}

int Editor::number\_line() const { // номер текущей строки - нумерация с 0

return line;

}

int Editor::number\_symbol() const { // номер текущего символа в строке

return symbol;

}

void Editor::move(int r, int c) { // перейти на строку, символ (r, c=0)

if (r > y2 - y1) // ограничение перемещения курсора

start = r - y2 + y1;

line = r;

if (text[line - 1].size() + 1< c)

symbol = text[line - 1].size() + 1;

else

symbol = c;

if (symbol > x2 - x1)

start = symbol - x2 + x1;

}

string Editor::print\_line(int r) const { // получить строку (r=-1) - string, r - номер строки, если -1, то текущую

if (r == -1)

return text[line - 1];

return text[r];

}

string Editor::print\_highlighted\_line(int r) const { // получить выделенную строку из буфера

return highlighted\_lines[r];

}

void Editor::replace\_line(string s, int r) { // заменить строку (string, r=-1)

if (r != -1)

text[r] = s;

else

text[line - 1] = s;

}

void Editor::remove\_line(int r) { // удалить строку (r=-1)

vector<string>::iterator it;

if (r != -1) {

it = text.begin() + r;

}

else {

it = text.begin() + line - 1;

r = line - 1;

}

text.erase(it);

line--;

}

void Editor::add\_line(string s, int r) { // добавить строку (string, r=-1)

if (r == -1) {

r = line;

}

vector<string>::iterator it = text.begin()+r;

text.insert(it, s);

if (r <= line)

line++;

fill\_text.resize(line);

}

void Editor::add\_symbol(string s) { // добавить символы (string) в текущее местоположение курсора

text[line - 1].insert(symbol - 1, s);

symbol++;

}

void Editor::remove\_symbol(int c) { // удалить символы (c=1) начиная с текущего

text[line - 1].erase(symbol - 1, c);

}

void Editor::fill\_lines(int n1, int n2) { // выделить строки (n1,n2) c n1-й по n2-ю, если оба аргумента = -1, убрать выделение

if (n1 == -1 && n2 == -1) { // очистить буфер выделенных строк и выленных строк

fill\_text.clear();

fill\_text.resize(count\_line());

highlighted\_lines.clear();

}

else

for (int i = n1; i <= n2; i++) { // выделение строк с n1-строки по n2-строку

fill\_text[i - 1] = 1;

highlighted\_lines.push\_back(print\_line(i - 1));

}

}

void Editor::choose\_button(void (\*f)(int)) { // установить обработчик нераспознанных клавиш (void (\*f)(int))

fun = f;

}

void Editor::update\_screen() { // обработка экрана

window(x1, y1, x2, y2); // огрнаичение области экрана

clrscr(); // очистка экрана

for (int i = begin - 1; i < y2 - y1 + begin; i++) { // вывод на экран текста

for (int j = start - 1; j < text[i].size(); j++) {

if (j > x2 - y1 + start - 1) continue;

gotoxy(j - start + 2, i - begin + 2);

if (fill\_text[i] == 1) // выделяет строку

textattr(color\_fill);

else // отрисовывает обычным цветом

textattr(color\_text);

putch(text[i][j]);

}

}

}

void draw\_frame(string s) { // отрисовка рамки

clrscr();

textattr(BLUE \* 16 + WHITE);

for (int i = 1; i < 27; i++) {

for (int j = 1; j < 121; j++) {

gotoxy(j, i);

putwch(32);

}

}

for (int i = 2; i < 120; i++) {

gotoxy(i, 1);

putwch(0x2550);

}

for (int i = 2; i < 27; i++) {

gotoxy(1, i);

putwch(0x2551);

}

for (int i = 2; i < 120; i++) {

gotoxy(i, 27);

putwch(0x2550);

}

for (int i = 2; i < 27; i++) {

gotoxy(120, i);

putwch(0x2551);

}

gotoxy(1, 1);

putwch(0x2554);

gotoxy(120, 1);

putwch(0x2557);

gotoxy(1, 27);

putwch(0x255A);

gotoxy(120, 27);

putwch(0x255D);

textattr(BLUE \* 16 + WHITE);

gotoxy(2, 1);

cputs(s.c\_str());

textattr(DARKGRAY \* 16 + WHITE);

gotoxy(2, 29);

string line;

ifstream input("bar.txt");

if (!input.is\_open()) exit(1);

while (getline(input, line)) {

input >> line;

cputs(line.c\_str());

}

input.close();

}

### А.3 Файл text\_editor.cpp

#include <text\_editor.hpp>

#include <cstring>

#include <iostream>

using namespace std;

Editor text(2, 2, 119, 26, 0x1F, 0xF0); // текст для редактирования

Editor help(5, 3, 116, 25, 0x5F); // файл помощи с кратким описанием программы

string file; // имя файла

int l1 = -1, l2 = -1; // первая и вторая строка выделения

int F5flag = -1; // флаг для проверки нажатия клафиши F5

int F6flag = -1; // флаг для проверки нажатия клафиши F6

int main(int argc, char \*argv[]) {

setlocale(LC\_ALL,""); // русский язык для консоли

file = argv[2]; // сохраняет имя файла из командной строки

// получает функцию для обработки специальных клавиш

text.choose\_button(special\_key); // получает функцию для обработки специальных клавиш

// разрешение на редактирование/просмотр

if (strcmp(argv[1], "edit") == 0)

text.setView(false);

//чтение из файла

open\_file(&text, file);

open\_file(&help, "help.txt");

// отрисовка окна

draw\_frame(file);

// редактирвоание файла

text.edit();

return 0;

}

void special\_key(int k) { // функция для обратоки специальных клавиш

switch (k) {

case KEY\_F1 + KEY\_SPECIAL: // вывод помощи

help.update\_screen(); // вывод на экран помощи

getch();

break;

case KEY\_F2 + KEY\_SPECIAL: // сохранение в файл

save\_file(text, file); // сохранить в файл

break;

case KEY\_F3 + KEY\_SPECIAL: // начать выделение

l1 = text.number\_line(); // получить номер первой строки

break;

case KEY\_F4 + KEY\_SPECIAL: // закончить выделение

l2 = text.number\_line(); // получить номер второй строки

break;

case KEY\_F5 + KEY\_SPECIAL: // выделить и скопировать текст

if (F5flag == -1 && l1 != -1 && l2 != -1) { // проверка на первое нажатие

if (l1 > l2) { // меняет местами номера строк, если введены были снизу вверх

int t = l2;

l2 = l1;

l1 = t;

}

F5flag \*= -1; // изменение флага

}

else {

for (int i = 0; i <= l2 - l1; i++) {

text.add\_line(text.print\_highlighted\_line(i)); // вставить текст из буфера

}

l1 = -1; // убрать выделение

l2 = -1;

F5flag \*= -1; // изменение флага

}

text.fill\_lines(l1, l2); // выделяет или уберает выделение

break;

case KEY\_F6 + KEY\_SPECIAL: // выделить и скопировать текст

if (F6flag == -1 && l1 != -1 && l2 != -1) {

if (l1 > l2) { // меняет местами номера строк, если введены были снизу вверх

int t = l2;

l2 = l1;

l1 = t;

}

F6flag \*= -1; // изменение флага

}

else {

for (int i = l1; i <= l2; i++)

text.remove\_line(l1 - 1); // удаление строк

l1 = -1;

l2 = -1;

F6flag \*= -1; // изменение флага

}

text.fill\_lines(l1, l2); // выделяет или уберает выделение

break;

case KEY\_F10 + KEY\_SPECIAL: // закрыть программу

save\_file(text, file); // сохранить в файл

exit(1);

break;

}

}

void open\_file(Editor \*t, string f) { // открыть файл и записать в файл

string line;

ifstream input(f);

if (!input.is\_open()) exit(1);

while (getline(input, line)) {

t->add\_line(line);

}

input.close();

}

void save\_file(Editor t, string f) { // сохранить в файл из класса

ofstream output(f);

for (int i = 0; i < t.count\_line(); i++) {

output << t.print\_line(i) << endl;

}

output.close();

}