|  |  |
| --- | --- |
| 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

|  |
| --- |
| Наследование. |
| 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](#_Toc12009546)

[ВВЕДЕНИЕ 4](#_Toc12009547)

[Техническое задание 6](#_Toc12009548)

[Классы строк 6](#_Toc12009549)

[Классы Интерфейса 10](#_Toc12009550)

[Обязательные правила отображения элементов: 11](#_Toc12009551)

[1. Реализация строк 12](#_Toc12009552)

[1.1. Базовый класс Строка 12](#_Toc12009553)

[Данные-члены класса, объявленные как protected: 12](#_Toc12009554)

[Методы класса \_строка\_: 12](#_Toc12009555)

[1.2. Производный класс. Строка идентификатор 13](#_Toc12009556)

[Методы класса ID\_String: 13](#_Toc12009557)

[1.3. Производный класс. Двоичная строка 13](#_Toc12009558)

[Методы класса BIN\_String: 13](#_Toc12009559)

[2. Реализация элементов управления 15](#_Toc12009560)

[2.1. Базовый класс Control 15](#_Toc12009561)

[Данные-члены класса, объявленные как protected: 15](#_Toc12009562)

[Методы класса Control: 16](#_Toc12009563)

[2.2. Производный класс Button 17](#_Toc12009564)

[Данные-члены класса, объявленные как private: 17](#_Toc12009565)

[Методы класса Button: 17](#_Toc12009566)

[2.3. Производный клаcc Input 18](#_Toc12009567)

[Данные-члены класса, объявленные как private: 18](#_Toc12009568)

[Методы класса Input: 18](#_Toc12009569)

[2.4. Производный класс ItemList 19](#_Toc12009570)

[Данные-члены класса, объявленные как private: 19](#_Toc12009571)

[Методы класса ItemList: 20](#_Toc12009572)

[2.5. Производный класс DropDownList 21](#_Toc12009573)

[Данные-члены класса, объявленные как private: 21](#_Toc12009574)

[Методы класса DropDownList: 22](#_Toc12009575)

[2.6. Производный класс TextField 25](#_Toc12009576)

[Данные-члены класса, объявленные как private: 25](#_Toc12009577)

[Методы класса TextField: 25](#_Toc12009578)

[2.7. Производный класс Label 25](#_Toc12009579)

[Данные-члены класса, объявленные как private: 25](#_Toc12009580)

[Методы класса Label: 26](#_Toc12009581)

[3. Реализация интерфейса 27](#_Toc12009582)

[Класс Interface 27](#_Toc12009583)

[Данные-члены класса, объявленные как private: 27](#_Toc12009584)

[Публичные методы класса Interface: 28](#_Toc12009585)

[Приватные методы класса Interface: 29](#_Toc12009586)

[Скриншоты программы 30](#_Toc12009587)

[Заключение 34](#_Toc12009588)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 35](#_Toc12009589)

# 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

## Классы строк

Разработать иерархию классов по следующей схеме:



c обязательной поддержкой заданных членов и методов.

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).

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».

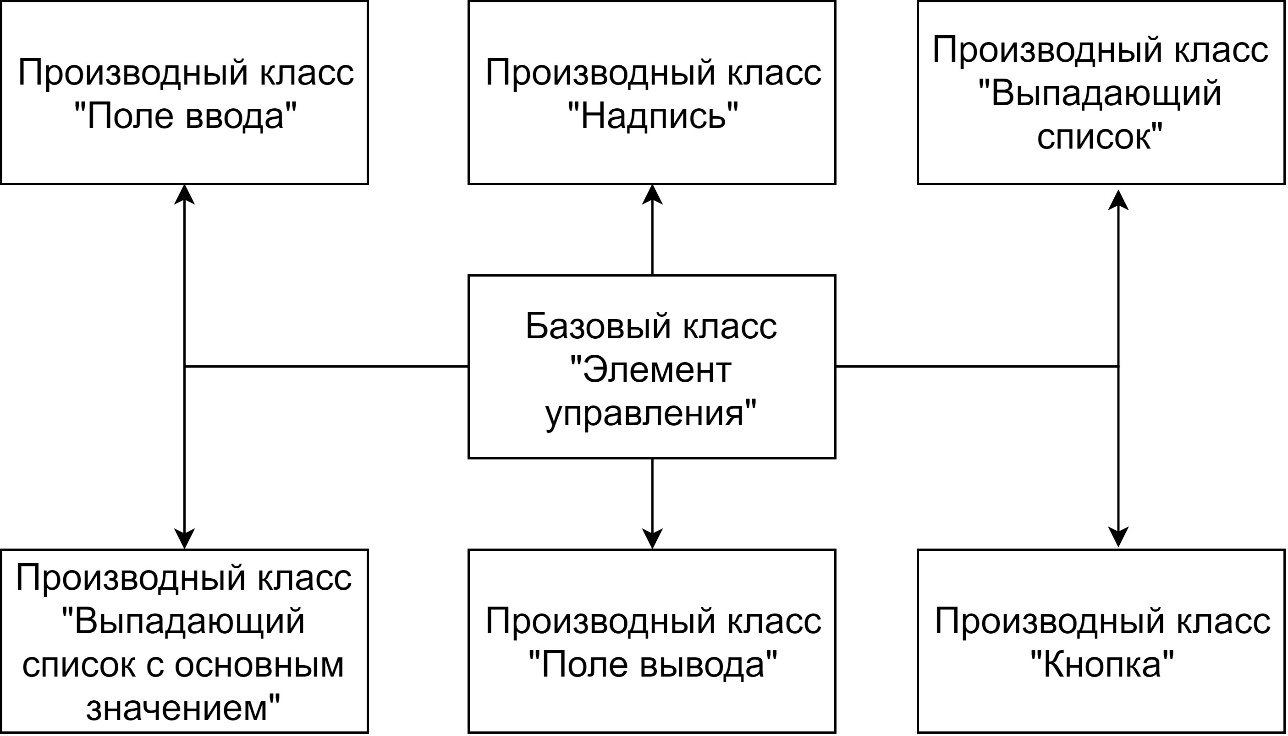
После выбора одного из пунктов меню предлагается выбрать один из методов списка всех обязательных методов (кроме конструкторов и деструкторов), связанных с выбранным подпунктом. Результат выводится на экран.

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.

## Классы Интерфейса

Разработать пользовательский интерфейс на основе графической библиотеки SDL версии 2, а также иерархию классов по следющей схеме:

для описания элементов управления с поддержкой заданных членов и методов, а также основной класс интерфейса для объединения классов строк и элементов управления.

Описать базовый класс «Элемент управления» и производные от него классы «Кнопка», «Поле ввода», «Надпись», «Выпадающий список», «Выпадающий список с основным значением» и «Поле вывода»

### Обязательные правила отображения элементов:

1. При нажатии на кнопку, она должна быть обозначена как нажимаемая, например, обводкой другого цвета. При отпускании кнопка должна принимать свой стандартный вид;
2. При фокусировке на поле ввода, оно должно обводится другим цветом, для обозначения фокуса;
3. При выборе элемента в выпадающем списке, он должен подсвечиваться, при наведении;
4. При клике на пространство вне тела и элементов списка, список должен автоматически скрываться;
5. При открытом выпадающем списке, открытие друго списка должно приводить к закрытию ранее открытого списка.

# 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, 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 Label:

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.

## 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;

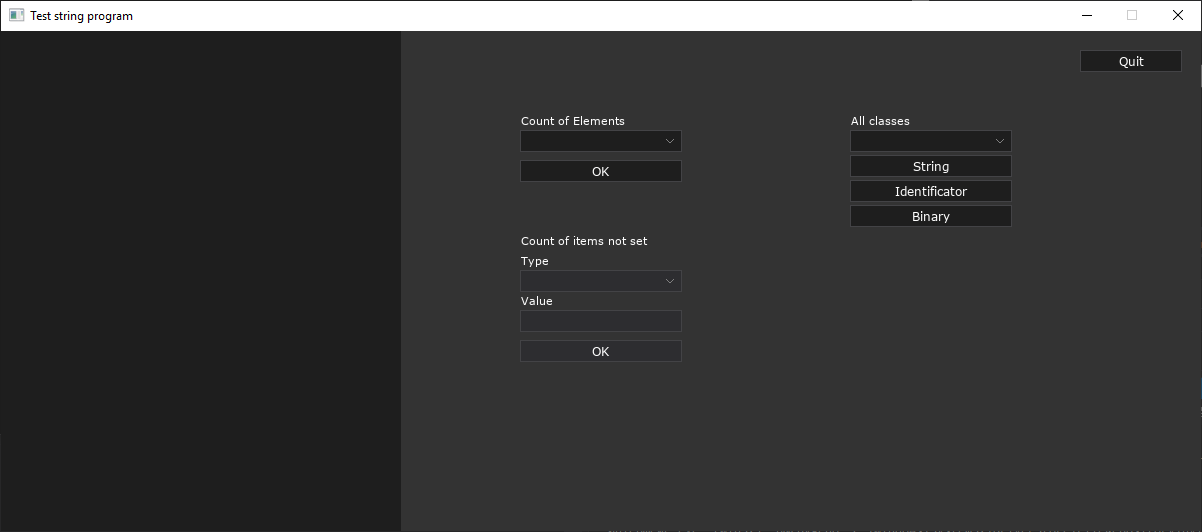
### Публичные методы класса Interface:

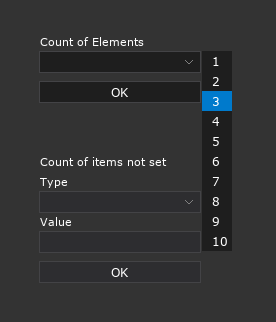
1. Interface() — базовый конструктор класса;
2. Interface(short width, short height) — конструктор принимающий два значения типа short, определяющие ширину и высоту окна программы;
3. ~Interface() — деструктор класса;
4. int onExecute() — основная функция, вызывающая все остальные функции.

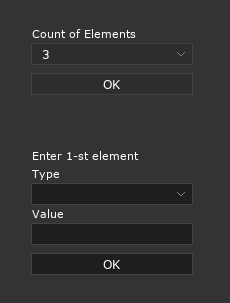
### Приватные методы класса Interface:

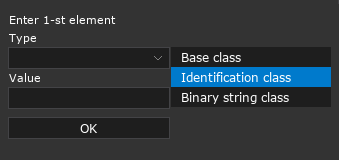
1. void render() — функция изначально отображающая все элементы интерфейса;
2. void onEvent() — основная функция для обработки событий SDL;
3. bool init() — функция предстартовой инициализации, таких компонентов, как SDL, SDL\_ttf, SDL\_image;
4. void mouseButtonDown(SDL\_Event\* event) — вспомогательная функция для обработки событий, когда кнопка мыши только нажата;
5. void mouseButtonUp(SDL\_Event\* event — вспомогательная функция для обработки событий, когда кнопка мыши была отпущена;
6. void mouseMotion(SDL\_Event\* event) — вспомогательная функция для обработки событий движения мыши;
7. void keyDown(SDL\_Event\* event) — вспомогательная функция для обработки событий нажатия клавиш на клавиатуре;
8. void quit() — функция, для выхода, очищающая всю выделенную память, в процессе рыботы программы.

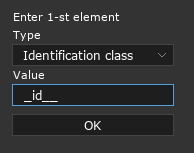
# Скриншоты программы

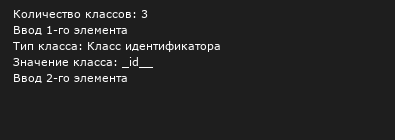
Рисунок 1 — Основной интерфейс программы

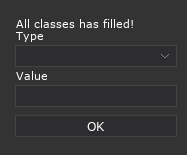
Рисунок 2 — выбор количества указателей на базовый класс

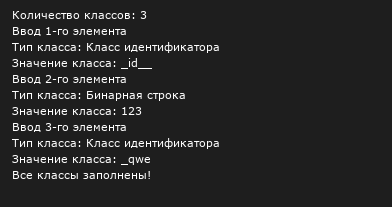
Рисунок 3 — количество элементов выбрано

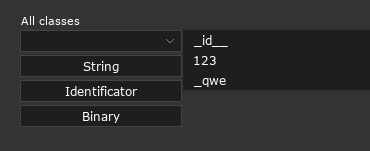
Рисунок 4 — выбор типа класса

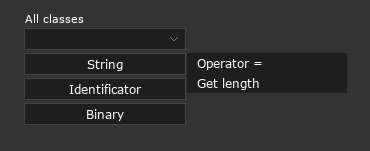
Рисунок 5 — ввод текста

Рисунок 6 — после ввода элементов вся информация о классе выводится в текстовый вывод

Рисунок 7 — после заполнения всех классов

Рисунок 8 — после заполнения всех классов, в выводе

Рисунок 9 — список классов

Рисунок 10 — операции базового класса

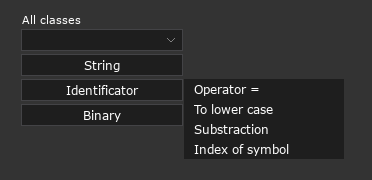
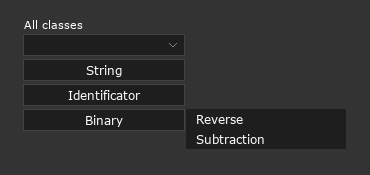


Рисунок 11 — операции класса идентификатора

Рисунок 12 — операции класса бинарной строки

# Заключение

Цель курсовой работы достигнута. Были созданы тестируемые классы: базовый класс Строка, Строка-идентификатор, Двоичная строка. Так же были реализованы классы управляющих элементов и класс интерфейса. Методы тестируемых классов были подключены к выпадающим спискам и полям ввода. Тестируемые классы и интерфейс работают корректно.

Из интерфейса был исключен пункт меню «Тестирование – Задать операнд», так как операнды вводились непосредственно при вызове конкретного метода тестируемого класса.

# 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.
4. Лафоре, Р. Объектно-ориентированное программирование в C++: Пер. с англ./ Р. Лафоре; Пер. А. Кузнецов, Пер. М. Назаров, Пер. В. Шрага. - 4-е изд. - СПб.: Питер, 2003. - 923 с.
5. Официальный сайт графической библиотеки SDL [Электронный ресурс] 2019. URL: <https://www.libsdl.org> (Дата обращения 01.05.2019)

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)