|  |  |
| --- | --- |
| voenmeh | 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 программная инженерия |
|  | шифр | наименование |
| Дисциплина | Программирование на языке высокого уровня | |

КУРСОВАЯ РАБОТА

на тему

Разработка приложения c использованием SDL2

|  |  |
| --- | --- |
| Выполнил студент группы | И506Б |
| Еремишин Н.Н. | |
| Фамилия И.О. |  |
| **РУКОВОДИТЕЛЬ** | |
| Магомедов И.Н. | |
| Фамилия И.О. Подпись | |
| Оценка |  |
| « » | 2021 г. |

САНКТ-ПЕТЕРБУРГ 2021 г.

Содержание

[ВВЕДЕНИЕ 3](#_bookmark0)

1. [Описание разработанной программы 4](#_bookmark1)
   1. [**Структура программы 4**](#_bookmark2)
   2. [**Иерархия классов**](#_bookmark3) **5**
   3. [**Структура меню**](#_bookmark4) **9**
   4. [**Внешние файлы 9**](#_bookmark5)
2. [Тестирование программы 10](#_bookmark6)

[ЗАКЛЮЧЕНИЕ 15](#_bookmark17)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_bookmark18) 16

[ПРИЛОЖЕНИЕ А 17](#_bookmark19)

# ВВЕДЕНИЕ

Цель работы - ознакомиться с библиотекой SDL2, научиться делать приложения, используя данную библиотеку.

В качестве курсового проекта я выбрал следующее приложение (игру):

Сверху падают платформы. Скорость падения со временем увеличивается. Задача игрока набрать наибольшее количество очков, прыгая вверх по платформам.

Также было необходимо реализовать меню и таблицу рекордов.

1. **ОПИСАНИЕ РАЗРАБОТАННОЙ ПРОГРАММЫ**
   * + 1. Структура программы

Программа состоит из следующих файлов:

character.cpp – класс Character, описывающий персонажа, которым управляет игрок.

character.h – заголовочный файл для character.cpp.

fileManager.h – класс FileWithRecords.

GameObject.h – класс GameObject. Данный класс описывает игровой объект – то есть все методы и функции, которые должен иметь каждый объект в игре, независимо от того, для чего данный объект предназначен.

SDL\_Kurs.cpp – файл, который содержит в себе реализацию сцен меню, игры и ввода имени.

settings.h – общие настройки. Включают в себя ширину и высоту экрана, размер плитки в тайлсете в пикселях и ускорение свободного падения.

tilemap.cpp – класс Tile, который описывает плитку и TilesRow, описывающий рад из плиток класса Tile.

tilemap.h – заголовочный файл для tilemap.cpp.

Time.cpp – класс Time, с помощью которого можно замерить время между кадрами. Замер данного времени требуется для обеспечения корректной работы для любого устройства. Если же использование такого замера не будет, то на компьютерах с разной вычислительной мощностью будет разная скорость движения игровых объектов: на слабых устройствах – медленная, а на мощных – быстрая.

Time.h - заголовочный файл для Time.cpp.

utils.cpp – функции для упрощения работы с SDL. Включает в себя следующие функции: упрощённый рендер текста, создание окна и рендера, загрузка и рендер текстуры.

utils.h – заголовочный файл для utils.cpp.

* + - 1. Иерархия классов

Для реализации задуманной игры было необходимо описать несколько классов. Иерархия этих классов представлена далее.

Иерархия класса GameObject и его наследников представлена на рисунке 1:

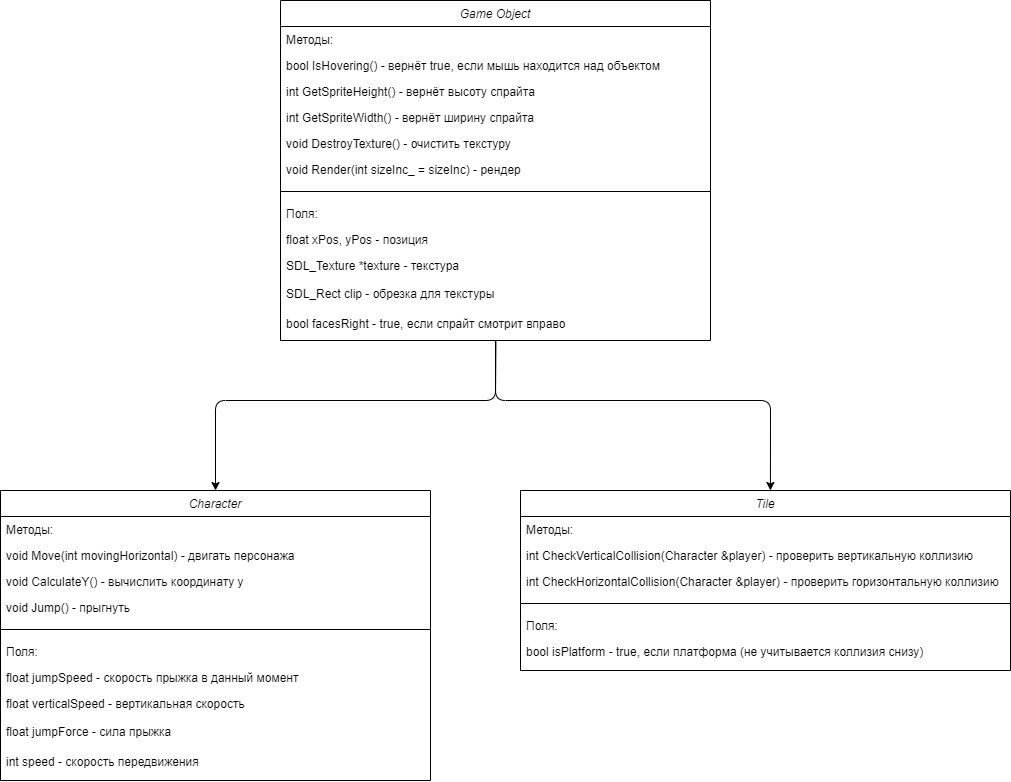


Рисунок 1 – Иерархия класса GameObject и его наследников

Класс GameObject описывает какой-либо объект в игре. От GameObject унаследованы два класса – Tile и Character. GameObject имеет следующие поля и методы:

Поля:

float xPos, yPos – позиция объекта на экране в координатах X и Y.

SDL\_Texture \*texture – текстура

Bool facecRight - показывает, смотрит ли спрайт вправо

Методы:

int GetSpriteHeight(), int GetSpriteWidth – эти методы находят высоту и ширину спрайта в пикселях соответсвенно

void DestroyTexture - очистка текстуры, то есть удаление изображения из оперативной памяти

void Render(int sizeInc\_ = sizeInc) – рендер объекта, принимает параметр – во сколько раз необходимо увеличить объект при рендере, по умолчанию значение sizeInc из файла с настройками

bool IsHovering() - проверка, находится ли мышь над объектом в данный момент

Character – это персонаж, в нашем случае тот, за которого играет пользователь. Character содержит следующие поля и методы:

Поля:

float verticalSpeed, jumpSpeed - вертикальная скорость и скорость прыжка соответственно в данный момент

float jumpForce - сила прыжка

int speed - скорость предвижения

Методы:

void Move(int movingHorizontal) – метод, который передвигает персонажа. Принимает направление движения (-1 – двигаться влево, 1 – двигаться вправо)

void CalculateY() - вычисление координаты y

void Jump() - Совершить прыжок

Tile – плитка, падающая сверху. По этим плиткам будет прыгать персонаж. Tile имеет следующие поля и методы:

Поля:

bool isPlatform - является ли плитка платформой. Если да, то она имеет коллайдер снизу.

Методы:

int CheckHorizontalCollision(Character &player) - проверка горизонтальной коллизии с игроком

int CheckVerticalCollision(Character &player) - проверка вертикальной коллизии с игроком

Также присутствуют следующие классы: FileWithRecords, Time и TilesRow. Они не унаследованы от каких-либо классов, а также от них ничего не наследуется. Эти классы изображены на рисунке 2.

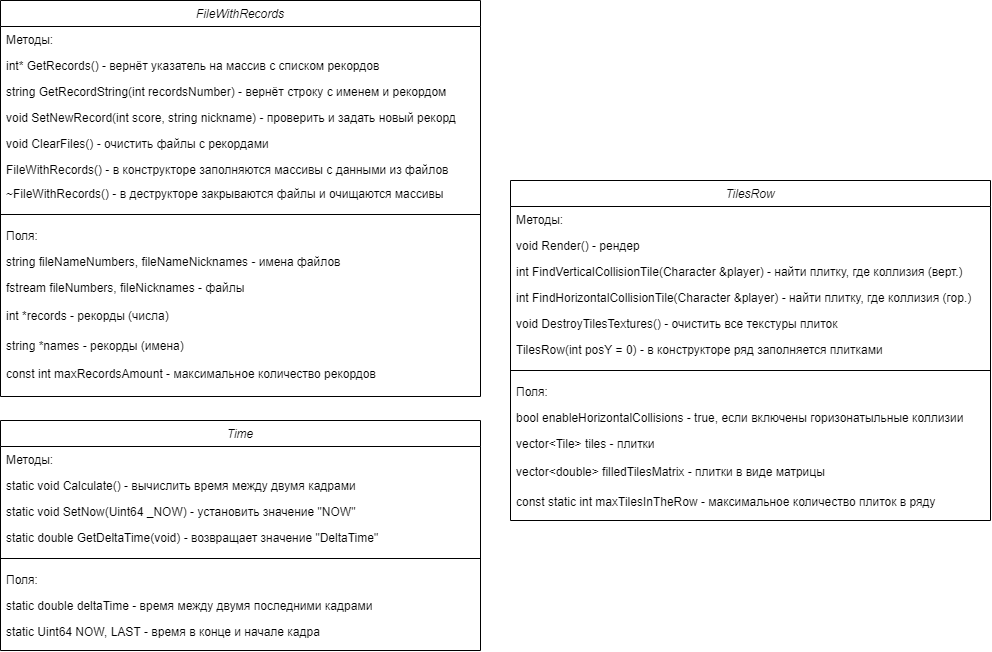


Рисунок 2 – Классы FileWithRecords, Time и TilesRow

FileWithRecrods – организовывает запись рекордов в файл.

Поля:

string fileNameNumbers, fileNameNicknames - имена файлов. Один файл содержит рекорды (числа), а второй – имена игроков, которые установили данные рекорды.

fstream fileNumbers, fileNicknames – файлы

int \*records – массив с рекордами (числа)

string \*names - массив с рекордами (имена)

const int maxRecordsAmount - максимальное количество рекордов, которое будет хранится в файле

Методы:

void ClearFiles() - очистить файл с рекордами и файл с именами

void SetNewRecord(int score, string nickname) – проверить, является ли результат одним из лучших и если да, то задать новый рекорд

string GetRecordString(int recordsNumber) - вернёт строку с именем и рекордом вида ‘имя - рекорд”

Time – методы данного замеряют время между двумя последними кадрами. Это нужно, чтобы независимо от мощности компьютера, движение объектов было плавным и одинаковым.

Поля:

static double deltaTime – временной промежуток между двумя последними кадрами

static Uint64 NOW, LAST - время в конце и начале кадра

Методы:

static void Calculate() - вычислить время между двумя последними кадрами

TilesRow – ряд из плиток Tile.

Поля:

bool enableHorizontalCollisions - true, если включены горизонатыльные коллизии

vector<Tile> tiles - массив с плитками

vector<double> filledTilesMatrix - массив с матрицей плиток (0 – нет плитки, 1 – есть плитка)

const static int maxTilesInTheRow - максимальное количество плиток в ряду

Методы:

void Render() - рендер всего ряда плиток

int FindVerticalCollisionTile(Character &player) - найти плитку, где коллизия (верт.)

int FindHorizontalCollisionTile(Character &player) - найти плитку, где коллизия (гор.)

void DestroyTilesTextures() - очистить все текстуры плиток

* + - 1. Структура меню

При запуске игры пользователя встречает экран, где ему необходимо ввести своё имя. Далее идёт сцена с меню, где можно посмотреть управление, таблицу рекордов и начать игру.

* + - 1. Внешние файлы

Для хранения информации используется два бинарных файла «records.bin» и «recordsNames.bin». В данных файлах хранятся записи о рекордах. В одном из файлов хранятся имена, а в другом – числа.

# 2. ТЕСТИРОВАНИЕ ПРОГРАММЫ

2.1 Ввод имени пользователя. Пример приведен на рисунке 1:

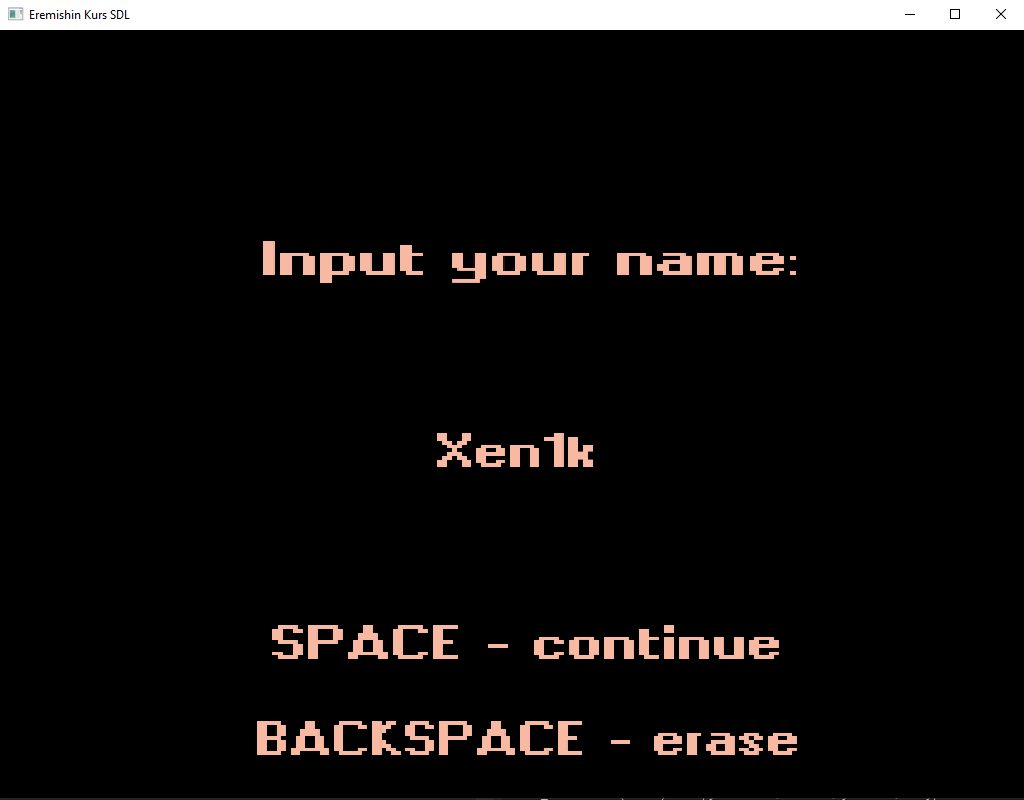


Рисунок 1 – Ввод имени пользователя

2.2 Главное меню.

Слева – таблица рекордов. Пробел – начать игру, esc – выйти из игры. Нажав на значок “i” в правом верхнем углу, можно посмотреть управление. Пример приведен на рисунке 2:

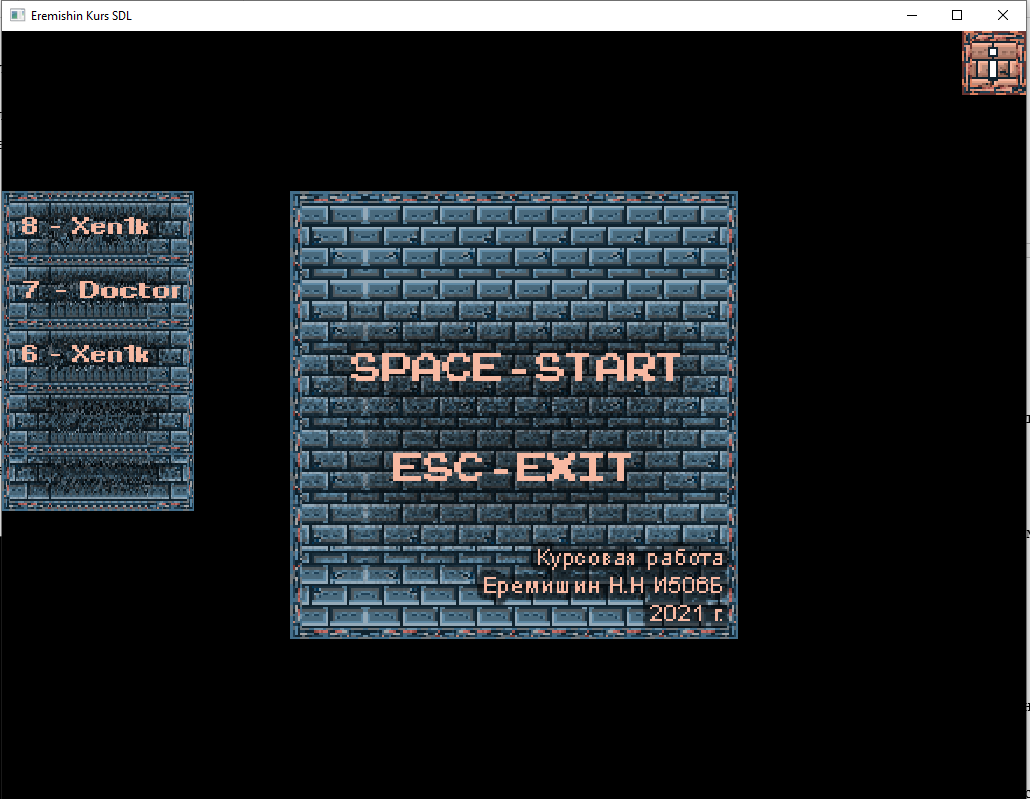


Рисунок 2 – Главное меню

2.3 Управление. Пример приведен на рисунке 3:

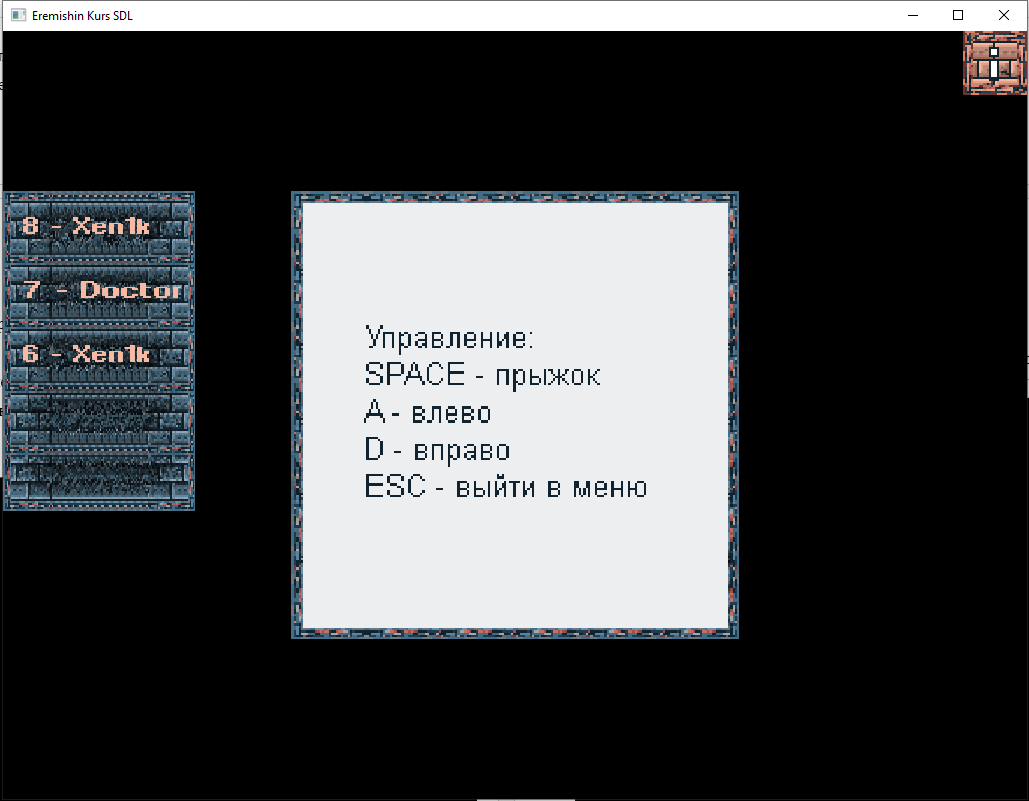


Рисунок 3 – Управление

2.4 Игровой процесс: Пример приведен на рисунках 4 и 5:

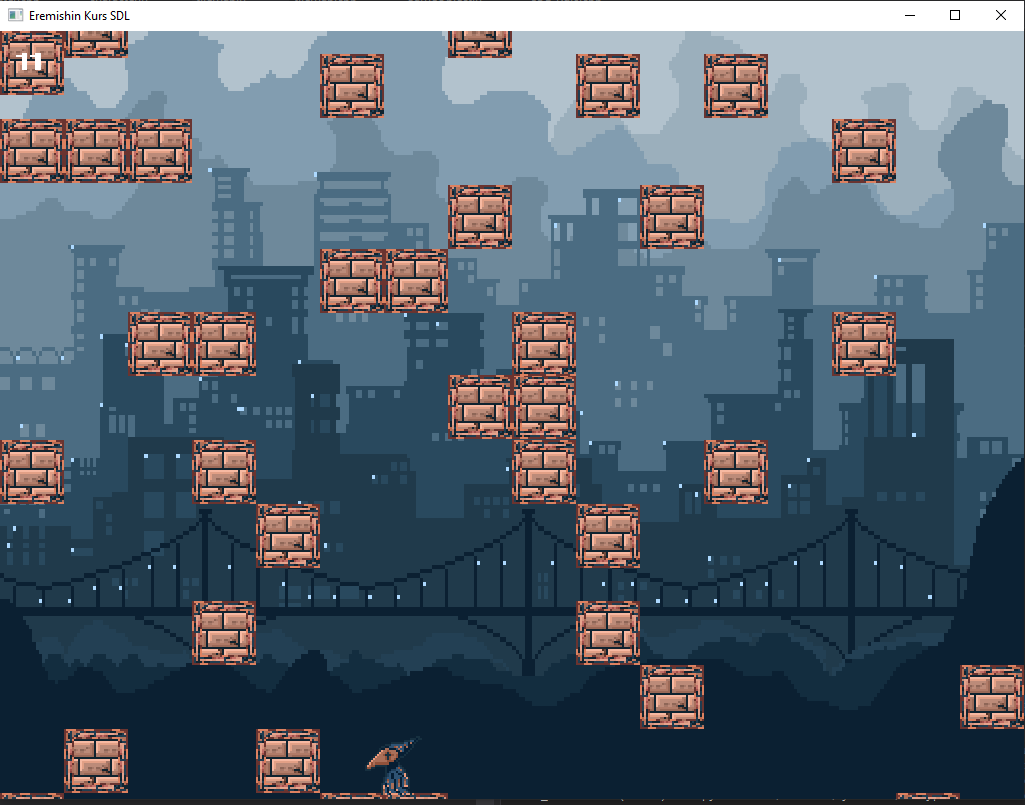


Рисунок 4 – Игровой процесс



Рисунок 5 – Игровой процесс

2.6 Экран, появляющийся после проигрыша. Нажав на R, можно перезапустить игру, а на M – перейти в меню. Пример приведен на рисунке 6.



Рисунок 6 – Экран, появляющийся после проигрыша

# ЗАКЛЮЧЕНИЕ

В рамках курсовой работы было разработано и протестировано приложение (игра) с использованием библиотеки SDL2. В процессе выполнения курсовой работы было изучено то, как пользоваться данной библиотекой, (например - создание окна, рендер спрайтов в окне).

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. — СПб.: Питер, 2003 —461 с: ил. ISBN 5-94723-568-4
2. Работа с бинарными файлами [Электронный ресурс] URL: https://proginfo.ru/binary-file/ (дата обращения 18.05.2021).
3. Страуструп. Язык программирования С++ [Электронный ресурс] <http://www.8361.ru/6sem/books/Straustrup-Yazyk_programmirovaniya_c.pdf> (дата обращения 18.05.2021).
4. ЯЗЫК Си / Б.В. Керниган, Д.М. Ричи. [Электронный ресурс] URL: https://[www.packoplock.se/media/wysiwyg/1\_K\_R.pdf](http://www.packoplock.se/media/wysiwyg/1_K_R.pdf) / (дата обращения 18.05.2021).
5. Алгоритмы обработки массивов и вспомогательные алгоритмы: учебное пособие / А.Н. Гущин [и др.]; под. ред. И.К. Раковой; Балт. гос. техн. ун- т. – СПб., 2016. – 210 с. ISBN 978-5-85546-984-4

# ПРИЛОЖЕНИЕ А

Исходные тексты программ представлены на прилагаемом электронном носителе.