|  |  |
| --- | --- |
| https://studfiles.net/html/2706/219/html_4NnFGVyFmL.LWVf/img-KuTuVC.png | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| БГТУ.СМК-Ф-4.2-К5-01 |

Факультет И Информационные и управляющие системы

шифр наименование

Кафедра И5 Информационные системы и программные технологии

шифр наименование

Дисциплина Программирование на языке высокого уровня

КУРСОВАЯ РАБОТА

|  |
| --- |
| Создание базового класса строка, строки идентификатора |
| и двоичной строки. |
| Создание графического интерфейса. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | | | | | | *И-582* |
| *Махнев П.С.* | | | | | | |
| Фамилия И.О. | | | | | | |
|  | | | | | | |
| **РУКОВОДИТЕЛЬ** | | | | | | |
|  | | | | | | |
| *Спирин Д.О.* | | | |  |  | |
| Фамилия И.О. | | | |  | Подпись | |
|  | | | | | | |
| Оценка | | |  | | |  |
| « |  | » |  | | | 2019 г. |

САНКТ-ПЕТЕРБУРГ

2019 г.

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc11252376)

[Введение 3](#_Toc11252377)

[Техническое задание 4](#_Toc11252378)

[1. Класс Строка 8](#_Toc11252379)

[1.1. Базовый класс Строка 8](#_Toc11252380)

[1.2. Производный класс. Строка идентификатор 9](#_Toc11252381)

[1.3. Производный класс. Двоичная строка 10](#_Toc11252382)

[2. Класс Элемент управления 11](#_Toc11252383)

[2.1. Базовый класс Control 11](#_Toc11252384)

[2.2. Производный класс Button 13](#_Toc11252385)

[2.3. Производный клаcc Input 14](#_Toc11252386)

[2.4. Производный класс ItemList 15](#_Toc11252387)

[2.5. Производный класс DropDownList 17](#_Toc11252388)

[3. Текст программы 20](#_Toc11252389)

[3.1. Файл main.cpp 20](#_Toc11252390)

[3.2. Файл Interface.h 20](#_Toc11252391)

[3.3. Файл Interface.cpp 21](#_Toc11252392)

[3.4. Файл Interface\_OnEvent.cpp 23](#_Toc11252393)

[3.5. Файл Interface\_OnEvent\_Function.cpp 24](#_Toc11252394)

[3.6. Файл Interface\_OnExecute.cpp 40](#_Toc11252395)

[3.7. Файл color.h 44](#_Toc11252396)

[3.8. Файл String.h 45](#_Toc11252397)

[3.9. Файл String.cpp 45](#_Toc11252398)

# Введение

Целью данной работы является создание класса «строка» и производных классов от него, а также разработка интерфейса для ввода начальных значений, отображения результатов и тестирования методов классов.

Для достижения поставленной цели были осуществлены следующие задачи:

1. Разработка классов: базовая строка, строка идентификатор, двоичная (бинарная) строка согласно техническому заданию.
2. Разработка класса элемента управления. Данный класс является базовым для всех возможных управляющих элементов. От данного класса будут наследоваться классы таких элементов управления, как Button (кнопка), Input (поле ввода), Label (надпись), ListBox (список с выбором элемента из списка), DropDownList (список с собственным значением) и TextField (текстовое поле).
3. Разработка класса интерфейса. Данный класс является связующим между классами строк и элементами управления, который позволит конечному пользователю взаимодействовать с методами выбранного класса. Данный класс будет реализовывать отслеживание событий, первоначальную инициализацию и отображение управляющих элементов интерфейса.
4. Разработка цветовой схемы для интерфейса.

# Техническое задание

Описать три класса: базовый класс «Строка» и производные от него класс «Строка-идентификатор» и класс «Двоичная строка». Введем для ясности следующие обозначения:

1. Базовый класс строки обозначется, как \_строка\_
2. Производный класс идентификатора обозначется, как \_строка\_идентификатор\_
3. Производный класс двоичной строки обозначется, как \_двоичная\_строка\_
4. Базовый класс \_строка\_
   1. Обязательные поля класса:
      1. Указатель на char — хранит адрес динамически выделенной памяти для размещения символов строки;
      2. Значение типа unsigned int — хранит длину строки в байтах.
   2. Обязательные методы класса:
      1. Конструктор без параметров;
      2. Конструктор, принимающий в качестве параметра Си-строку (заканчивается нулевым байтом);
      3. Конструктор, принимающий в качестве параметра символ (char);
      4. Конструктор копирования;
      5. Деструктор;
      6. Функция получения длины.
   3. Переопределенные операции:
      1. Операция «=», присваивание одного объекта класса другому.
5. Производный класс \_строка\_идентификатор\_

Строки данного класса строятся по правилам записи идентификаторов в СИ, и могут включать в себя только те символы, которые могут входить в состав Си-идентификаторов[[1]](#footnote-1). Если исходные данные противоречат правилам записи идентификатора, то создается пустая \_строка\_идентификатор\_.

* 1. Обязательные методы класса:
     1. Конструктор без параметров;
     2. Конструктор, принимающий в качестве параметра Си-строку (заканчивается нулевым байтом);
     3. Конструктор копирования;
     4. Деструктор;
     5. Поиск первого вхождения символа в строку;
     6. Перевод всех буквенных символов строки в нижний регистр;
  2. Переопределенные операции:
     1. Операция вычитания «-» (из первого операнда удаляются все символы, входящие во второй операнд).

1. Производный класс \_двоичная\_строка\_

Строки данного класса могут содержать только символы '0' или '1'. Если в составе инициализирующей строки будут встречены любые символы, отличные от допустимых, \_двоичная\_строка\_ принимает нулевое значение. Содержимое данных строк рассматривается как двоичное число. Отрицательные числа хранятся в дополнительном коде.

* 1. Обязательные методы класса:
     1. Конструктор без параметров;
     2. Конструктор, принимающий в качестве параметра Си-строку (заканчивается нулевым байтом);
     3. Конструктор копирования;
     4. Деструктор;
     5. Инвертирование числа.
  2. Переопределенные операции:
     1. Операция вычитания «-» (длина строки результата равна длине большей из строк, в случае необходимости, более короткая битовая строка расширяется влево знаковым разрядом).

**Задание:**

Разработать иерархию классов по следующей схеме:

с обязательной поддержкой заданных членов и методов.

Написать тестовую программу, которая:

1. динамически выделяет массив указателей на базовый класс (4-6 шт.);
2. в режиме диалога заполняет этот массив указателями на производные классы, при этом экземпляры производных классов создаются динамически с заданием начальных значений;
3. для созданных экземпляров производных классов выполняется проверка всех разработанных методов (в соответствии с вариантом задания), с выводом исходных данных и результатов на дисплей.

Основные пункты:

1. «Инициализация».

Подпункты:

* 1. "Число элементов" — задает число элементов в массиве указателей на базовый класс. Запрещается после задания числа элементов пользоваться этим пунктом меню.
  2. "Начальное значение" — с помощью этого пункта меню можно задать номер элемента, его тип и начальное значение.

1. «Тестирование»

В качестве подпунктов указываются типы строк из варианта задания и

обязательные для всех подпункты "строка", например,

2.1. «Строка»;

2.2. «Строка-идентификатор»;

2.3. «Двоичная строка».

После выбора одного из пунктов меню предлагается выбрать один из методов списка всех обязательных методов (кроме конструкторов и деструкторов), связанных с выбранным подпунктом. Результат выводится на экран.

1. «Выход»

Значения экземпляров классов по мере их инициализации выводятся на дисплей и остаются на все время выполнения программы.

Задавать начальные значения и работать с другими пунктами меню запрещается до тех пор, пока не будет задано число элементов. Допускается задать новое начальное значение несколько раз.

# Класс Строка

## 1.1. Базовый класс Строка

Название класса: String. Является базовым классов для классов \_двоичная\_строка\_ и \_строка\_идентификатор\_.

Данные-члены класса, объявленные как protected:

1. char\* str — указатель на массив символов;
2. unsigned int length — переменная, хранящая длину строки.

Методы класса \_строка\_:

1. String() — конструктор без параметров, создает строку нулевой длины;
2. String(const char\*) — конструктор, в качестве параметра принимающий Си-строку.;
3. String(const char) — конструктор, принимающий в качестве параметра один символ;
4. String(const String&) — конструктор копирования;
5. ~String() — деструктор;
6. unsigned int getLength() — функция получения длины;
7. char\* getString() — дополнительная функция, для получения строки в виде массива char;
8. void operator= (String& obj) — перегрузка операции «=», для присваивания одного объекта класса другому.

## 1.2. Производный класс. Строка идентификатор

Название класса: ID\_String. Данный класс является производным классом класса String.

Методы класса ID\_String:

1. ID\_String() — конструктор;
2. ID\_String(const char\*) — конструктор, принимающий в качестве параметра си-строку.
3. ID\_String(const Identifier&) — конструктор копирования;
4. ~ID\_String() — деструктор;
5. bool isCorrect(char\* str) — функция для проверки корректности строки;
6. void toLowerCase() — функция перевода всех буквенных символов в нижний регистр;
7. int indexOf(char symbol) — функция для нахождения первого вхождения символа в строке;
8. ID\_String& operator- (const ID\_String& obj) — переопределение операции «-».

## 1.3. Производный класс. Двоичная строка

Название класса: BIN\_String. Данный класс является производным классом класса String.

Методы класса BIN\_String:

1. BIN\_String() — конструктор;
2. BIN\_String(const char\*) — конструктор, принимающий в качестве параметра си-строку.
3. BIN\_String(const Identifier&) — конструктор копирования;
4. ~BIN\_String() — деструктор;
5. bool isCorrect(char\* str) — функция для проверки корректности строки;
6. void invert() — функция инвертирующая двоичную строку;
7. BIN\_String& operator- (BIN\_String& obj) — перегрузка операции «-» для вычитания одной строки из другой, как двух двоичных чисел из другого.

# 2. Класс Элемент управления

## 2.1. Базовый класс Control

Название класса: Control. Является базовым классов для классов Button, Input, ItemList, DropDownList, TextField. Так же косвенно влияет на класс Label.

Данные-члены класса, объявленные как protected:

1. SDL\_Rect\* sizes — указатель на структуру, в которой хранятся данные о размере и метоположении объекта, поля структуры:
   1. int x — положение по x;
   2. int y — положение по y;
   3. int w — ширина;
   4. int h — высота.
2. SDL\_Event event — переменная хранящая событие SDL;
3. TTF\_Font\* font — переменная хранящая указатель на загруженный шрифт;
4. SDL\_Renderer\* renderer — переменная хранящая указатель на renderer;
5. COLOR Colors — объект структуры COLOR, хранящей в себе цветовую схему проекта;
6. int font\_size — переменная хранящая размер шрифта;
7. bool block — переменная указывающая на то, заблокирован объект или нет;
8. bool display — переменная указывающая на то, отображается объект или нет;
9. bool click — переменная указывающая на то, нажат ли объект или нет.

Методы класса Control:

1. Control(SDL\_Renderer \* renderer, SDL\_Rect\* \_sizes, string \_font, int \_font\_size) — конструктор, принимающий на вход renderer, структуру \_sizes типа SDL\_Rect, путь к загружаемому шрифту \_font и размер шрифта \_font\_size;
2. virtual void render() — виртуальная функция отображения, базовая реализация отсутствует;
3. virtual void renderLabel(string text, SDL\_Rect\* renderRect, int type) — виртуальная функция отображения текста, базовая реализация присутствует. На вход получает текст, который необходимо отобразить, координаты в виде SDL\_Rect и тип выравнивания, определенный следующим перечислением:

enum TYPE\_ALIGN {

CENTERED\_ALIGN,

LEFT\_ALIGN

};

1. void unlock() — функция по результату разблокировывающая объект, по факту устанавливающая переменной block значение false;
2. void blocked() — функция по результату блокирующая объект, по факту устанавливающая переменной block значение true;
3. void hide() — функция по результату скрывающая объект, по факту устанавливающая переменной display значение false;
4. void show() — функция по результату показывающая объект, по факту устанавливающая переменной display значение true;
5. void Click(bool) — функция устанавливающая значение переменной click равное переданному значению типа bool;
6. bool getShow() — функция возвращающая текущее значение переменной display;
7. bool checkHover(int x, int y) — функция проверяющая, является ли точка с координатами x и y, точкой внутри текущего объекта;
8. virtual void onEvent(SDL\_Event\* event) — виртуальная функция для отслеживания событий данного элемента. Базовая реализация отсутствует.

## 2.2. Производный класс Button

Название класса: Button. Является производным классом от класса Control.

Данные-члены класса, объявленные как private:

1. string text — строка, хранящая в себе текст, выводимый на теле кнопки.
2. int type — переменная хранящая в себе тип кнопки, определенный следующим перечислением:

enum BUTTON\_TYPES {

BUTTON\_OK\_COUNT\_ELEMENT,

BUTTON\_OK\_SET\_START\_VALUE,

BUTTON\_OK\_PERFORM\_OPERATION,

BUTTON\_QUIT

};

Методы класса Button:

1. Button(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, string \_text, int \_type, string \_font, int \_font\_size) — конструктор, принимающий на вход renderer, структуру \_sizes типа SDL\_Rect хранящая в себе размеры и положение кнопки, тип кнопки, определенный перечислением BUTTON\_TYPES, путь к загружаемому шрифту \_font и размер шрифта \_font\_size;
2. void render() — функция отображения кнопки;

## 2.3. Производный клаcc Input

Название класса: Input. Является производным классом от класса Control.

Данные-члены класса, объявленные как private:

1. string text — строка, хранящая в себе текст, выводимый на теле кнопки.
2. bool focus — переменная указывающая на то, в фокусе ли данный input или нет;
3. int type — переменная хранящая в себе тип кнопки, определенный следующим перечислением:

enum INPUT\_TYPES {

INPUT\_COUNT\_ELEMENT,

INPUT\_START\_VALUE,

INPUT\_SYMBOL\_INDEX\_OF

};

Методы класса Input:

1. Input(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, string \_text, int \_type, string \_font, int \_font\_size) — конструктор, принимающий на вход renderer, структуру \_sizes типа SDL\_Rect хранящая в себе размеры и положение поля ввода, тип поля ввода, определенный перечислением INPUT\_TYPES, путь к загружаемому шрифту \_font и размер шрифта \_font\_size;
2. void render() — функция отображения поля ввода;
3. void onEvent(SDL\_Event\* event) — функция отслеживания событий конкретного поля ввода;
4. void clear() — функция очистки поля ввода;
5. string getValue() — функция возвращающая строку, введенную в данное поле ввода;
6. void Focus(bool value) — функция устанавливающая переменной focus значение равное переданному значению типа bool;

## 2.4. Производный класс ItemList

Название класса: ItemList. Является производным классом от класса Control.

Данные-члены класса, объявленные как private:

1. vector <Item\*> List — массив хранящий в себе список указателей на элементы списка. Элемент списка определяется структурой Item;
   1. Структура Item:
      1. Поля структуры:
         1. string text — поле, хранящее в себе текст элемента;
         2. bool hover — переменная указывающая на то, наведен ли курсор на объект или нет;
         3. int flag — переменная хранящая в себе флаг, указывающий на то, какие действия делает данный элемент списка. Определяется следующим перечислением

enum ITEM\_LIST\_TYPES {

NULLELE,

TEST\_STRING\_EQUAL,

TEST\_STRING\_GET\_LENGTH,

TEST\_IDENTIFICATION\_OPERATOR\_TO\_LOWER\_CASE,

TEST\_IDENTIFICATION\_OPERATOR\_EQUAL,

TEST\_IDENTIFICATION\_OPERATOR\_SUBTRACTION,

TEST\_IDENTIFICATION\_INDEX\_OF,

TEST\_BINARY\_REVERSE,

TEST\_BINARY\_SUBTRACTION

};

* + - 1. bool blocked — переменная указывающая на то, заблокирован этот элемент списка или нет.
    1. Методы структуры:
       1. Item() — конструктор структуры;
       2. void Hover(bool value) — функция устанавливающая значение переменной hover равное переданному значению типа bool;
       3. bool Hover() — функция возвращающая текущее значение переменной hover;
       4. void Block(bool value) — функция устанавливающая значение переменной blocked равное переданному значению типа bool.

1. SDL\_Rect\* itemSizes — указатель на структуру SDL\_Rect, в которой хранятся данные о размере элемента списка;
2. string mainLabel — переменная хранящая в себе строку, выводимую на тело списка;
3. bool show\_menu — переменная указывающая на то, открыт список или нет.

Методы класса ItemList:

1. void render() — функция отображения списка;
2. void add(string text, int flag) — функция добавления нового элемента списка, принимает строку string, которая будет отображаться и флаг, который будет указывать на действие совешаемое этим элементом;
3. void open(bool value) — функция устанавливающая переменной show\_menu значение равное переданному значению типа bool. По факту, функция открывающая список, или его скрывающая;
4. bool open() — функция возвращающая текущее значение переменной show\_menu;
5. int checkItemHover(int x, int y) — функция проверяющая, является ли переданная точка с координатами x и y точкой внутри какого-либо элемента списка, в случае истины, возвращает флаг этого элемента.

## 2.5. Производный класс DropDownList

Название класса: DropDownList. Является производным классом от класса Control.

Данные-члены класса, объявленные как private:

1. vector < DropDownItem\*> List — массив хранящий в себе список указателей на элементы списка. Элемент списка определяется структурой DropDownItem;
   1. Структура DropDownItem:
      1. Поля структуры:
         1. string text — поле, хранящее в себе текст элемента;
         2. bool hover — переменная указывающая на то, наведен ли курсор на объект или нет;
         3. int flag — переменная хранящая в себе флаг, указывающий на то, какие действия делает данный элемент списка. Определяется перечислением ITEM\_LIST\_TYPES[[2]](#footnote-2), описанным ранее.
         4. bool blocked — переменная указывающая на то, заблокирован этот элемент списка или нет.
      2. Методы структуры:
         1. DropDownItem() — конструктор структуры;
         2. void Hover(bool value) — функция устанавливающая значение переменной hover равное переданному значению типа bool;
         3. bool Hover() — функция возвращающая текущее значение переменной hover;
         4. void Block(bool value) — функция устанавливающая значение переменной blocked равное переданному значению типа bool.
2. SDL\_Rect\* itemSizes — указатель на структуру SDL\_Rect, в которой хранятся данные о размере элемента списка;
3. string mainLabel — переменная хранящая в себе строку, выводимую на тело списка;
4. bool show\_menu — переменная указывающая на то, открыт список или нет.
5. int flag\_select — переменная хранящая в себе флаг, выбранного пользователем элемента списка;
6. int type — переменная хранящая в себе тип выпадающего списка, определенный следующим перечислением:

enum DROP\_ITEM\_LIST\_TYPES {

DROP\_BASE\_CLASS = 10,

DROP\_ID\_CLASS = 11,

DROP\_BIN\_CLASS = 12

};

1. SDL\_Texture\* dropButton — указатель на текстуру SDL хранящую изображение стрелки для списка.

Методы класса ItemList:

1. void render() — функция отображения списка;
2. DropDownItem\* add(string text, int flag) — функция добавления нового элемента списка, принимает строку string, которая будет отображаться, и флаг, который будет указывать на действие совешаемое этим элементом. Возвращает указатель на добавленный элемент, для возможности вызова функций по работе с элементом в той же строке, в которой она и добавляется;
3. void open(bool value) — функция устанавливающая переменной show\_menu значение равное переданному значению типа bool. По факту, функция открывающая список, или его скрывающая;
4. bool open() — функция возвращающая текущее значение переменной show\_menu;
5. int checkItemHover(int x, int y) — функция проверяющая, является ли переданная точка с координатами x и y точкой внутри какого-либо элемента списка, в случае истины, возвращает флаг этого элемента.

# 3. Текст программы

## 3.1. Файл main.cpp

#include "iostream"

#include "app.h"

using namespace std;

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

setlocale(LC\_ALL, "Rus");

Interface newInterface(1200, 500);

return newInterface.onExecute();

}

## 3.2. Файл Interface.h

#include "iostream"

#include "SDL.h"

#include "SDL\_image.h"

#include "SDL\_ttf.h"

#include "vector"

#include "Color.h"

#include "Elements/button\_.h"

#include "Elements/input\_.h"

#include "Elements/itemList\_.h"

#include "Elements/label.h"

#include "Elements/ListDown.h"

#include "Elements/TextField.h"

#include "String/String.h"

#include "String/ID\_String.h"

#include "String/BIN\_String.h"

using namespace std;

enum STRING\_CLASS\_ID {

STRING\_BASE,

STRING\_ID,

STRING\_BIN

};

enum INTERFACE\_STATES {

INTERFACE\_START,

INTERFACE\_ENTER\_COUNT,

INTERFACE\_ENTER\_START\_VALUES,

INTERFACE\_TESTING

};

class Interface {

private:

bool running;

SDL\_Window\* window;

SDL\_Renderer\* renderer;

SDL\_Event event;

SDL\_Texture\* texture;

short screenWidth;

short screenHeight;

int state;

COLOR Colors;

// элементы интерфейса

vector <Button\_\*> Buttons;

vector <Input\_\*> Inputs;

vector <ItemList\_\*> ItemLists;

vector <DropDownList\*> DropDownLists;

vector <Label\*> Labels;

vector <TextField\*> TextFields;

//

// Элементы приложения

bool ready\_fill\_classes;

int count\_ready;

int count\_element;

vector <String\*> ptrBaseClass;

vector <int> type\_ptrBaseClasses;

int nowOperation;

public:

Interface();

Interface(short width, short height);

~Interface();

private:

void render();

void onEvent();

bool init();

void mouseButtonDown(SDL\_Event\* event);

void mouseButtonUp(SDL\_Event\* event);

void mouseMotion(SDL\_Event\* event);

void keyDown(SDL\_Event\* event);

void quit();

public:

int onExecute();

};

## 3.3. Файл Interface.cpp

#include "App.h"

Interface::Interface()

{

state = INTERFACE\_START;

running = true;

window = nullptr;

renderer = nullptr;

texture = nullptr;

screenWidth = 1280;

screenHeight = 720;

count\_ready = 0;

count\_element = 0;

nowOperation = 0;

}

Interface::Interface(short width, short height)

{

state = INTERFACE\_START;

running = true;

window = nullptr;

renderer = nullptr;

texture = nullptr;

screenWidth = width;

screenHeight = height;

count\_ready = 0;

count\_element = 0;

ready\_fill\_classes = false;

nowOperation = 0;

}

bool Interface::init()

{

setlocale(LC\_ALL, "Rus");

if ((SDL\_Init(SDL\_INIT\_EVERYTHING)) < 0) {

cout << "Error of initialize SDL: " << SDL\_GetError() << endl;

return false;

}

if ((window = SDL\_CreateWindow("Test string program", SDL\_WINDOWPOS\_CENTERED, SDL\_WINDOWPOS\_CENTERED, screenWidth, screenHeight, SDL\_WINDOW\_SHOWN)) == nullptr) {

cout << "Error of initialize new Window: " << SDL\_GetError() << endl;

return false;

}

if ((renderer = SDL\_CreateRenderer(window, -1, SDL\_RENDERER\_ACCELERATED)) == nullptr) {

printf("Renderer could not be created! SDL Error: %s\n", SDL\_GetError());

return false;

}

if (!(IMG\_Init(IMG\_INIT\_PNG) & IMG\_INIT\_PNG)) {

printf("SDL\_image could not initialize! SDL\_image Error: %s\n", IMG\_GetError());

return false;

}

if (TTF\_Init() == -1){

printf("SDL\_ttf could not initialize! SDL\_ttf Error: %s\n", TTF\_GetError());

return false;

}

return true;

}

void Interface::quit()

{

SDL\_DestroyTexture(texture);

SDL\_DestroyRenderer(renderer);

SDL\_DestroyWindow(window);

window = nullptr;

renderer = nullptr;

texture = nullptr;

SDL\_Quit();

running = false;

for (size\_t i = 0; i < Buttons.size(); i++)

delete Buttons.at(i);

for (size\_t i = 0; i < Inputs.size(); i++)

delete Inputs.at(i);

for (size\_t i = 0; i < ItemLists.size(); i++)

delete ItemLists.at(i);

for (size\_t i = 0; i < DropDownLists.size(); i++)

delete DropDownLists.at(i);

for (size\_t i = 0; i < Labels.size(); i++)

delete Labels.at(i);

for (size\_t i = 0; i < TextFields.size(); i++)

delete TextFields.at(i);

Buttons.clear();

Inputs.clear();

ItemLists.clear();

DropDownLists.clear();

Labels.clear();

TextFields.clear();

for (size\_t i = 0; i < ptrBaseClass.size(); i++)

delete ptrBaseClass.at(i);

ptrBaseClass.clear();

type\_ptrBaseClasses.clear();

TTF\_Quit();

ready\_fill\_classes = false;

}

## 3.4. Файл Interface\_OnEvent.cpp

#include "app.h"

void Interface::onEvent()

{

while (running) {

while (SDL\_PollEvent(&event)) {

switch (event.type)

{

case SDL\_MOUSEMOTION:

mouseMotion(&event);

break;

case SDL\_MOUSEBUTTONDOWN:

mouseButtonDown(&event);

break;

case SDL\_MOUSEBUTTONUP:

mouseButtonUp(&event);

break;

case SDL\_QUIT:

quit();

break;

case SDL\_KEYDOWN:

keyDown(&event);

break;

default: break;

}

}

}

}

## 3.5. Файл Interface\_OnEvent\_Function.cpp

#include "app.h"

// mouse events

void Interface::mouseMotion(SDL\_Event\* event) {

int x, y;

SDL\_GetMouseState(&x, &y);

for (int i = 0; i < ItemLists.size(); i++) {

if (ItemLists.at(i)->checkItemHover(x, y))

ItemLists.at(i)->render();

}

for (int i = 0; i < DropDownLists.size(); i++) {

if (DropDownLists.at(i)->checkItemHover(x, y))

DropDownLists.at(i)->render();

}

}

void Interface::mouseButtonDown(SDL\_Event\* event)

{

int x, y;

SDL\_GetMouseState(&x, &y);

for (int i = 0; i < Buttons.size(); i++) {

if (Buttons.at(i)->checkHover(x, y)) {

Buttons.at(i)->Click(true);

Buttons.at(i)->render();

switch (i)

{

case BUTTON\_OK\_COUNT\_ELEMENT: {

string data = DropDownLists.at(0)->getValue();

if (data == "")

break;

TextFields.at(0)->add("Количество классов: " + data);

TextFields.at(0)->renderText();

count\_element = atoi(data.c\_str());

DropDownLists.at(0)->blocked();

DropDownLists.at(0)->render();

Buttons.at(0)->blocked();

Buttons.at(0)->render();

Buttons.at(1)->unlock();

Buttons.at(1)->render();

DropDownLists.at(1)->unlock();

DropDownLists.at(1)->render();

Inputs.at(0)->unlock();

Inputs.at(0)->render();

TextFields.at(0)->add("Ввод 1-го элемента");

TextFields.at(0)->renderText();

Labels.at(1)->setText("Enter 1-st element");

Labels.at(1)->render();

break;

}

case BUTTON\_OK\_SET\_START\_VALUE: {

if (ready\_fill\_classes)

break;

string value = Inputs.at(0)->getValue();

if (value == "")

break;

int type = DropDownLists.at(1)->getFlag();

DropDownLists.at(1)->clear();

DropDownLists.at(1)->render();

Inputs.at(0)->clear();

Inputs.at(0)->render();

switch (type)

{

case DROP\_BASE\_CLASS:

ptrBaseClass.push\_back(new String(value.c\_str()));

type\_ptrBaseClasses.push\_back(STRING\_BASE);

DropDownLists.at(2)->add(value, 10 + count\_ready);

DropDownLists.at(2)->render();

TextFields.at(0)->add("Тип класса: Базовый класс");

TextFields.at(0)->renderText();

break;

case DROP\_ID\_CLASS:

ptrBaseClass.push\_back(new ID\_String(value.c\_str()));

type\_ptrBaseClasses.push\_back(STRING\_ID);

DropDownLists.at(2)->add(value, 10 + count\_ready);

DropDownLists.at(2)->render();

TextFields.at(0)->add("Тип класса: Класс идентификатора");

TextFields.at(0)->renderText();

break;

case DROP\_BIN\_CLASS:

ptrBaseClass.push\_back(new BIN\_String(value.c\_str()));

type\_ptrBaseClasses.push\_back(STRING\_BIN);

DropDownLists.at(2)->add(value, 10 + count\_ready);

DropDownLists.at(2)->render();

TextFields.at(0)->add("Тип класса: Бинарная строка");

TextFields.at(0)->renderText();

break;

default: break;

}

if (count\_ready + 1 == count\_element) {

TextFields.at(0)->add("Значение класса: " + value);

TextFields.at(0)->renderText();

TextFields.at(0)->add("Все классы заполнены!");

TextFields.at(0)->renderText();

Labels.at(1)->setText("All classes has filled!");

Labels.at(1)->render();

ready\_fill\_classes = true;

DropDownLists.at(1)->blocked();

DropDownLists.at(1)->render();

Inputs.at(0)->blocked();

Inputs.at(0)->render();

Buttons.at(1)->blocked();

Buttons.at(1)->render();

}

else {

TextFields.at(0)->add("Значение класса: " + value);

TextFields.at(0)->renderText();

char\* count\_text = new char[5];

\_itoa(count\_ready + 2, count\_text, 10);

string countText(count\_text);

TextFields.at(0)->add("Ввод " + countText + "-го элемента");

TextFields.at(0)->renderText();

Labels.at(1)->setText("Enter " + countText + "-th element");

Labels.at(1)->render();

}

count\_ready++;

break;

}

case BUTTON\_OK\_PERFORM\_OPERATION: {

switch (nowOperation)

{

case TEST\_STRING\_EQUAL: {

int indexFirst = DropDownLists.at(3)->getFlag() - 10;

int indexSecond = DropDownLists.at(4)->getFlag() - 10;

/// ОСНОВНОЕ ПРИСВАИВАНИЕ

\*ptrBaseClass.at(indexFirst) = \*ptrBaseClass.at(indexSecond);

///

DropDownLists.at(2)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

DropDownLists.at(2)->add(newElement, 10 + i);

}

Labels.at(4)->show(false);

Labels.at(5)->show(false);

Labels.at(4)->render();

Labels.at(5)->render();

Buttons.at(2)->hide();

Buttons.at(2)->render();

DropDownLists.at(3)->hide();

DropDownLists.at(4)->hide();

DropDownLists.at(2)->render();

DropDownLists.at(3)->render();

DropDownLists.at(4)->render();

nowOperation = NULLELE;

break;

}

case TEST\_STRING\_GET\_LENGTH: {

int indexFirst = DropDownLists.at(3)->getFlag() - 10;

int length = ptrBaseClass.at(indexFirst)->getLength();

char\* length\_str = new char[10];

\_itoa(length, length\_str, 10);

string len\_num(length\_str);

string len\_str = "Длина строки равна: " + len\_num;

TextFields.at(0)->add(len\_str);

TextFields.at(0)->renderText();

Buttons.at(2)->hide();

Buttons.at(2)->render();

Labels.at(4)->show(false);

Labels.at(5)->show(false);

Labels.at(4)->render();

Labels.at(5)->render();

DropDownLists.at(3)->hide();

DropDownLists.at(4)->hide();

DropDownLists.at(3)->render();

DropDownLists.at(4)->render();

nowOperation = NULLELE;

break;

}

case TEST\_IDENTIFICATION\_OPERATOR\_TO\_LOWER\_CASE: {

int indexFirst = DropDownLists.at(3)->getFlag() - 10;

/// ОСНОВНАЯ ФУНКЦИЯ

((ID\_String\*)ptrBaseClass.at(indexFirst))->toLowerCase();

///

DropDownLists.at(2)->deleteItems()->clear();

DropDownLists.at(3)->deleteItems()->clear();

DropDownLists.at(4)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

DropDownLists.at(2)->add(newElement, 10 + i);

}

Labels.at(4)->show(false);

Labels.at(5)->show(false);

Labels.at(4)->render();

Labels.at(5)->render();

Buttons.at(2)->hide();

Buttons.at(2)->render();

DropDownLists.at(3)->hide();

DropDownLists.at(4)->hide();

DropDownLists.at(2)->render();

DropDownLists.at(3)->render();

DropDownLists.at(4)->render();

nowOperation = NULLELE;

break;

}

case TEST\_IDENTIFICATION\_OPERATOR\_EQUAL: {

int indexFirst = DropDownLists.at(3)->getFlag() - 10;

int indexSecond = DropDownLists.at(4)->getFlag() - 10;

/// ОСНОВНОЕ ПРИСВАИВАНИЕ

\*ptrBaseClass.at(indexFirst) = \*ptrBaseClass.at(indexSecond);

///

DropDownLists.at(2)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

DropDownLists.at(2)->add(newElement, 10 + i);

}

Labels.at(4)->show(false);

Labels.at(5)->show(false);

Labels.at(4)->render();

Labels.at(5)->render();

Buttons.at(2)->hide();

Buttons.at(2)->render();

DropDownLists.at(3)->hide();

DropDownLists.at(4)->hide();

DropDownLists.at(2)->render();

DropDownLists.at(3)->render();

DropDownLists.at(4)->render();

nowOperation = NULLELE;

break;

}

case TEST\_IDENTIFICATION\_OPERATOR\_SUBTRACTION: {

int indexFirst = DropDownLists.at(3)->getFlag() - 10;

int indexSecond = DropDownLists.at(4)->getFlag() - 10;

/// ОСНОВНОЕ ПРИСВАИВАНИЕ

ptrBaseClass.push\_back(new ID\_String());

type\_ptrBaseClasses.push\_back(STRING\_ID);

ID\_String\* first = (ID\_String\*)(ptrBaseClass.at(indexFirst));

ID\_String\* second = (ID\_String\*)(ptrBaseClass.at(indexSecond));

\*ptrBaseClass.at(ptrBaseClass.size() - 1) = \*first - \*second;

///

DropDownLists.at(2)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

DropDownLists.at(2)->add(newElement, 10 + i);

}

Labels.at(4)->show(false);

Labels.at(5)->show(false);

Labels.at(4)->render();

Labels.at(5)->render();

Buttons.at(2)->hide();

Buttons.at(2)->render();

DropDownLists.at(3)->hide();

DropDownLists.at(4)->hide();

DropDownLists.at(2)->render();

DropDownLists.at(3)->render();

DropDownLists.at(4)->render();

nowOperation = NULLELE;

break;

}

case TEST\_IDENTIFICATION\_INDEX\_OF: {

int indexFirst = DropDownLists.at(3)->getFlag() - 10;

char symbol = Inputs.at(1)->getValue().at(0);

ID\_String\* first = (ID\_String\*)(ptrBaseClass.at(indexFirst));

int place = 0;

/// ОСНОВНОЕ ПРИСВАИВАНИЕ

char\* strs = new char[2];

strs[0] = symbol;

strs[1] = '\0';

string symbolStr(strs);

delete[] strs;

string value = first->getString();

char\* num = new char[5];

place = first->indexOf(symbol);

\_itoa(place, num, 10);

string number(num);

if (place != -1)

TextFields.at(0)->add("Символ " + symbolStr + " найден в строке " + value + " на " + number + "-тои месте");

else

TextFields.at(0)->add("Символ " + symbolStr + " НЕ найден в строке " + value);

TextFields.at(0)->renderText();

///

Labels.at(6)->show(false);

Labels.at(4)->show(false);

Labels.at(6)->render();

Labels.at(4)->render();

Buttons.at(2)->hide();

Buttons.at(2)->render();

Inputs.at(1)->clear();

Inputs.at(1)->hide();

Inputs.at(1)->render();

DropDownLists.at(3)->hide();

DropDownLists.at(3)->render();

nowOperation = NULLELE;

break;

}

case TEST\_BINARY\_REVERSE: {

int indexFirst = DropDownLists.at(3)->getFlag() - 10;

/// ОСНОВНОЕ ПРИСВАИВАНИЕ

((BIN\_String\*)(ptrBaseClass.at(indexFirst)))->invert();

string text = ((BIN\_String\*)(ptrBaseClass.at(indexFirst)))->getString();

TextFields.at(0)->add("Результат инвертирования равен: " + text);

TextFields.at(0)->renderText();

///

DropDownLists.at(2)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

DropDownLists.at(2)->add(newElement, 10 + i);

}

Labels.at(4)->show(false);

Labels.at(5)->show(false);

Labels.at(4)->render();

Labels.at(5)->render();

Buttons.at(2)->hide();

Buttons.at(2)->render();

DropDownLists.at(3)->hide();

DropDownLists.at(2)->render();

DropDownLists.at(3)->render();

nowOperation = NULLELE;

break;

}

case TEST\_BINARY\_SUBTRACTION: {

int indexFirst = DropDownLists.at(3)->getFlag() - 10;

int indexSecond = DropDownLists.at(4)->getFlag() - 10;

/// ОСНОВНОЕ ПРИСВАИВАНИЕ

BIN\_String\* first = (BIN\_String\*)(ptrBaseClass.at(indexFirst));

BIN\_String\* second = (BIN\_String\*)(ptrBaseClass.at(indexSecond));

BIN\_String result;

result = \*first - \*second;

string text = result.getString();

TextFields.at(0)->add("Результат вычитания равен: " + text);

TextFields.at(0)->renderText();

///

DropDownLists.at(2)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

DropDownLists.at(2)->add(newElement, 10 + i);

}

Labels.at(4)->show(false);

Labels.at(5)->show(false);

Labels.at(4)->render();

Labels.at(5)->render();

Buttons.at(2)->hide();

Buttons.at(2)->render();

DropDownLists.at(3)->hide();

DropDownLists.at(4)->hide();

DropDownLists.at(2)->render();

DropDownLists.at(3)->render();

DropDownLists.at(4)->render();

nowOperation = NULLELE;

break;

}

default: break;

}

break;

}

case BUTTON\_QUIT: {

quit();

break;

}

default: break;

}

}

}

}

void Interface::mouseButtonUp(SDL\_Event\* event)

{

int x, y;

SDL\_GetMouseState(&x, &y);

for (int i = 0; i < Buttons.size(); i++) {

Buttons.at(i)->Click(false);

Buttons.at(i)->render();

}

for (int i = 0; i < Inputs.size(); i++) {

if (Inputs.at(i)->checkHover(x, y)) {

Inputs.at(i)->Focus(true);

Inputs.at(i)->render();

Inputs.at(i)->onEvent(event);

}

}

for (int i = 0; i < ItemLists.size(); i++) {

int flag = ItemLists.at(i)->checkItemHover(x, y);

if (ItemLists.at(i)->checkHover(x, y)) {

if (!ItemLists.at(i)->open()) {

for (int i = 0; i < ItemLists.size(); i++) {

ItemLists.at(i)->open(false);

ItemLists.at(i)->render();

}

ItemLists.at(i)->open(true);

ItemLists.at(i)->render();

}

else {

ItemLists.at(i)->open(false);

ItemLists.at(i)->render();

}

}

else {

if (flag) {

for (size\_t i = 0; i < ItemLists.size(); i++) {

ItemLists.at(i)->open(false);

ItemLists.at(i)->render();

}

switch (flag)

{

case TEST\_STRING\_EQUAL: {

short count = 0;

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++)

if (type\_ptrBaseClasses.at(i) == STRING\_BASE)

count++;

if (count < 2) {

// ошибка

TextFields.at(0)->add("Слишком мало объектов базового класса");

TextFields.at(0)->renderText();

return;

}

Buttons.at(2)->show();

Buttons.at(2)->render();

DropDownLists.at(3)->show();

DropDownLists.at(4)->show();

nowOperation = TEST\_STRING\_EQUAL;

Labels.at(4)->show(true);

Labels.at(5)->show(true);

Labels.at(4)->render();

Labels.at(5)->render();

DropDownLists.at(3)->deleteItems()->clear();

DropDownLists.at(4)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

if (type\_ptrBaseClasses.at(i) == STRING\_BASE){

DropDownLists.at(3)->add(newElement, 10 + i);

DropDownLists.at(4)->add(newElement, 10 + i);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_ID){

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_BIN) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

}

DropDownLists.at(3)->render();

DropDownLists.at(4)->render();

break;

}

case TEST\_STRING\_GET\_LENGTH: {

short count = 0;

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++)

if (type\_ptrBaseClasses.at(i) == STRING\_BASE)

count++;

if (count < 1) {

// ошибка

TextFields.at(0)->add("Слишком мало объектов базового класса");

TextFields.at(0)->renderText();

return;

}

Buttons.at(2)->show();

Buttons.at(2)->render();

DropDownLists.at(3)->show();

nowOperation = TEST\_STRING\_GET\_LENGTH;

Labels.at(4)->show(true);

Labels.at(4)->render();

DropDownLists.at(3)->deleteItems()->clear();

DropDownLists.at(4)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

if (type\_ptrBaseClasses.at(i) == STRING\_BASE) {

DropDownLists.at(3)->add(newElement, 10 + i);

DropDownLists.at(4)->add(newElement, 10 + i);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_ID) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_BIN) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

}

DropDownLists.at(3)->render();

break;

}

case TEST\_IDENTIFICATION\_OPERATOR\_TO\_LOWER\_CASE: {

short count = 0;

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++)

if (type\_ptrBaseClasses.at(i) == STRING\_ID)

count++;

if (count < 1) {

// ошибка

TextFields.at(0)->add("Слишком мало объектов класса идентификатора");

TextFields.at(0)->renderText();

return;

}

Buttons.at(2)->show();

Buttons.at(2)->render();

DropDownLists.at(3)->show();

nowOperation = TEST\_IDENTIFICATION\_OPERATOR\_TO\_LOWER\_CASE;

Labels.at(4)->show(true);

Labels.at(4)->render();

DropDownLists.at(3)->deleteItems()->clear();

DropDownLists.at(4)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

if (type\_ptrBaseClasses.at(i) == STRING\_BASE) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_ID) {

DropDownLists.at(3)->add(newElement, 10 + i);

DropDownLists.at(4)->add(newElement, 10 + i);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_BIN) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

}

DropDownLists.at(3)->render();

break;

}

case TEST\_IDENTIFICATION\_OPERATOR\_EQUAL: {

short count = 0;

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++)

if (type\_ptrBaseClasses.at(i) == STRING\_ID)

count++;

if (count < 2) {

// ошибка

TextFields.at(0)->add("Слишком мало объектов класса идентификатора");

TextFields.at(0)->renderText();

return;

}

Buttons.at(2)->show();

Buttons.at(2)->render();

nowOperation = TEST\_IDENTIFICATION\_OPERATOR\_EQUAL;

DropDownLists.at(3)->show();

DropDownLists.at(4)->show();

Labels.at(4)->show(true);

Labels.at(5)->show(true);

Labels.at(4)->render();

Labels.at(5)->render();

DropDownLists.at(3)->deleteItems()->clear();

DropDownLists.at(4)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

if (type\_ptrBaseClasses.at(i) == STRING\_BASE) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_ID) {

DropDownLists.at(3)->add(newElement, 10 + i);

DropDownLists.at(4)->add(newElement, 10 + i);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_BIN) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

}

DropDownLists.at(3)->render();

DropDownLists.at(4)->render();

break;

}

case TEST\_IDENTIFICATION\_OPERATOR\_SUBTRACTION: {

short count = 0;

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++)

if (type\_ptrBaseClasses.at(i) == STRING\_ID)

count++;

if (count < 2) {

// ошибка

TextFields.at(0)->add("Слишком мало объектов класса идентификатора");

TextFields.at(0)->renderText();

return;

}

Buttons.at(2)->show();

Buttons.at(2)->render();

nowOperation = TEST\_IDENTIFICATION\_OPERATOR\_SUBTRACTION;

DropDownLists.at(3)->show();

DropDownLists.at(4)->show();

Labels.at(4)->show(true);

Labels.at(5)->show(true);

Labels.at(4)->render();

Labels.at(5)->render();

DropDownLists.at(3)->deleteItems()->clear();

DropDownLists.at(4)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

if (type\_ptrBaseClasses.at(i) == STRING\_BASE) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_ID) {

DropDownLists.at(3)->add(newElement, 10 + i);

DropDownLists.at(4)->add(newElement, 10 + i);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_BIN) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

}

DropDownLists.at(3)->render();

DropDownLists.at(4)->render();

break;

}

case TEST\_IDENTIFICATION\_INDEX\_OF: {

short count = 0;

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++)

if (type\_ptrBaseClasses.at(i) == STRING\_ID)

count++;

if (count < 1) {

// ошибка

TextFields.at(0)->add("Слишком мало объектов класса идентификатора");

TextFields.at(0)->renderText();

return;

}

Buttons.at(2)->show();

Buttons.at(2)->render();

DropDownLists.at(3)->show();

Labels.at(6)->show(true);

Labels.at(6)->render();

Inputs.at(1)->show();

Inputs.at(1)->render();

nowOperation = TEST\_IDENTIFICATION\_INDEX\_OF;

Labels.at(4)->show(true);

Labels.at(4)->render();

DropDownLists.at(3)->deleteItems()->clear();

DropDownLists.at(4)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

if (type\_ptrBaseClasses.at(i) == STRING\_BASE) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_ID) {

DropDownLists.at(3)->add(newElement, 10 + i);

DropDownLists.at(4)->add(newElement, 10 + i);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_BIN) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

}

DropDownLists.at(3)->render();

break;

}

case TEST\_BINARY\_REVERSE: {

short count = 0;

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++)

if (type\_ptrBaseClasses.at(i) == STRING\_BIN)

count++;

if (count < 1) {

// ошибка

TextFields.at(0)->add("Слишком мало объектов класа десятичного числа");

TextFields.at(0)->renderText();

return;

}

Buttons.at(2)->show();

Buttons.at(2)->render();

nowOperation = TEST\_BINARY\_REVERSE;

DropDownLists.at(3)->show();

Labels.at(4)->show(true);

Labels.at(4)->render();

DropDownLists.at(3)->deleteItems()->clear();

DropDownLists.at(4)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

if (type\_ptrBaseClasses.at(i) == STRING\_BASE) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_ID) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_BIN) {

DropDownLists.at(3)->add(newElement, 10 + i);

DropDownLists.at(4)->add(newElement, 10 + i);

}

}

DropDownLists.at(3)->render();

DropDownLists.at(4)->render();

break;

}

case TEST\_BINARY\_SUBTRACTION: {

short count = 0;

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++)

if (type\_ptrBaseClasses.at(i) == STRING\_BIN)

count++;

if (count < 2) {

// ошибка

TextFields.at(0)->add("Слишком мало объектов класса двоичного числа");

TextFields.at(0)->renderText();

return;

}

Buttons.at(2)->show();

Buttons.at(2)->render();

nowOperation = TEST\_BINARY\_SUBTRACTION;

DropDownLists.at(3)->show();

DropDownLists.at(4)->show();

Labels.at(4)->show(true);

Labels.at(5)->show(true);

Labels.at(4)->render();

Labels.at(5)->render();

DropDownLists.at(3)->deleteItems()->clear();

DropDownLists.at(4)->deleteItems()->clear();

for (size\_t i = 0; i < type\_ptrBaseClasses.size(); i++) {

string newElement(ptrBaseClass.at(i)->getString());

if (type\_ptrBaseClasses.at(i) == STRING\_BASE) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_ID) {

DropDownLists.at(3)->add(newElement, 10 + i)->Block(true);

DropDownLists.at(4)->add(newElement, 10 + i)->Block(true);

}

else if (type\_ptrBaseClasses.at(i) == STRING\_BIN) {

DropDownLists.at(3)->add(newElement, 10 + i);

DropDownLists.at(4)->add(newElement, 10 + i);

}

}

DropDownLists.at(3)->render();

DropDownLists.at(4)->render();

break;

}

default: break;

}

}

else {

if (ItemLists.at(i)->open()) {

ItemLists.at(i)->open(false);

ItemLists.at(i)->render();

}

}

}

}

for (int i = 0; i < DropDownLists.size(); i++)

{

int flag = DropDownLists.at(i)->checkItemHover(x, y);

if (DropDownLists.at(i)->checkHover(x, y)) {

if (!DropDownLists.at(i)->open()) {

for (int j = 0; j < DropDownLists.size(); j++) {

if (DropDownLists.at(j)->open()) {

DropDownLists.at(j)->open(false);

DropDownLists.at(j)->render();

}

}

DropDownLists.at(i)->open(true);

}

else

DropDownLists.at(i)->open(false);

DropDownLists.at(i)->render();

}

else if (flag)

{

DropDownLists.at(i)->setValue(DropDownLists.at(i)->getItems().at(flag - 10)->text);

DropDownLists.at(i)->open(false);

DropDownLists.at(i)->render();

}

else {

if (DropDownLists.at(i)->open()) {

DropDownLists.at(i)->open(false);

DropDownLists.at(i)->render();

}

}

}

}

void Interface::keyDown(SDL\_Event\* event)

{

switch (event->key.keysym.sym)

{

case SDLK\_RETURN:

cout << "Нажат Enter в APP" << endl;

break;

}

}

## 3.6. Файл Interface\_OnExecute.cpp

#include "app.h"

int Interface::onExecute()

{

init();

SDL\_Rect coord = { 520, 80, 0, 20 };

Label\* label = new Label(renderer, &coord, BY\_LEFT, "Count of Elements", "Fonts/verdana.ttf", 11);

Labels.push\_back(label);

SDL\_Rect mainList = { 520, 100, 160, 20 };

SDL\_Rect item = { -1, -1, 30, 20 };

DropDownList\* DropDownList1 = new DropDownList(renderer, &mainList, &item, LEFT\_SIDE, "Fonts/verdana.ttf", 12);

for (size\_t i = 1; i < 11; i++){

char\* num = new char[10];

\_itoa(i, num, 10);

DropDownList1->add(num, i + 9);

}

DropDownLists.push\_back(DropDownList1);

SDL\_Rect rect1 = { 520, 130, 160, 20 };

Button\_\* newButton1 = new Button\_(

renderer,

&rect1,

"OK",

BUTTON\_OK\_COUNT\_ELEMENT,

"Fonts/verdana.ttf", 12

);

Buttons.push\_back(newButton1);

SDL\_Rect coord3 = { 520, 200, 0, 20 };

Label\* label3 = new Label(renderer, &coord3, BY\_LEFT, "Count of items not set", "Fonts/verdana.ttf", 11);

Labels.push\_back(label3);

SDL\_Rect coord1 = { 520, 220, 0, 20 };

Label\* label1 = new Label(renderer, &coord1, BY\_LEFT, "Type", "Fonts/verdana.ttf", 11);

Labels.push\_back(label1);

SDL\_Rect mainList1 = { 520, 240, 160, 20 };

SDL\_Rect item1 = { -1, -1, 160, 20 };

DropDownList\* DropDownList2 = new DropDownList(renderer, &mainList1, &item1, LEFT\_SIDE, "Fonts/verdana.ttf", 12);

DropDownList2->add("Base class", DROP\_BASE\_CLASS);

DropDownList2->add("Identification class", DROP\_ID\_CLASS);

DropDownList2->add("Binary string class", DROP\_BIN\_CLASS);

DropDownList2->blocked();

DropDownLists.push\_back(DropDownList2);

SDL\_Rect coord2 = { 520, 260, 0, 20 };

Label\* label2 = new Label(renderer, &coord2, BY\_LEFT, "Value", "Fonts/verdana.ttf", 11);

Labels.push\_back(label2);

SDL\_Rect rect = { 520, 280, 160, 20 };

Input\_\* newInput = new Input\_(renderer, &rect, INPUT\_COUNT\_ELEMENT, "Fonts/verdana.ttf", 12);

newInput->blocked();

Inputs.push\_back(newInput);

SDL\_Rect rect2 = { 520, 310, 160, 20 };

Button\_\* newButton2 = new Button\_(

renderer,

&rect2,

"OK",

BUTTON\_OK\_SET\_START\_VALUE,

"Fonts/verdana.ttf", 12

);

newButton2->blocked();

Buttons.push\_back(newButton2);

// Выпадающий список со всеми классами //

SDL\_Rect mainList2 = { 850, 100, 160, 20 };

SDL\_Rect item2 = { -1, -1, 200, 20 };

DropDownList\* DropDownList3 = new DropDownList(renderer, &mainList2, &item2, LEFT\_SIDE, "Fonts/verdana.ttf", 12);

DropDownLists.push\_back(DropDownList3);

//##### Выбор операций над классами ######//

SDL\_Rect itemList\_ItemRect = { -1, -1, 160, 20 };

// Выпадающий список для базового класса //

{

SDL\_Rect itemListRect1 = { 850, 125, 160, 20 };

ItemList\_\* itemList1 = new ItemList\_(renderer, &itemListRect1, &itemList\_ItemRect, "String", "Fonts/verdana.ttf", 12);

itemList1->add("Operator =", TEST\_STRING\_EQUAL);

itemList1->add("Get length", TEST\_STRING\_GET\_LENGTH);

ItemLists.push\_back(itemList1);

}

// Выпадающий список для класса идентификатора //

{

SDL\_Rect itemListRect2 = { 850, 150, 160, 20 };

ItemList\_\* itemList2 = new ItemList\_(renderer, &itemListRect2, &itemList\_ItemRect, "Identificator", "Fonts/verdana.ttf", 12);

itemList2->add("Operator =", TEST\_IDENTIFICATION\_OPERATOR\_EQUAL);

itemList2->add("To lower case", TEST\_IDENTIFICATION\_OPERATOR\_TO\_LOWER\_CASE);

itemList2->add("Substraction", TEST\_IDENTIFICATION\_OPERATOR\_SUBTRACTION);

itemList2->add("Index of symbol", TEST\_IDENTIFICATION\_INDEX\_OF);

ItemLists.push\_back(itemList2);

}

// Выпадающий список для класса десятичной строки //

{

SDL\_Rect itemListRect3 = { 850, 175, 160, 20 };

ItemList\_\* itemList3 = new ItemList\_(renderer, &itemListRect3, &itemList\_ItemRect, "Binary", "Fonts/verdana.ttf", 12);

itemList3->add("Reverse", TEST\_BINARY\_REVERSE);

itemList3->add("Subtraction", TEST\_BINARY\_SUBTRACTION);

ItemLists.push\_back(itemList3);

}

// Список для выбора первого операнда //

{

SDL\_Rect coord4 = { 750, 220, 0, 20 };

Label\* label4 = new Label(renderer, &coord4, BY\_LEFT, "First operand", "Fonts/verdana.ttf", 11);

label4->show(false);

Labels.push\_back(label4);

SDL\_Rect DropDownList\_Rect3 = { 750, 240, 160, 20 };

SDL\_Rect item\_Rect = { -1, -1, 160, 20 };

DropDownList\* DropDownList4 = new DropDownList(renderer, &DropDownList\_Rect3, &item\_Rect, DOWN\_SIDE, "Fonts/verdana.ttf", 12);

DropDownList4->hide();

DropDownLists.push\_back(DropDownList4);

}

// Список для выбора второго операнда //

{

SDL\_Rect coord5 = { 930, 220, 0, 20 };

Label\* label5 = new Label(renderer, &coord5, BY\_LEFT, "Second operand", "Fonts/verdana.ttf", 11);

label5->show(false);

Labels.push\_back(label5);

SDL\_Rect DropDownList\_Rect4 = { 930, 240, 160, 20 };

SDL\_Rect item\_Rect = { -1, -1, 160, 20 };

DropDownList\* DropDownList5 = new DropDownList(renderer, &DropDownList\_Rect4, &item\_Rect, DOWN\_SIDE, "Fonts/verdana.ttf", 12);

DropDownList5->hide();

DropDownLists.push\_back(DropDownList5);

}

// Поле для ввода символа для поиска //

{

SDL\_Rect coord8 = { 930, 220, 0, 20 };

Label\* label8 = new Label(renderer, &coord8, BY\_LEFT, "Field for symbol", "Fonts/verdana.ttf", 11);

label8->show(false);

Labels.push\_back(label8);

SDL\_Rect coord9 = { 930, 240, 160, 18 };

Input\_\* newInput1 = new Input\_(renderer, &coord9, INPUT\_SYMBOL\_INDEX\_OF, "Fonts/verdana.ttf", 12);

newInput1->hide();

Inputs.push\_back(newInput1);

}

// Кнопка ввода информации

{

SDL\_Rect rect3 = { 850, 450, 160, 20 };

Button\_\* newButton3 = new Button\_(

renderer,

&rect3,

"OK",

BUTTON\_OK\_PERFORM\_OPERATION,

"Fonts/verdana.ttf", 12

);

newButton3->hide();

Buttons.push\_back(newButton3);

}

{

SDL\_Rect coord6 = { 0, 0, 400, 500 };

TextField\* textField = new TextField(renderer, &coord6, "Fonts/verdana.ttf", 11);

TextFields.push\_back(textField);

}

SDL\_Rect coord7 = { 850, 80, 0, 20 };

Label\* label7 = new Label(renderer, &coord7, BY\_LEFT, "All classes", "Fonts/verdana.ttf", 11);

Labels.push\_back(label7);

SDL\_Rect rect21 = { 1080, 20, 100, 20 };

Button\_\* newButton21 = new Button\_(

renderer,

&rect21,

"Quit",

BUTTON\_QUIT,

"Fonts/verdana.ttf", 12

);

Buttons.push\_back(newButton21);

render();

onEvent();

return 0;

}

## 3.7. Файл color.h

#pragma once

#include "SDL.h"

struct COLOR {

// Темная тема

COLOR() {

background = { 0x33, 0x33, 0x33, 0x00 };

element\_blocked = { 0x2D, 0x2D, 0x30, 0x00 };

element\_border = { 0x43, 0x43, 0x46, 0x00 };

element\_background\_click = { 0x2D, 0x2D, 0x2D, 0x00 };

element\_background\_focus = { 0x1E, 0x1E, 0x1E, 0x00 };

element\_background\_unfocus = { 0x1E, 0x1E, 0x1E, 0x00 };

element\_border\_click = { 0x50, 0xA0, 0xDF, 0x00 };

element\_list\_hover = { 0x00, 0x7A, 0xCC, 0x00 };

element\_background = { 0x1E, 0x1E, 0x1E, 0x00 };

element\_text = { 0xFF, 0xFF, 0xFF, 0xFF };

}

// Светлая тема

/\*COLOR() {

background = { 0xF0, 0xF0, 0xF0, 0x00 };

element\_blocked = { 0xA0, 0xA0, 0xA0, 0x00 };

element\_border = { 0xAD, 0xAD, 0xAD, 0x00 };

element\_background\_click = { 0xE5, 0xF1, 0xFB, 0x00 };

element\_background\_focus = { 0xFD, 0xFD, 0xFD, 0x00 };

element\_background\_unfocus = { 0xFD, 0xFD, 0xFD, 0x00 };

element\_border\_click = { 0x50, 0xA0, 0xDF, 0x00 };

element\_list\_hover = { 0x50, 0xA0, 0xDF, 0x00 };

element\_background = { 0xE1, 0xE1, 0xE1, 0x00 };

element\_text = { 0x00, 0x00, 0x00, 0x00 };

}\*/

SDL\_Color background;

SDL\_Color element\_blocked;

SDL\_Color element\_border;

SDL\_Color element\_background\_click;

SDL\_Color element\_background\_focus;

SDL\_Color element\_background\_unfocus;

SDL\_Color element\_border\_click;

SDL\_Color element\_list\_hover;

SDL\_Color element\_background;

SDL\_Color element\_text;

};

## 3.8. Файл String.h

#pragma once

#include "iostream"

using namespace std;

class String {

protected:

char\* str;

unsigned int length;

private:

static unsigned int calls;

public:

String();

String(const String& obj);

String(char\* \_str);

String(const char\* \_str);

String(const char symbol);

~String();

public:

unsigned int getLength();

void operator= (String& obj);

char\* getString();

};

## 3.9. Файл String.cpp

#include "String.h"

unsigned int String::calls = 0;

String::String()

{

str = nullptr;

length = 0;

cout << "Базовый класс. Сработал конструктор без параметров" << endl;

}

String::String(const String& obj)

{

char\* \_str = new char[obj.length + 1];

length = 0;

if (\_str != nullptr) {

for (size\_t i = 0; i < obj.length; i++)

\_str[i] = obj.str[i];

\_str[obj.length] = '\0';

str = \_str;

length = obj.length;

}

else

str = nullptr;

calls++;

cout << "Базовый класс. Конструктор копирования сработал в " << calls << "-раз" << endl;

}

String::String(char\* new\_str)

{

bool flag = false;

if (new\_str[0] == '+')

flag = true;

int length\_new\_str = strlen(new\_str) - flag;

char\* \_str = new char[length\_new\_str + 1];

length = 0;

if (\_str != nullptr) {

for (size\_t i = flag; i < length\_new\_str; i++)

\_str[i] = new\_str[i];

\_str[length\_new\_str] = '\0';

str = \_str;

length = length\_new\_str;

}

else

str = nullptr;

cout << "Базовый класс. Сработал конструктор с СИ-строкой" << endl;

}

String::String(const char\* new\_str)

{

bool flag = false;

if (new\_str[0] == '+')

flag = true;

int length\_new\_str = strlen(new\_str) - flag;

char\* \_str = new char[length\_new\_str + 1];

length = 0;

if (\_str != nullptr) {

for (size\_t i = flag; i < length\_new\_str + flag; i++)

\_str[i - flag] = new\_str[i];

\_str[length\_new\_str] = '\0';

str = \_str;

length = length\_new\_str;

}

else

str = nullptr;

cout << "Базовый класс. Сработал конструктор с константной СИ-строкой" << endl;

}

String::String(const char symbol)

{

char\* \_str = new char;

\*\_str = symbol;

str = \_str;

length = strlen(\_str);

cout << "Базовый класс. Сработал конструктор с одним символом" << endl;

}

String::~String()

{

if (str != nullptr)

delete[] str;

cout << "Базовый класс. Сработал деструктор" << endl;

}

unsigned int String::getLength()

{

cout << "Сработала функция получения длины" << endl;

return length;

}

void String::operator=(String& obj)

{

char\* \_str = new char[obj.length + 1];

length = 0;

if (\_str != nullptr) {

for (size\_t i = 0; i < obj.length; i++)

\_str[i] = obj.str[i];

\_str[obj.length] = '\0';

str = \_str;

length = obj.length;

}

else

str = nullptr;

cout << "Сработала перегрузка операции '='" << endl;

}

char\* String::getString()

{

return str;

}

# Скриншоты программы

1. В спецификации Си-идентификатора первым символом может быть буква или знак подчеркивания, остальные символы могут быть, как буквы или числа, так и знак подчеркивания. [↑](#footnote-ref-1)
2. См. страницу 15. [↑](#footnote-ref-2)