|  |  |
| --- | --- |
| 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](#_Toc11364994)

[ВВЕДЕНИЕ 5](#_Toc11364995)

[Техническое задание 6](#_Toc11364996)

[1. Реализация строк 10](#_Toc11364997)

[1.1. Базовый класс Строка 10](#_Toc11364998)

[Данные-члены класса, объявленные как protected: 10](#_Toc11364999)

[Методы класса \_строка\_: 10](#_Toc11365000)

[1.2. Производный класс. Строка идентификатор 11](#_Toc11365001)

[Методы класса ID\_String: 11](#_Toc11365002)

[1.3. Производный класс. Двоичная строка 12](#_Toc11365003)

[Методы класса BIN\_String: 12](#_Toc11365004)

[2. Реализация элементов управления 13](#_Toc11365005)

[2.1. Базовый класс Control 13](#_Toc11365006)

[Данные-члены класса, объявленные как protected: 13](#_Toc11365007)

[Методы класса Control: 14](#_Toc11365008)

[2.2. Производный класс Button 16](#_Toc11365009)

[Данные-члены класса, объявленные как private: 16](#_Toc11365010)

[Методы класса Button: 16](#_Toc11365011)

[2.3. Производный клаcc Input 17](#_Toc11365012)

[Данные-члены класса, объявленные как private: 17](#_Toc11365013)

[Методы класса Input: 17](#_Toc11365014)

[2.4. Производный класс ItemList 18](#_Toc11365015)

[Данные-члены класса, объявленные как private: 18](#_Toc11365016)

[Методы класса ItemList: 19](#_Toc11365017)

[2.5. Производный класс DropDownList 21](#_Toc11365018)

[Данные-члены класса, объявленные как private: 21](#_Toc11365019)

[Методы класса DropDownList: 22](#_Toc11365020)

[2.6. Производный класс TextField 24](#_Toc11365021)

[Данные-члены класса, объявленные как private: 24](#_Toc11365022)

[Методы класса TextField: 24](#_Toc11365023)

[2.7. Производный класс Label 25](#_Toc11365024)

[Данные-члены класса, объявленные как private: 25](#_Toc11365025)

[Методы класса TextField: 25](#_Toc11365026)

[3. Реализация интерфейса 26](#_Toc11365027)

[Данные-члены класса, объявленные как private: 26](#_Toc11365028)

[Публичные методы класса Interface: 27](#_Toc11365029)

[Приватные методы класса Interface: 28](#_Toc11365030)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 29](#_Toc11365031)

[ПРИЛОЖЕНИЕ 30](#_Toc11365032)

[4. Код программы 30](#_Toc11365033)

[4.1. Файл main.cpp 30](#_Toc11365035)

[4.2. Файл App.h 30](#_Toc11365036)

[4.3. Файл Interface.cpp 31](#_Toc11365037)

[4.4. Файл Interface\_OnEvent.cpp 33](#_Toc11365038)

[4.5. Файл Interface\_OnEvent\_Function.cpp 34](#_Toc11365039)

[4.6. Файл Interface\_OnExecute.cpp 51](#_Toc11365040)

[4.7. Файл Interface\_Render.cpp 54](#_Toc11365041)

[4.8. Файл color.h 55](#_Toc11365042)

[4.9. Файл String.h 56](#_Toc11365043)

[4.10. Файл String.cpp 56](#_Toc11365044)

[4.11. Файл ID\_String.h 58](#_Toc11365045)

[4.12. Файл ID\_String.cpp 59](#_Toc11365046)

[4.13. Файл BIN\_String.h 61](#_Toc11365047)

[4.14. Файл BIN\_String.cpp 61](#_Toc11365048)

[4.15. Файл control.h 65](#_Toc11365049)

[4.16. Файл control.cpp 66](#_Toc11365050)

[4.17. Файл button.h 67](#_Toc11365051)

[4.18. Файл button.cpp 68](#_Toc11365052)

[4.19. Файл input.h 69](#_Toc11365053)

[4.20. Файл input.cpp 69](#_Toc11365054)

[4.21. Файл itemList.h 71](#_Toc11365055)

[4.22. Файл itemList.cpp 72](#_Toc11365056)

[4.23. Файл label.h 74](#_Toc11365057)

[4.24. Файл label.cpp 75](#_Toc11365058)

[4.25. Файл listDown.h 76](#_Toc11365059)

[4.26. Файл listDown.cpp 77](#_Toc11365060)

[4.27. Файл textField.h 80](#_Toc11365061)

[4.28. Файл textField.cpp 81](#_Toc11365062)

[Скриншоты программы 82](#_Toc11365063)

# 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.

# Реализация строк

Реализация классов для работы со строками, является основой данной работы, для которой в дальнейшем разрабатывается графический интерфейс. Далее расмотрены реализации для каждого из требуемых классов.

## 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. Реализация элементов управления

Элементы управления, позволяют конечному пользователю удобно взаимодействовать с разработанной программой. Под элементами управления, в рамках данной работы, подразумеваются следующие элементы:

1. Кнопки;
2. Поля ввода;
3. Поля вывода;
4. Выпадающие списки;
5. Выпадающие списки с основным значением;
6. Надписи.

Далее расмотрены реализации каждого из этих элементов.

## 2.1. Базовый класс Control

Название класса: Control. Является базовым классов для классов Button, Input, ItemList, DropDownList, TextField и 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, указатель на 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, указатель на 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, указатель на 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;

#### Структура Item:

##### 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.

##### 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. ItemList(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, SDL\_Rect\* \_itemSizes, string \_mainLabel, string \_font, int \_font\_size) — конструктор, принимающий на вход renderer, указатель на структуру \_sizes типа SDL\_Rect хранящая в себе размеры и положение списка, указатель на структуру \_itemSizes типа SDL\_Rect хранящая в себе размеры и положение выпадающей части списка, текст на главном теле списка типа string, путь к загружаемому шрифту \_font и размер шрифта \_font\_size;
2. void render() — функция отображения списка;
3. 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;
4. void open(bool value) — функция устанавливающая переменной show\_menu значение равное переданному значению типа bool. По факту, функция открывающая список, или его скрывающая;
5. bool open() — функция возвращающая текущее значение переменной show\_menu;
6. 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;

#### Структура DropDownItem:

##### 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.

##### 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.

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.
4. int flag\_select — переменная хранящая в себе флаг, выбранного пользователем элемента списка;
5. 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 DropDownList:

1. DropDownList(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, SDL\_Rect\* \_itemSizes, int \_type, string \_font, int \_font\_size) — конструктор, принимающий на вход renderer, указатель на структуру \_sizes типа SDL\_Rect хранящая в себе размеры и положение списка, указатель на структуру \_itemSizes типа SDL\_Rect хранящая в себе размеры и положение выпадающей части списка, флаг type, путь к загружаемому шрифту \_font и размер шрифта \_font\_size;
2. void render() — функция отображения списка;
3. 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;
4. void open(bool value) — функция устанавливающая переменной show\_menu значение равное переданному значению типа bool. По факту, функция открывающая список, или его скрывающая;
5. bool open() — функция возвращающая текущее значение переменной show\_menu;
6. 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.
7. void setValue(string value) — функция устанавливающая новый текст, отображаемый в теле списка, из переданной переменной типа string;
8. string getValue() — функция возвращающая строку, отображаемую в основном теле списка;
9. int getFlag() — функция возвращающая флаг, который описывает, какой элемент списка был выбран;
10. vector <DropDownItem\*> getItems() — функция возвращаюшая массив указателей на элементы списка;
11. void clear() — фунция устанавливающая пустую строку в основном теле списка;
12. DropDownList\* deleteItems() — функция удаляющая все элементы списка. Возвращает указатель на на текущий объект класса, для возможности дальнейшего вызова функций в этой же строке.

## 2.6. Производный класс TextField

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

### Данные-члены класса, объявленные как private:

1. vector <string> lines — массив, хранящий в себе набор строк типа string;
2. int count\_lines — переменная хранящая в себе количество строк.

### Методы класса TextField:

1. TextField (SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, string \_font, int \_font\_size) — конструктор, принимающий на вход renderer, указатель на структуру \_sizes типа SDL\_Rect хранящая в себе размеры и положение поля вывода, путь к загружаемому шрифту \_font и размер шрифта \_font\_size;
2. void add(string newLine) — функция для добавления новой строки;
3. void render() — функция отображения поля вывода;
4. void renderText() — функция отображения текста текстового поля;

## 2.7. Производный класс Label

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

### Данные-члены класса, объявленные как private:

1. string label — строка типа string;
2. int type — тип выравнивания (по левому или по правому краю). Определяется следующим перечислением:

enum TYPES {

BY\_RIGHT,

BY\_LEFT

};

### Методы класса TextField:

1. Label(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, int \_type, string \_new\_string, const char\* \_font, int \_font\_size) — конструктор, принимающий на вход renderer, указатель на структуру \_sizes типа SDL\_Rect хранящая в себе размеры и положение надписи, флаг \_type, определяющий тип выравнивания, путь к загружаемому шрифту \_font и размер шрифта \_font\_size;
2. void render() — функция отображения надписи;
3. void show(bool value) — функция устанавливающая переменной display значение равное переданному значению типа bool;
4. void setText(string text) — функция устанавливающая новый текст надписи.

# Реализация интерфейса

Интерфейс является одной из важнейших вещей в программе, так как с его помощью конечный пользователь взаимодействует с программой. Далее расмотрен способ реализации графического интерфейса.

* 1. Класс Interface

Класс Interface связывает элементы управления и классы строк.

### Данные-члены класса, объявленные как private:

1. bool running — переменная указывающая на то, что программа отслеживает события пользователя, при выходе из цикла отслеживания событий, то есть когда running = false, программа завершает свою работу;
2. SDL\_Window\* window — указатель на объект Window в SDL, хранящий по адресу информацию о главном окне программы;
3. SDL\_Renderer\* renderer — указатель на renderer, хранящий по адресу объект для отрисовки SDL;
4. SDL\_Event event — переменная хранящая в себе текущее событие SDL;
5. SDL\_Texture\* texture — указатель хранящий по своему адресу текстуру SDL, которая является основной для отрисовки;
6. short screenWidth — переменная хранящая ширину экрана;
7. short screenHeight — переменная хранящая высоту экрана;
8. COLOR Colors — объект структуры COLOR, описывающей цветовую схему интерфейса;
9. vector <Button\*> Buttons — массив указателей на все объекты кнопок, которые присутствуют в интерфейсе;
10. vector <Input\*> Inputs — массив указателей на все объекты полей ввода, которые присутствуют в интерфейсе;
11. vector <ItemList\*> ItemLists — массив указателей на все объекты выпадающих списков, которые присутствуют в интерфейсе;
12. vector <DropDownList\*> DropDownLists — массив указателей на все объекты выпадающих списков с основным значением, которые присутствуют в интерфейсе;
13. vector <Label\*> Labels — массив указателей на все объекты надписей, которые присутствуют в интерфейсе;
14. vector <TextField\*> TextFields — массив указателей на все объекты полей вывода, которые присутствуют в интерфейсе;
15. bool ready\_fill\_classes — переменная указывающая на то, что все классы были заполнены;
16. int count\_ready — переменная-счетчик, для количества уже введеных классов;
17. int count\_element — переменная, хранящая количество необходимых классов;
18. vector <String\*> ptrBaseClass — массив указателей на базовый класс;
19. vector <int> type\_ptrBaseClasses — массив типов для каждого класса, определенные следующим перечислением:

enum STRING\_CLASS\_ID {

STRING\_BASE,

STRING\_ID,

STRING\_BIN

};

для определения типа для каждого из классов.

1. int nowOperation — переменная хранящая текущую операцию приложения;

### Публичные методы класса Interface:

1. Interface() — базовый конструктор класса;
2. Interface(short width, short height) — конструктор принимающий два значения типа short, определяющие ширину и высоту окна программы;
3. ~Interface() — деструктор класса;
4. int onExecute() — основная функция, вызывающая все остальные функции;

### Приватные методы класса Interface:

1. void render();
2. void onEvent();
3. bool init();
4. void mouseButtonDown(SDL\_Event\* event);
5. void mouseButtonUp(SDL\_Event\* event);
6. void mouseMotion(SDL\_Event\* event);
7. void keyDown(SDL\_Event\* event);
8. void quit();

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Б.И. Березин. Начальный курс С и С++. – М.:Издательство Диалог-МИФИ, 2005 г. – 248 с.
2. Р. Лафоре. Объектно-ориентированное программирование в С++. 4-е издание. – Спб.: Издательство ПИТЕР, 2004 г. – 902 с.
3. Б. Страуструп. Язык программирования С++. Специальное издание. Пер. с англ. – М.: Издательство Бином, 2011 г. – 1136 с.

# ПРИЛОЖЕНИЕ

## Код программы



## 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();

}

## Файл App.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

};

class Interface {

private:

bool running;

SDL\_Window\* window;

SDL\_Renderer\* renderer;

SDL\_Event event;

SDL\_Texture\* texture;

short screenWidth;

short screenHeight;

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();

};

## Файл Interface.cpp

#include "App.h"

Interface::Interface()

{

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)

{

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;

}

## 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;

}

}

}

}

## 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;

}

}

## 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;

}

## Файл Interface\_Render.cpp

#include "app.h"

void Interface::render()

{

SDL\_SetRenderDrawColor(renderer, Colors.background.r, Colors.background.g, Colors.background.b, Colors.background.a);

SDL\_RenderFillRect(renderer, NULL);

SDL\_RenderPresent(renderer);

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

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

}

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

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

}

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

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

}

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

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

}

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

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

}

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

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

}

}

## 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;

};

## 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();

};

## 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;

}

## Файл ID\_String.h

#pragma once

#include "string.h"

class ID\_String : public String{

private:

static unsigned int calls;

public:

ID\_String() : String(){};

ID\_String(const ID\_String& obj);

ID\_String(char\* \_str);

ID\_String(const char\* \_str);

~ID\_String();

public:

bool isCorrect(char\* str);

void toLowerCase();

void print();

int indexOf(char symbol);

ID\_String& operator- (const ID\_String& obj);

};

## Файл ID\_String.cpp

#include "ID\_String.h"

unsigned int ID\_String::calls = 0;

ID\_String::ID\_String(char\* \_str) : String(\_str)

{

if (!isCorrect((char\*)\_str)) {

str = nullptr;

length = 0;

}

cout << "Производный класс 'Строка-идентификатор'. Сработал конструктор с СИ-строкой" << endl;

}

ID\_String::ID\_String(const char\* \_str) : String(\_str)

{

if (!isCorrect((char\*)\_str)) {

str = nullptr;

length = 0;

}

cout << "Производный класс 'Строка-идентификатор'. Сработал конструктор с константной СИ-строкой" << endl;

}

ID\_String::~ID\_String()

{

cout << "Производный класс 'Строка-идентификатор'. Сработал деструктор" << endl;

}

ID\_String::ID\_String(const ID\_String& obj) : String(obj)

{

if (!isCorrect((char\*)obj.str)) {

str = nullptr;

length = 0;

}

calls++;

cout << "Производный класс 'Строка-идентификатор'. Конструктор копирования сработал " << calls << "-раз" << endl;

}

bool ID\_String::isCorrect(char\* \_str)

{

cout << "Производный класс 'Строка-идентификатор'. Сработала функция проверки корректности строки" << endl;

if ((isalpha(\_str[0]) || \_str[0] == '\_') && \_str != nullptr) {

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

if (!isalpha(\_str[i]) && !isdigit(\_str[i]) && \_str[i] != '\_')

{

if (str != nullptr) {

str = nullptr;

length = 0;

}

return false;

}

}

return true;

}

return false;

}

void ID\_String::toLowerCase()

{

if (str != nullptr)

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

if (isalpha(str[i]))

str[i] = (char)tolower(str[i]);

cout << "Производный класс 'Строка-идентификатор'. Сработала функция приведения строки в нижний регистр" << endl;

}

int ID\_String::indexOf(char symbol)

{

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

if (str[i] == symbol)

return i;

return -1;

}

ID\_String& ID\_String::operator-(const ID\_String& obj)

{

cout << "Производный класс 'Строка-идентификатор'. Сработала перегрузка оператора '-'" << endl;

ID\_String\* backString = nullptr;

if (str != nullptr) {

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

if (\_str != nullptr) {

bool flag = true;

unsigned int newLength = 0;

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

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

if (str[i] == obj.str[j])

flag = false;

if (flag) {

\_str[newLength] = str[i];

newLength++;

}

else

flag = true;

}

if (newLength != 0) {

\_str[newLength] = '\0';

backString = new ID\_String(\_str);

delete[] \_str;

}

else {

delete[] \_str;

\_str = nullptr;

backString = new ID\_String;

}

return \*backString;

}

}

else {

backString = new ID\_String;

return \*backString;

}

}

## Файл BIN\_String.h

#pragma once

#include "string.h"

class BIN\_String : public String {

private:

static unsigned int calls;

public:

BIN\_String() : String() {};

BIN\_String(const BIN\_String& obj);

BIN\_String(char\* \_str);

BIN\_String(const char\* \_str);

~BIN\_String();

public:

bool isCorrect(char\* str);

void invert();

BIN\_String& operator- (BIN\_String& obj);

};

## Файл BIN\_String.cpp

#include "BIN\_String.h"

unsigned int BIN\_String::calls = 0;

BIN\_String::BIN\_String(char\* \_str) : String(\_str)

{

if (!isCorrect(\_str)) {

str = nullptr;

length = 0;

}

cout << "Производный класс 'Десятичная строка'. Сработал конструктор с СИ-строкой" << endl;

}

BIN\_String::BIN\_String(const char\* \_str) : String(\_str)

{

if (!isCorrect((char\*)\_str)) {

str = nullptr;

length = 0;

}

cout << "Производный класс 'Десятичная строка'. Сработал конструктор с константной СИ-строкой" << endl;

}

BIN\_String::~BIN\_String()

{

cout << "Производный класс 'Десятичная строка'. Сработал деструктор" << endl;

}

BIN\_String::BIN\_String(const BIN\_String& obj) : String(obj.str)

{

if (!isCorrect(obj.str)) {

str = nullptr;

length = 0;

}

calls++;

cout << "Производный класс 'Десятичная строка'. Конструктор копирования сработал " << calls << "-раз" << endl;

}

bool BIN\_String::isCorrect(char\* str)

{

cout << "Производный класс 'Десятичная строка'. Сработала функция проверки корректности строки" << endl;

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

if (str[i] != '0' && str[i] != '1')

return false;

return true;

}

void BIN\_String::invert()

{

if (str != nullptr) {

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

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

if (str[i] == '0')

\_str[i] = '1';

else

\_str[i] = '0';

}

\_str[length] = '\0';

delete[] str;

str = \_str;

}

}

BIN\_String& BIN\_String::operator-(BIN\_String& obj)

{

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

char\* result = new char[length + 1];

if (length > obj.length) {

for (int j = 0; j < length - obj.length; j++)

\_str[j] = '0';

strcpy(\_str + length - obj.length, obj.str);

char symbol;

bool flag = false;

for (size\_t i = length - 1; i > 0; i--) {

symbol = \_str[i];

if (str[i] == '1' && symbol == '1') {

if (flag)

result[i] = '1';

else

result[i] = '0';

flag = true;

}

else if (str[i] == '0' && symbol == '1' || str[i] == '1' && symbol == '0') {

if (flag) {

result[i] = '0';

flag = true;

}

else {

result[i] = '1';

flag = false;

}

}

else {

if (flag) {

result[i] = '1';

flag = false;

}

else {

result[i] = '0';

flag = false;

}

}

}

if (flag)

result[0] = '1';

else

result[0] = '0';

result[length] = '\0';

}

else if (length < obj.length) {

for (int j = 0; j < obj.length - length; j++)

\_str[j] = '0';

strcpy(\_str + obj.length - length, str);

char symbol;

bool flag = false;

for (size\_t i = obj.length - 1; i > 0; i--) {

symbol = obj.str[i];

if (\_str[i] == '1' && symbol == '1') {

if (flag)

result[i] = '0';

else

result[i] = '1';

flag = true;

}

else if (\_str[i] == '0' && symbol == '1' || \_str[i] == '1' && symbol == '0') {

if (flag) {

result[i] = '0';

flag = true;

}

else {

result[i] = '1';

flag = false;

}

}

}

if (flag) {

result[0] = '1';

result[length] = '\0';

}

else {

result[0] = '0';

result[length] = '\0';

}

}

else {

char symbol;

bool flag = false;

for (size\_t i = obj.length - 1; i > 0; i--) {

symbol = obj.str[i];

if (str[i] == '1' && symbol == '1') {

if (flag)

result[i] = '0';

else

result[i] = '1';

flag = true;

}

else if (str[i] == '0' && symbol == '1' || str[i] == '1' && symbol == '0') {

if (flag) {

result[i] = '0';

flag = true;

}

else {

result[i] = '1';

flag = false;

}

}

else {

if (flag) {

result[i] = '1';

flag = false;

}

else {

result[i] = '0';

flag = false;

}

}

}

if (flag) {

result[0] = '1';

result[length] = '\0';

}

else {

result[0] = '0';

result[length] = '\0';

}

}

BIN\_String \* ret = new BIN\_String(result);

return \*ret;

}

## Файл control.h

#pragma once

#include "string"

#include "vector"

#include "SDL.h"

#include "SDL\_image.h"

#include "SDL\_ttf.h"

#include "iostream"

#include "../Color.h"

using namespace std;

enum TYPE\_ALIGN {

CENTERED\_ALIGN,

LEFT\_ALIGN

};

class Control {

protected:

SDL\_Rect\* sizes;

SDL\_Event event;

TTF\_Font\* font;

SDL\_Renderer\* renderer;

COLOR Colors;

int font\_size;

bool block;

bool display;

bool click;

public:

Control(SDL\_Renderer \* renderer, SDL\_Rect\* \_sizes, string \_font, int \_font\_size);

public:

virtual void render();

virtual void renderLabel(string text, SDL\_Rect\* renderRect, int type);

void unlock();

void blocked();

void hide();

void show();

void Click(bool);

bool getShow();

bool checkHover(int x, int y);

virtual void onEvent(SDL\_Event\* event);

};

## Файл control.cpp

#include "control.h"

Control::Control(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, string \_font, int \_font\_size)

{

renderer = \_renderer;

block = false;

sizes = \_sizes;

font\_size = \_font\_size;

display = true;

click = false;

font = TTF\_OpenFont(\_font.c\_str(), \_font\_size);

}

void Control::render()

{

// virtual

}

void Control::renderLabel(string text, SDL\_Rect\* renderRect, int type)

{

SDL\_Texture\* textTexture = nullptr;

SDL\_Surface\* textSurface = nullptr;

SDL\_Rect textRect;

if ((textSurface = TTF\_RenderUTF8\_Blended(font, text.c\_str(), Colors.element\_text)) != nullptr) {

textRect.w = textSurface->w;

textRect.h = textSurface->h;

if (type == LEFT\_ALIGN)

textRect.x = renderRect->x + 10;

else if (type == CENTERED\_ALIGN)

textRect.x = renderRect->x + renderRect->w / 2 - textSurface->w / 2;

textRect.y = renderRect->y + renderRect->h / 2 - textSurface->h / 1.88;

textTexture = SDL\_CreateTextureFromSurface(renderer, textSurface);

SDL\_RenderCopy(renderer, textTexture, NULL, &textRect);

SDL\_FreeSurface(textSurface);

SDL\_DestroyTexture(textTexture);

}

SDL\_RenderPresent(renderer);

}

void Control::unlock()

{

block = false;

}

void Control::blocked()

{

block = true;

}

void Control::hide()

{

display = false;

}

void Control::show()

{

display = true;

}

void Control::Click(bool value)

{

click = value;

}

bool Control::getShow()

{

return display;

}

bool Control::checkHover(int \_x, int \_y)

{

SDL\_Point point = { \_x, \_y };

if (!block && display)

return SDL\_PointInRect(&point, sizes);

return false;

}

void Control::onEvent(SDL\_Event\* event)

{

// virtual

}

## Файл button.h

#pragma once

#include "control.h"

enum BUTTON\_TYPES {

BUTTON\_OK\_COUNT\_ELEMENT,

BUTTON\_OK\_SET\_START\_VALUE,

BUTTON\_OK\_PERFORM\_OPERATION,

BUTTON\_QUIT

};

class Button\_ : public Control {

private:

string text;

int type;

public:

Button\_(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, string \_text, int \_type, string \_font, int \_font\_size) : Control(\_renderer, \_sizes, \_font, \_font\_size)

{

type = \_type;

text = \_text;

};

public:

void render();

};

## Файл button.cpp

#include "button\_.h"

void Button\_::render()

{

if (!display) {

SDL\_Rect clearRect = \*sizes;

clearRect.x--;

clearRect.y--;

clearRect.w += 2;

clearRect.h += 2;

SDL\_SetRenderDrawColor(renderer, Colors.background.r, Colors.background.g, Colors.background.b, Colors.background.a);

SDL\_RenderFillRect(renderer, &clearRect);

SDL\_RenderPresent(renderer);

return;

}

SDL\_Texture\* textTexture;

SDL\_Surface\* textSurface;

SDL\_Rect textRect;

// Отрисовка обводки

{

SDL\_Rect border = \*sizes;

border.x--;

border.y--;

border.w += 2;

border.h += 2;

if (click && !block)

SDL\_SetRenderDrawColor(renderer, Colors.element\_border\_click.r, Colors.element\_border\_click.g, Colors.element\_border\_click.b, Colors.element\_border\_click.a);

else

SDL\_SetRenderDrawColor(renderer, Colors.element\_border.r, Colors.element\_border.g, Colors.element\_border.b, Colors.element\_border.a);

SDL\_RenderFillRect(renderer, &border);

}

// Отрисовка заднего фона в зависимости от состояния

{

if (block)

SDL\_SetRenderDrawColor(renderer, Colors.element\_blocked.r, Colors.element\_blocked.g, Colors.element\_blocked.b, Colors.element\_blocked.a);

else

if (click)

SDL\_SetRenderDrawColor(renderer, Colors.element\_background\_click.r, Colors.element\_background\_click.g, Colors.element\_background\_click.b, Colors.element\_background\_click.a);

else

SDL\_SetRenderDrawColor(renderer, Colors.element\_background.r, Colors.element\_background.g, Colors.element\_background.b, Colors.element\_background.a);

SDL\_RenderFillRect(renderer, sizes);

}

renderLabel(text, sizes, CENTERED\_ALIGN);

SDL\_RenderPresent(renderer);

}

## Файл input.h

#pragma once

#include "control.h"

enum INPUT\_TYPES {

INPUT\_COUNT\_ELEMENT,

INPUT\_START\_VALUE,

INPUT\_SYMBOL\_INDEX\_OF

};

class Input\_ : public Control {

private:

string inputText;

bool focus;

int type;

public:

Input\_(SDL\_Renderer \* \_renderer, SDL\_Rect\* \_sizes, int \_type, string \_font, int \_font\_size) : Control(\_renderer, \_sizes, \_font, \_font\_size)

{

type = \_type;

focus = false;

inputText = "";

};

public:

void render();

void onEvent(SDL\_Event\* event);

void clear();

string getValue();

void Focus(bool value);

};

## Файл input.cpp

#include "input\_.h"

void Input\_::render()

{

if (!display) {

SDL\_Rect\* clearRect = sizes;

clearRect->x--;

clearRect->y--; clearRect;

clearRect->w += 2;

clearRect->h += 2;

SDL\_SetRenderDrawColor(renderer, Colors.background.r, Colors.background.g, Colors.background.b, Colors.background.a);

SDL\_RenderFillRect(renderer, clearRect);

return;

}

// Отрисовка обводки

{

SDL\_Rect border = \*sizes;

border.x--;

border.y--;

border.w += 2;

border.h += 2;

if (focus && !block)

SDL\_SetRenderDrawColor(renderer, Colors.element\_border\_click.r, Colors.element\_border\_click.g, Colors.element\_border\_click.b, Colors.element\_border\_click.a);

else

SDL\_SetRenderDrawColor(renderer, Colors.element\_border.r, Colors.element\_border.g, Colors.element\_border.b, Colors.element\_border.a);

SDL\_RenderFillRect(renderer, &border);

}

// Отрисовка заднего фона в зависимости от состояния

{

if (focus && !block)

SDL\_SetRenderDrawColor(renderer, Colors.element\_background\_focus.r, Colors.element\_background\_focus.g, Colors.element\_background\_focus.b, Colors.element\_background\_focus.a);

else {

if (block)

SDL\_SetRenderDrawColor(renderer, Colors.element\_blocked.r, Colors.element\_blocked.g, Colors.element\_blocked.b, Colors.element\_blocked.a);

else

SDL\_SetRenderDrawColor(renderer, Colors.element\_background\_unfocus.r, Colors.element\_background\_unfocus.g, Colors.element\_background\_unfocus.b, Colors.element\_background\_unfocus.a);

}

}

SDL\_RenderFillRect(renderer, sizes);

renderLabel(inputText, sizes, LEFT\_ALIGN);

}

void Input\_::onEvent(SDL\_Event\* event)

{

while (focus) {

SDL\_StartTextInput();

while (SDL\_PollEvent(event)) {

switch (event->type)

{

case SDL\_KEYDOWN:

{

if (event->key.keysym.sym == SDLK\_BACKSPACE && inputText.length() > 0)

inputText.pop\_back();

else if (event->key.keysym.sym == SDLK\_c && SDL\_GetModState() & KMOD\_CTRL)

SDL\_SetClipboardText(inputText.c\_str());

else if (event->key.keysym.sym == SDLK\_v && SDL\_GetModState() & KMOD\_CTRL)

inputText = SDL\_GetClipboardText();

break;

}

case SDL\_TEXTINPUT:

if (!((event->text.text[0] == 'c' || event->text.text[0] == 'C') && (event->text.text[0] == 'v' || event->text.text[0] == 'V') && SDL\_GetModState() & KMOD\_CTRL))

if (inputText.size() \* (font\_size - 2) + 10 < sizes->w)

inputText += event->text.text;

break;

case SDL\_MOUSEBUTTONDOWN:

{

int x, y;

SDL\_GetMouseState(&x, &y);

SDL\_Point point = { x, y };

if (!SDL\_PointInRect(&point, sizes))

focus = false;

render();

break;

}

}

render();

}

}

SDL\_StopTextInput();

}

void Input\_::clear()

{

inputText = "";

}

string Input\_::getValue()

{

return inputText;

}

void Input\_::Focus(bool value)

{

focus = value;

}

## Файл itemList.h

#pragma once

#include "control.h"

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

};

typedef struct Item {

Item() {

hover = false;

flag = 0;

blocked = false;

}

string text;

bool hover;

int flag;

bool blocked;

void Hover(bool value) {

hover = value;

}

void Block(bool value) {

blocked = value;

}

bool Hover() {

return hover;

}

} Item;

class ItemList\_ : public Control {

private:

vector <Item \*> List;

SDL\_Rect\* itemSizes;

string mainLabel;

bool show\_menu;

public:

ItemList\_(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, SDL\_Rect\* \_itemSizes, string \_mainLabel,string \_font, int \_font\_size) : Control(\_renderer, \_sizes, \_font, \_font\_size)

{

mainLabel = \_mainLabel;

itemSizes = \_itemSizes;

show\_menu = false;

};

public:

void render();

void add(string text, int flag);

void open(bool value);

bool open();

int checkItemHover(int x, int y);

};

## Файл itemList.cpp

#include "itemList\_.h"

void ItemList\_::render()

{

if (!display)

return;

// sizes — размеры главного объекта

// itemSizes — размеры одного эдемента списка

SDL\_Rect renderRect = \*itemSizes;

renderRect.x = sizes->x + sizes->w + 2;

renderRect.y = sizes->y - 1;

// Отрисовка обводки

{

SDL\_Rect border = \*sizes;

border.x--;

border.y--;

border.w += 2;

border.h += 2;

SDL\_SetRenderDrawColor(renderer, Colors.element\_border.r, Colors.element\_border.g, Colors.element\_border.b, Colors.element\_border.a);

SDL\_RenderFillRect(renderer, &border);

}

if (block)

SDL\_SetRenderDrawColor(renderer, Colors.element\_blocked.r, Colors.element\_blocked.g, Colors.element\_blocked.b, Colors.element\_blocked.a);

else

SDL\_SetRenderDrawColor(renderer, Colors.element\_background.r, Colors.element\_background.g, Colors.element\_background.b, Colors.element\_background.a);

SDL\_RenderFillRect(renderer, sizes);

renderLabel(mainLabel, sizes, CENTERED\_ALIGN);

if (show\_menu && !block) {

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

{

if (List.at(i)->Hover())

SDL\_SetRenderDrawColor(renderer, Colors.element\_list\_hover.r, Colors.element\_list\_hover.g, Colors.element\_list\_hover.b, Colors.element\_list\_hover.a);

else

SDL\_SetRenderDrawColor(renderer, Colors.element\_background.r, Colors.element\_background.g, Colors.element\_background.b, Colors.element\_background.a);

SDL\_RenderFillRect(renderer, &renderRect);

renderLabel(List[i]->text, &renderRect, LEFT\_ALIGN);

renderRect.y += renderRect.h;

}

}

else{

SDL\_Rect clearRect = { renderRect.x, renderRect.y, renderRect.w, renderRect.h \* List.size() };

SDL\_SetRenderDrawColor(renderer, Colors.background.r , Colors.background.g, Colors.background.b, Colors.background.a);

SDL\_RenderFillRect(renderer, &clearRect);

}

SDL\_RenderPresent(renderer);

}

void ItemList\_::add(string text, int flag)

{

Item\* newItem = new Item;

newItem->text = text;

newItem->flag = flag;

List.push\_back(newItem);

}

void ItemList\_::open(bool value)

{

show\_menu = value;

}

bool ItemList\_::open()

{

return show\_menu;

}

int ItemList\_::checkItemHover(int \_x, int \_y)

{

if (block || !display || !show\_menu)

return false;

SDL\_Point point = { \_x, \_y };

SDL\_Rect itemRect = \*itemSizes;

itemRect.x = sizes->x + sizes->w;

itemRect.y = sizes->y;

int flag = 0;

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

bool hov = SDL\_PointInRect(&point, &itemRect);

itemRect.y += itemRect.h;

List.at(i)->Hover(false);

if (hov) {

flag = List.at(i)->flag;

List.at(i)->Hover(true);

//return true;

}

}

return flag;

}

## Файл label.h

#pragma once

#include "control.h"

using namespace std;

enum TYPES {

BY\_RIGHT,

BY\_LEFT

};

class Label : public Control{

private:

string label;

int type;

public:

Label(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, int \_type, string \_new\_string, const char\* \_font, int \_font\_size);

public:

void render();

void show(bool value);

void setText(string text);

};

## Файл label.cpp

#include "Label.h"

Label::Label(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, int \_type, string \_new\_string, const char\* \_font, int \_font\_size) : Control(\_renderer, \_sizes, \_font, \_font\_size)

{

renderer = \_renderer;

display = true;

sizes = \_sizes;

type = \_type;

label = \_new\_string;

font = TTF\_OpenFont(\_font, \_font\_size);

};

void Label::render()

{

if (!display) {

SDL\_SetRenderDrawColor(renderer, Colors.background.r, Colors.background.g, Colors.background.b, Colors.background.a);

SDL\_Rect clearRect = \*sizes;

clearRect.w = 160;

SDL\_RenderFillRect(renderer, &clearRect);

return;

}

SDL\_Texture\* textTexture = nullptr;

SDL\_Surface\* textSurface = nullptr;

SDL\_SetRenderDrawColor(renderer, Colors.background.r, Colors.background.g, Colors.background.b, Colors.background.a);

SDL\_Rect clearRect = \*sizes;

clearRect.w = 160;

SDL\_RenderFillRect(renderer, &clearRect);

if ((textSurface = TTF\_RenderUTF8\_Blended(font, label.c\_str(), Colors.element\_text)) != nullptr) {

sizes->w = textSurface->w;

sizes->h = textSurface->h;

if (type == BY\_RIGHT)

sizes->x -= sizes->w - 10;

else if (type == BY\_LEFT)

sizes->x = sizes->x;

sizes->y += sizes->h / 2 - textSurface->h / 2.8;

textTexture = SDL\_CreateTextureFromSurface(renderer, textSurface);

SDL\_RenderCopy(renderer, textTexture, NULL, sizes);

SDL\_FreeSurface(textSurface);

SDL\_DestroyTexture(textTexture);

SDL\_RenderPresent(renderer);

}

}

void Label::show(bool value)

{

display = value;

}

void Label::setText(string text)

{

label = text;

}

## Файл listDown.h

#pragma once

#include "control.h"

enum DROP\_ITEM\_LIST\_TYPES {

DROP\_BASE\_CLASS = 10,

DROP\_ID\_CLASS = 11,

DROP\_BIN\_CLASS = 12

};

enum TYPE {

LEFT\_SIDE = 1,

DOWN\_SIDE = 2

};

typedef struct DropDownItem {

DropDownItem() {

hover = false;

flag = 0;

blocked = false;

}

string text;

bool hover;

int flag;

bool blocked;

void Hover(bool value) {

hover = value;

}

void Block(bool value) {

blocked = value;

}

bool Block() {

return blocked;

}

bool Hover() {

return hover;

}

} DropDownItem;

class DropDownList : public Control {

private:

vector <DropDownItem\*> List;

SDL\_Rect\* itemSizes;

string mainLabel;

bool show\_menu;

int flag\_select;

int type;

SDL\_Texture\* dropButton;

public:

DropDownList(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, SDL\_Rect\* \_itemSizes, int \_type, string \_font, int \_font\_size) : Control(\_renderer, \_sizes, \_font, \_font\_size)

{

type = \_type;

itemSizes = \_itemSizes;

dropButton = IMG\_LoadTexture(renderer, "sfx/drop\_button.png");

show\_menu = false;

};

public:

void render();

DropDownItem\* add(string text, int flag);

void open(bool value);

bool open();

int checkItemHover(int x, int y);

void setValue(string value);

string getValue();

int getFlag();

vector <DropDownItem\*> getItems();

void clear();

DropDownList\* deleteItems();

};

## Файл listDown.cpp

#include "ListDown.h"

void DropDownList::render()

{

if (!display) {

SDL\_Rect clearRect = \*sizes;

clearRect.x--;

clearRect.y--;

clearRect.w += 2;

clearRect.h += 2;

SDL\_SetRenderDrawColor(renderer, Colors.background.r, Colors.background.g, Colors.background.b, Colors.background.a);

SDL\_RenderFillRect(renderer, &clearRect);

return;

}

// sizes — размеры главного объекта

// itemSizes — размеры одного эдемента списка

SDL\_Rect renderRect = \*itemSizes;

if (type == LEFT\_SIDE) {

renderRect.x = sizes->x + sizes->w + 2;

renderRect.y = sizes->y - 1;

}

else if (type == DOWN\_SIDE){

renderRect.x = sizes->x;

renderRect.y = sizes->y + sizes->h + 2;

}

// Отрисовка обводки

{

SDL\_Rect \* border = new SDL\_Rect;

\*border = \*sizes;

border->x--;

border->y--;

border->w += 2;

border->h += 2;

SDL\_SetRenderDrawColor(renderer, Colors.element\_border.r, Colors.element\_border.g, Colors.element\_border.b, Colors.element\_border.a);

SDL\_RenderFillRect(renderer, border);

delete border;

}

// Отрисовка тела

{

if (block)

SDL\_SetRenderDrawColor(renderer, Colors.element\_blocked.r, Colors.element\_blocked.g, Colors.element\_blocked.b, Colors.element\_blocked.a);

else

SDL\_SetRenderDrawColor(renderer, Colors.element\_background\_unfocus.r, Colors.element\_background\_unfocus.g, Colors.element\_background\_unfocus.b, Colors.element\_background\_unfocus.a);

}

// Отрисовка кнопки

SDL\_Rect dropButtonRect;

{

dropButtonRect = { sizes->x + sizes->w - 15, sizes->y + 6, 8, 8 };

SDL\_RenderFillRect(renderer, sizes);

}

renderLabel(mainLabel, sizes, LEFT\_ALIGN);

SDL\_RenderCopy(renderer, dropButton, NULL, &dropButtonRect);

if (show\_menu && !block) {

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

{

if (List.at(i)->Block()) {

SDL\_SetRenderDrawColor(renderer, Colors.element\_blocked.r, Colors.element\_blocked.g, Colors.element\_blocked.b, Colors.element\_blocked.a);

}

else {

if (List.at(i)->Hover())

SDL\_SetRenderDrawColor(renderer, Colors.element\_list\_hover.r, Colors.element\_list\_hover.g, Colors.element\_list\_hover.b, Colors.element\_list\_hover.a);

else

SDL\_SetRenderDrawColor(renderer, Colors.element\_background.r, Colors.element\_background.g, Colors.element\_background.b, Colors.element\_background.a);

}

SDL\_RenderFillRect(renderer, &renderRect);

renderLabel(List[i]->text, &renderRect, LEFT\_ALIGN);

renderRect.y += renderRect.h;

}

}

else {

SDL\_Rect clearRect = { renderRect.x, renderRect.y, renderRect.w, renderRect.h \* List.size() };

SDL\_SetRenderDrawColor(renderer, Colors.background.r, Colors.background.g, Colors.background.b, Colors.background.a);

SDL\_RenderFillRect(renderer, &clearRect);

}

SDL\_RenderPresent(renderer);

}

DropDownItem\* DropDownList::add(string text, int flag)

{

DropDownItem\* newItem = new DropDownItem;

newItem->text = text;

newItem->flag = flag;

List.push\_back(newItem);

return newItem;

}

void DropDownList::open(bool value)

{

show\_menu = value;

}

bool DropDownList::open()

{

return show\_menu;

}

void DropDownList::setValue(string value)

{

mainLabel = value;

}

string DropDownList::getValue()

{

return mainLabel;

}

int DropDownList::getFlag()

{

return flag\_select;

}

vector<DropDownItem\*> DropDownList::getItems()

{

return List;

}

void DropDownList::clear()

{

mainLabel = "";

}

DropDownList \* DropDownList::deleteItems()

{

List.clear();

return this;

}

int DropDownList::checkItemHover(int \_x, int \_y)

{

if (block || !display || !show\_menu)

return false;

SDL\_Point point = { \_x, \_y };

SDL\_Rect itemRect = \*itemSizes;

if (type == LEFT\_SIDE) {

itemRect.x = sizes->x + sizes->w;

itemRect.y = sizes->y;

}

else if (type == DOWN\_SIDE) {

itemRect.x = sizes->x;

itemRect.y = sizes->y + sizes->h;

}

int flag = 0;

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

List.at(i)->Hover(false);

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

bool hov = SDL\_PointInRect(&point, &itemRect);

itemRect.y += itemRect.h;

if (hov) {

if (!List.at(i)->Block()) {

flag = List.at(i)->flag;

List.at(i)->Hover(true);

}

}

}

flag\_select = flag;

return flag;

}

## Файл textField.h

#pragma once

#include "control.h"

class TextField : public Control {

private:

vector <string> lines;

int count\_lines;

public:

TextField(SDL\_Renderer\* \_renderer, SDL\_Rect\* \_sizes, string \_font, int \_font\_size) : Control(\_renderer, \_sizes, \_font, \_font\_size)

{

count\_lines = 0;

};

public:

void add(string newLine);

void render();

void renderText();

};

## Файл textField.cpp

#include "../Elements/TextField.h"

void TextField::add(string newLine)

{

if (lines.size() < 30) {

lines.push\_back(newLine);

count\_lines++;

}

else {

auto iter = lines.cbegin();

lines.erase(iter);

lines.push\_back(newLine);

count\_lines--;

}

}

void TextField::render()

{

SDL\_SetRenderDrawColor(renderer, Colors.element\_background.r, Colors.element\_background.g, Colors.element\_background.b, Colors.element\_background.a);

SDL\_RenderFillRect(renderer, sizes);

renderText();

}

void TextField::renderText()

{

SDL\_Rect rect = { sizes->x + 5, sizes->y + 10, sizes->w - 10, 15 };

rect.y += 16 \* (count\_lines - 1);

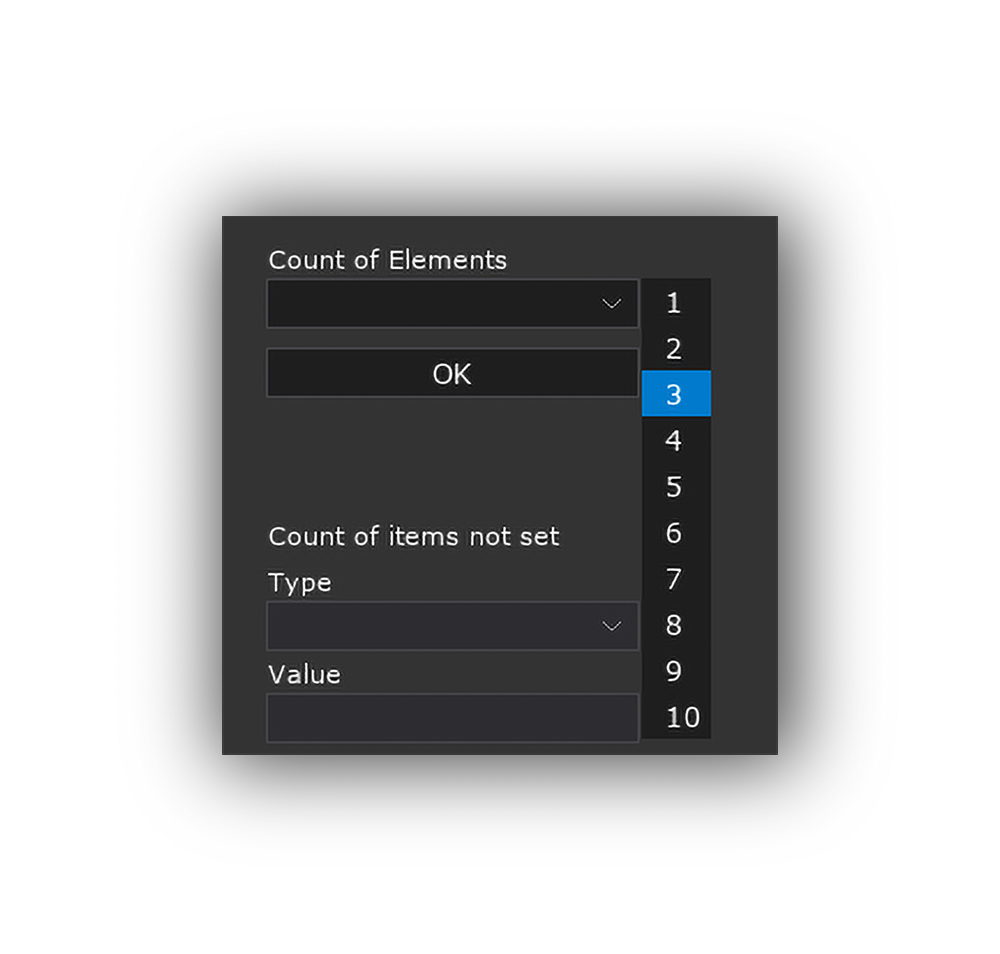
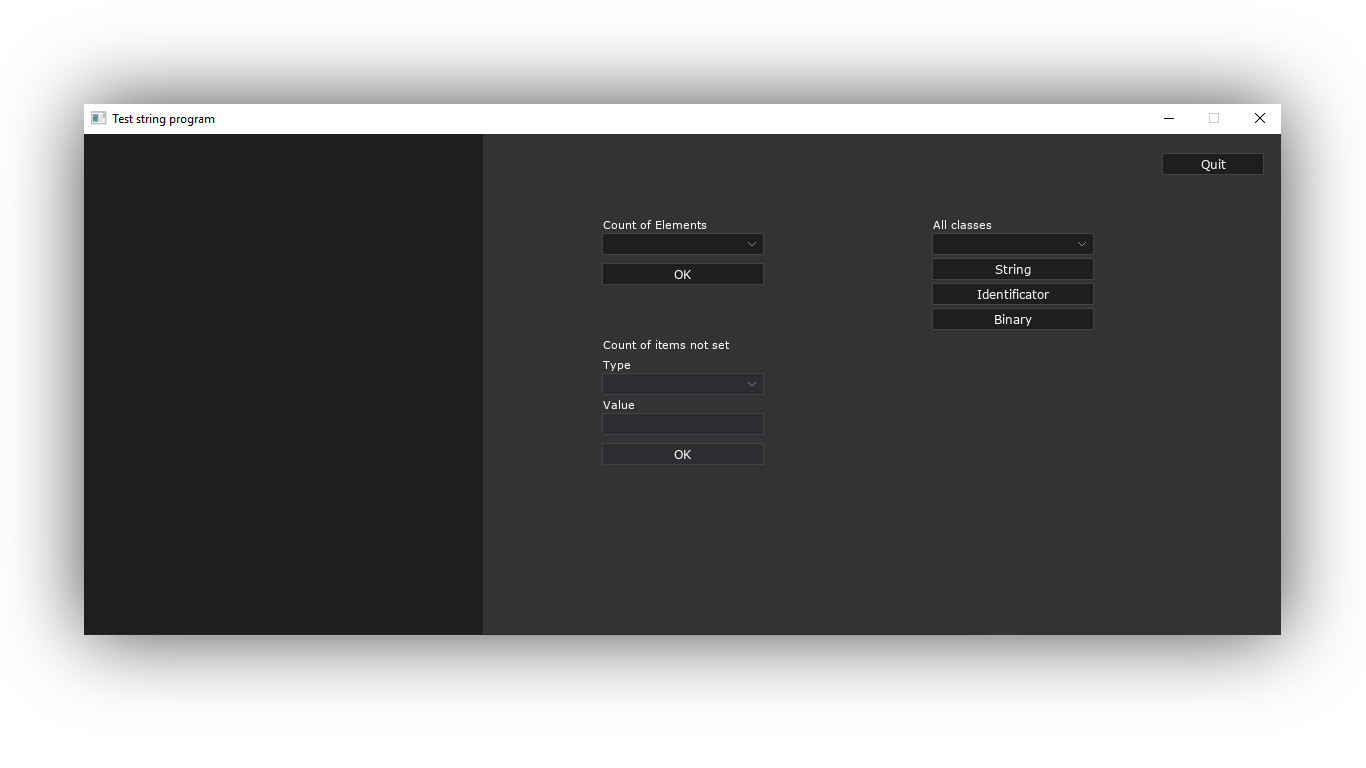
if (count\_lines > 0)

renderLabel(lines.at(count\_lines - 1), &rect, LEFT\_ALIGN);

SDL\_RenderPresent(renderer);

}

## 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)