МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

« МИРЭА Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Вычислительной техники

УЧЕБНОЕ ЗАДАНИЕ

по дисциплине

« Объектно-ориентированное программирование»

Наименование задачи:

« Задача 3_1_3 »

С тудент группы	ИКБО-27-21	Родионов А.А.
Руководитель практики	Ассистент	Морозов В.А.
Работа представлена	«___»_____ 2022 г.	_____
		(подпись студента)
Оценка		_____
		(подпись руководителя)

Москва 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Постановка задачи

Создать класс для объекта стек. Стек хранит целые числа. Имеет характеристики: наименование (строка, не более 10 символов) и размер (целое). Размер стека больше или равно 1. Функционал стека:
- добавить элемент и вернуть признак успеха (логическое);
- извлечь элемент (НЕ вывести!) и вернуть признак успеха (логическое);
- получить имя стека (строка);
- получить размер стека (целое);
- получить текущее количество элементов в стеке (целое).

В классе определить параметризированный конструктор, которому передается имя стека и размер. При переполнении стека очередной элемент не добавлять и определяется соответствующий признак успеха.

В основной программе реализовать алгоритм:

1. Ввести имя и размер для первого стека.
2. Создать объект первого стека.
3. Ввести имя и размер для второго стека.
4. Создать объект второго стека.
5. В цикле:
    5.1. Считывать очередное значение элемента.
    5.2. Добавлять элемент в первый стек, при переполнении завершить цикл.
    5.3. Добавлять элемент во второй стек, при переполнении завершить цикл.
6. Построчно вывести содержимое стеков.

Описание входных данных

Первая строка:

«имя стека 1»«размер стека»

Вторая строка:

«имя стека 2»«размер стека»

Третья строка:

Последовательность целых чисел, разделенных пробелами, в количестве не менее чем размер одного из стеков + 1.

Описание выходных данных

Первая строка:

«имя стека 1»«размер»

Вторая строка:

«имя стека 2»«размер»

Третья строка:

«имя стека 1»«имя стека 2»

Каждое имя стека в третьей строке занимает поле длины 15 позиции и прижата к левому краю.

Четвертая строка и далее построчно, вывести все элементы стеков: «значение элемента стека 1»«значение элемента стека 2»

Вывод значений элементов стеков производиться последовательным извлечением.

Каждое значение занимает поле из 15 позиции и прижата к правому краю.

Метод решения

Для выполнения задачи потребовалось:

-Переменная типа integer

-Объект ввода\вывода потока данных cin/cout(iostream)

-Условный оператор if

-Цикл for, while

-Объект st1 класса Stack

-Объект st2 класса Stack

Класс Stack :

Поля :

Модификатор доступа public :

Наименование : Stack

Параметры : string name, int capacity

Функционал : конструктор

Наименование : ~Stack

Функционал : деструктор

Наименование : add_element

Возвращаемый тип : void

Параметры : int element, bool *success

Функционал : добавляет элемент в стек

Наименование : get_element

Возвращаемый тип : void

Параметры : int *element, bool *success

Функционал : достает элемент из стека

Наименование : get_name

Возвращаемый тип : string

Функционал : возвращает имя стека

Наименование : get_capacirt

Возвращаемый тип : int

Функционал : возвращает размер стека

Наименование : get_size

Возвращаемый тип : int

Функционал : возвращает количество элементов в стеке

Описание алгоритма

Согласно этапам разработки, после определения необходимого инструментария в разделе «Метод», составляются подробные описания алгоритмов для методов классов и функций.

Конструктор класса: Stack

Модификатор доступа: public

Функционал: Конструктор

Параметры: string name, int capacity

Алгоритм конструктора представлен в таблице 1.

Таблица 1. Алгоритм конструктора класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1		this->name = name	2	
2		this->capacity = capacity	3	
3		this->last_index = 0	4	
4		this->stack = new int[capacity]	∅	

Деструктор класса: Stack

Модификатор доступа: public

Функционал: Дестркуктор

Параметры: нет

Алгоритм деструктора представлен в таблице 2.

Таблица 2. Алгоритм деструктора класса Stack

№	Предикат	Действия	№ перехода	Комментарий

1		delete this->stack	∅	

Класс объекта: Stack

Модификатор доступа: public

Метод: add_element

Функционал: Добавляет элемент в стек

Параметры: int element, bool *success

Возвращаемое значение: void

Алгоритм метода представлен в таблице 3.

Таблица 3. Алгоритм метода add_element класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1	last_index < this->get_capacity()		2	
			7	
2	last_index < 0		3	
			4	
3		last_index = 0	4	
4		this->stack[this->last_index] = element	5	
5		this->last_index++	6	
6		*success = true	∅	
7		*success = false	∅	

Класс объекта: Stack

Модификатор доступа: public

Метод: get_element

Функционал: Достаёт элемент из стека

Параметры: int *element, bool * success

Возвращаемое значение: void

Алгоритм метода представлен в таблице 4.

Таблица 4. Алгоритм метода get_element класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1	last_index - 1 >= 0		2	
			5	
2		last_index--	3	
3		*element = this->stack[this->last_index]	4	
4		*success = true	∅	
5		*success = false	∅	

Класс объекта: Stack

Модификатор доступа: public

Метод: get_name

Функционал: Возвращает имя стека

Параметры: нет

Возвращаемое значение: string

Алгоритм метода представлен в таблице 5.

Таблица 5. Алгоритм метода get_name класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1		return this->name	∅	

Класс объекта: Stack

Модификатор доступа: public

Метод: get_capacity

Функционал: Возвращает размер стека

Параметры: нет

Возвращаемое значение: int

Алгоритм метода представлен в таблице 6.

Таблица 6. Алгоритм метода get_capacity класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1		return this->capacity	∅	

Класс объекта: Stack

Модификатор доступа: public

Метод: get_size

Функционал: Возвращает количество элементов в стеке

Параметры: нет

Возвращаемое значение: int

Алгоритм метода представлен в таблице 7.

Таблица 7. Алгоритм метода get_size класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1		return this->last_index	∅	

Функция: main

Функционал: Главная функция программы

Параметры: нет

Возвращаемое значение: void

Алгоритм функции представлен в таблице 8.

Таблица 8. Алгоритм функции main

№	Предикат	Действия	№ перехода	Комментарий
1		Инициализация переменнной типа string string name	2	
2		Инициализация переменной типа int int capacity	3	
3		Ввод значений переменных name, capacity	4	
4		Инициализация объекта типа Stack Stack st1(name, capacity)	5	
5		Ввод значений переменных name, capacity	6	
6		Инициализация объекта типа Stack Stack st2(name, capacity)	7	
7		Инициализация переменной типа bool bool success	8	
8		while(true)	9	
9		Инициализация переменной типа int int element	10	

10		Ввод значения переменной element	11	
11			∅	

Блок-схема алгоритма

Представим описание алгоритмов в графическом виде на рисунках ниже.

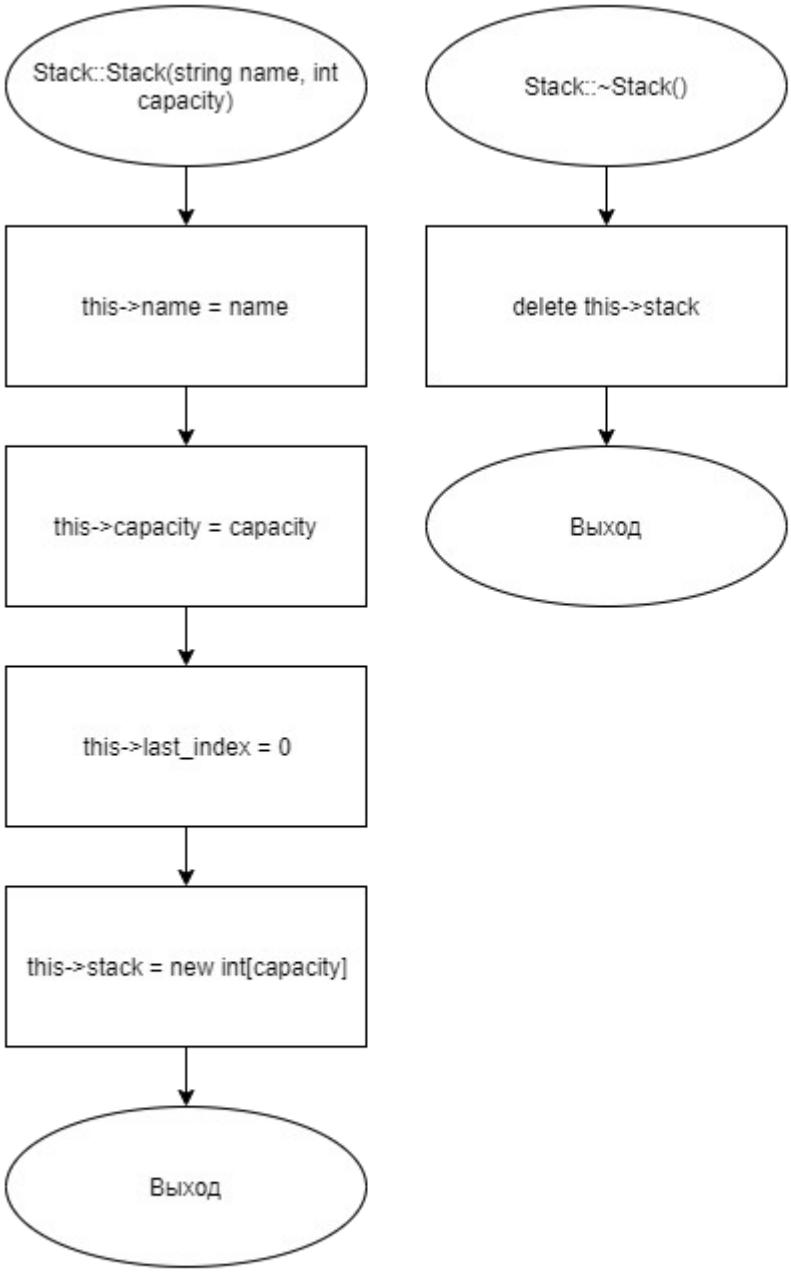

Рис. 1. Блок-схема алгоритма.

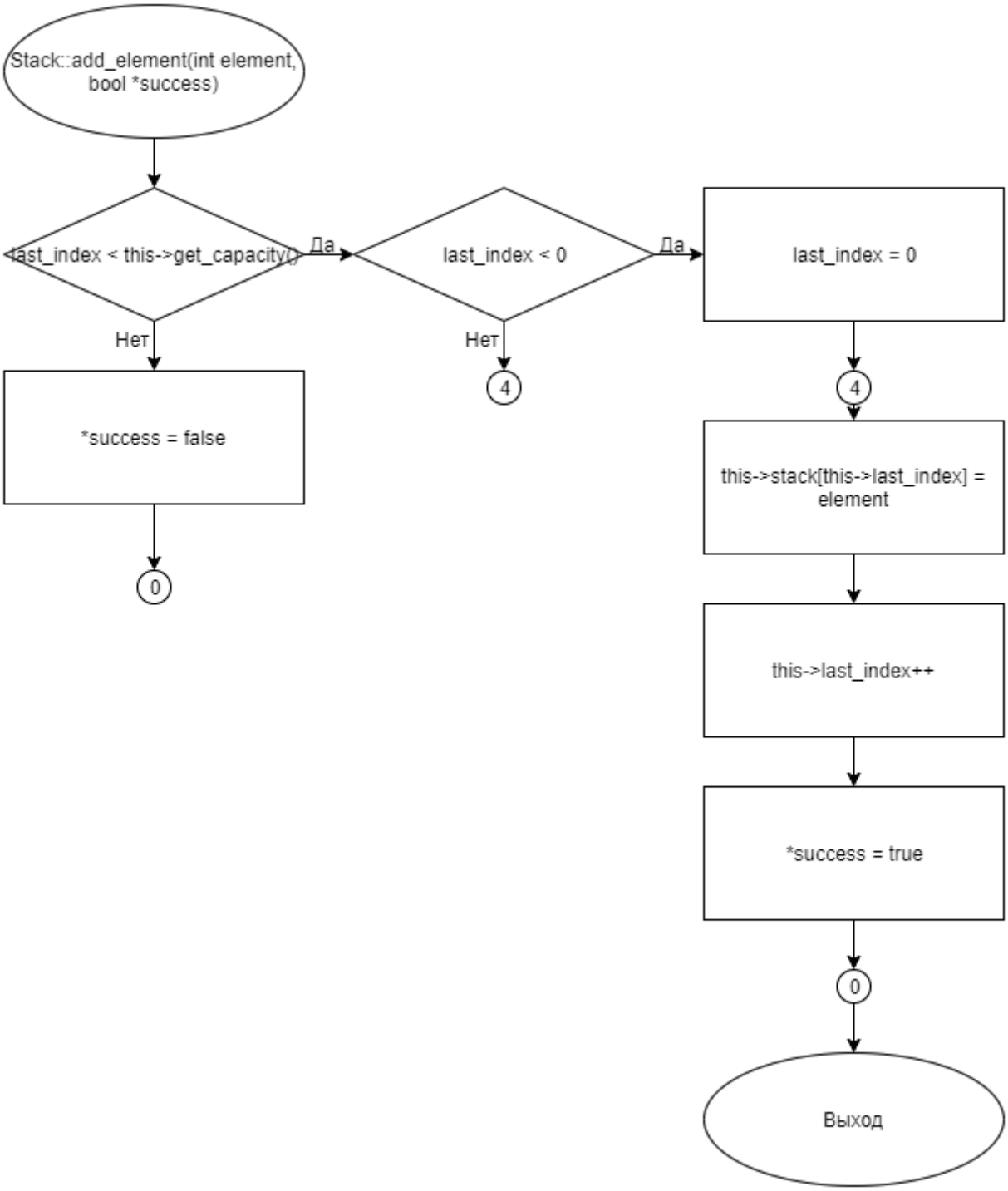

Рис. 2. Блок-схема алгоритма.

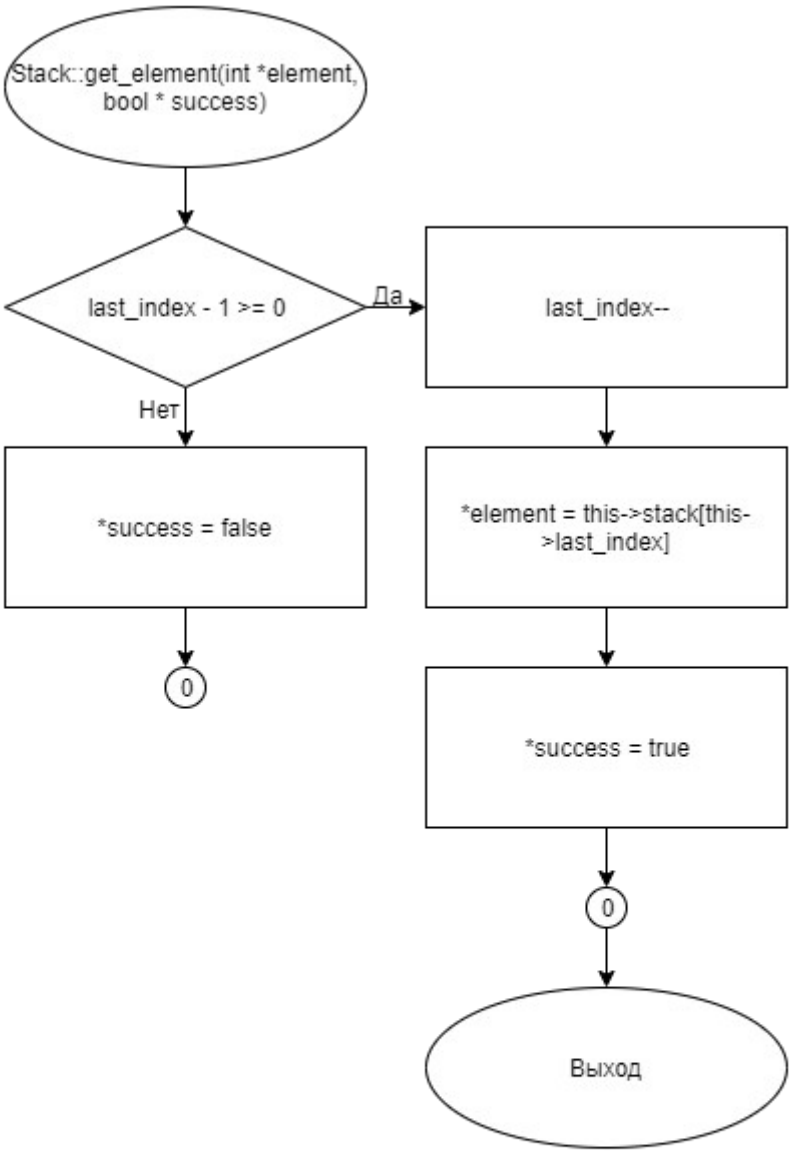

Рис. 3. Блок-схема алгоритма.

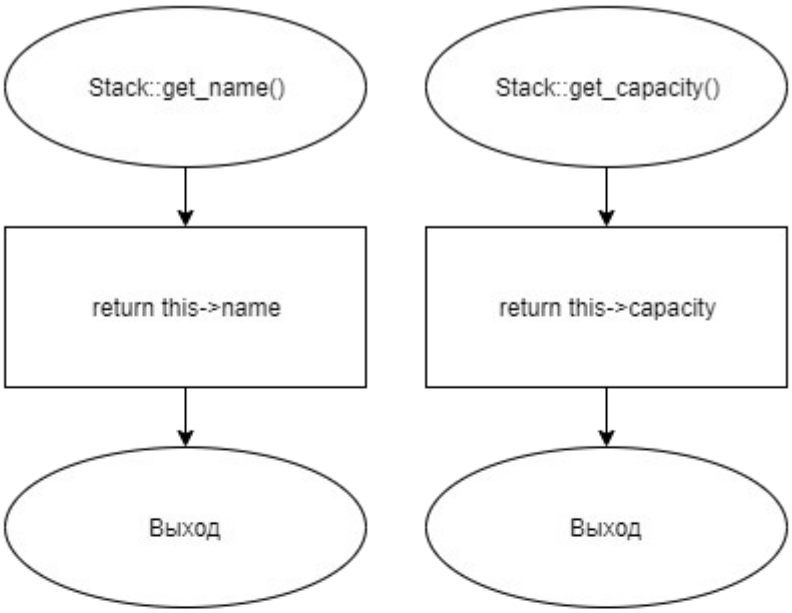

Рис. 4. Блок-схема алгоритма.

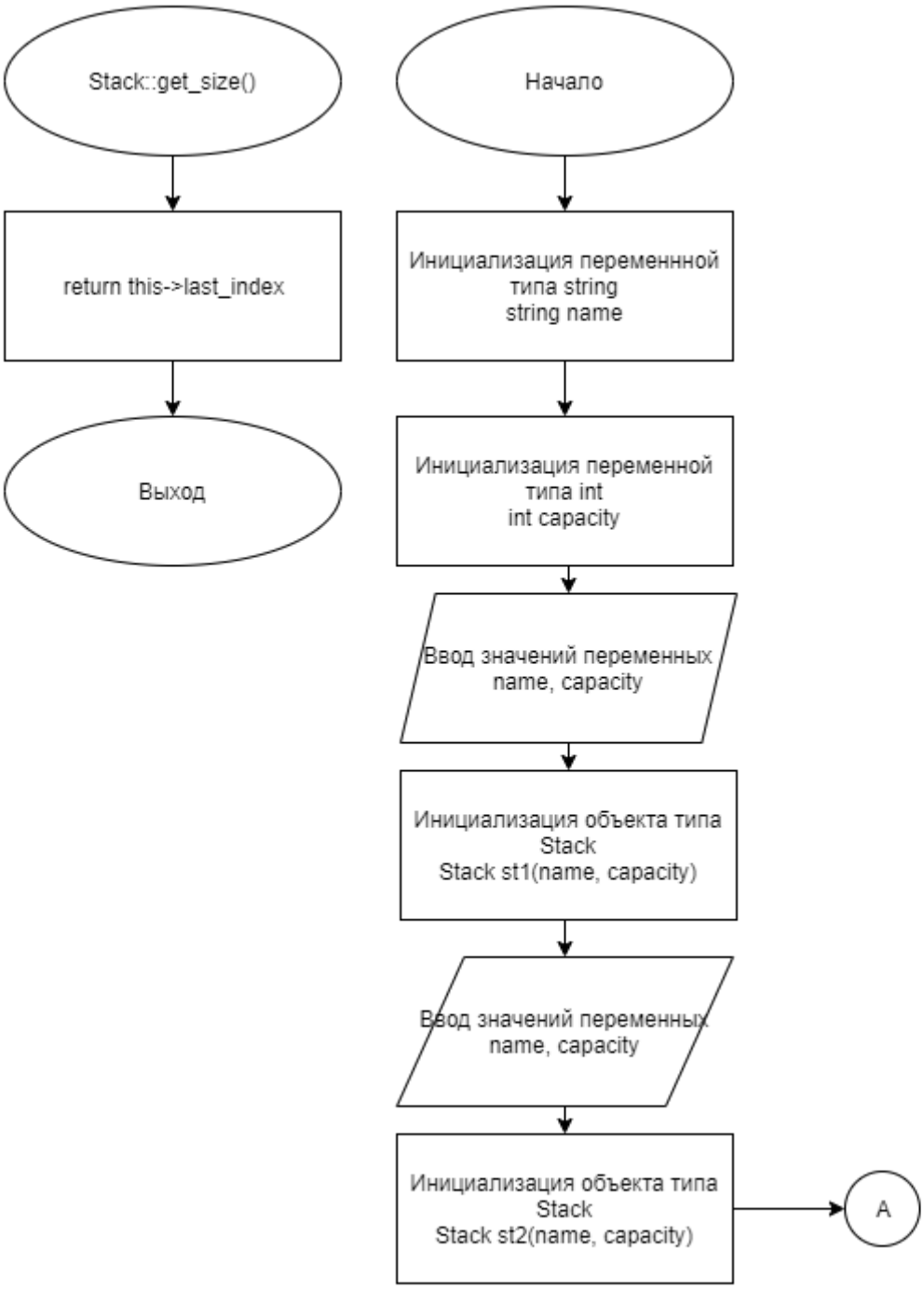

Рис. 5. Блок-схема алгоритма.

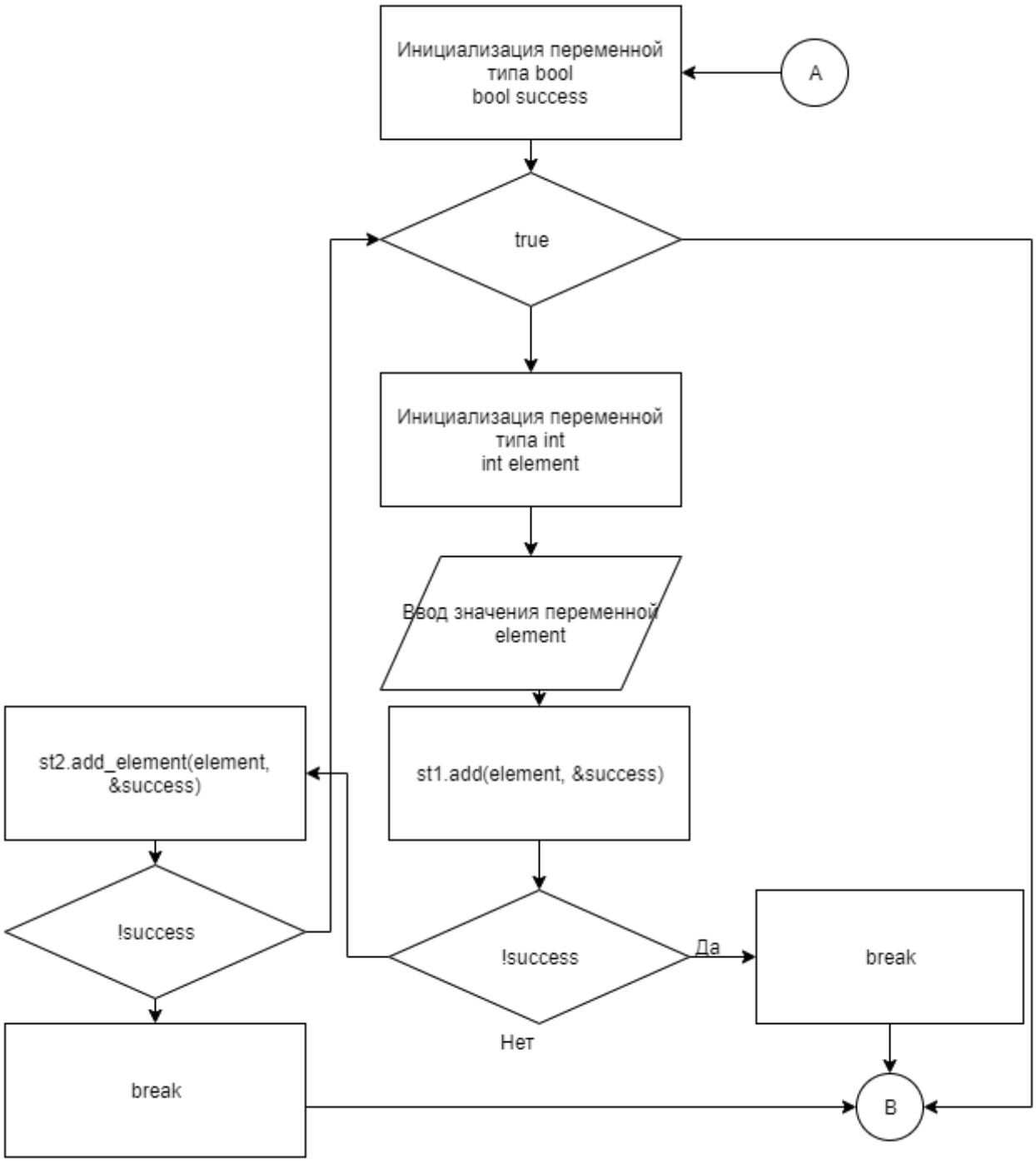

Рис. 6. Блок-схема алгоритма.

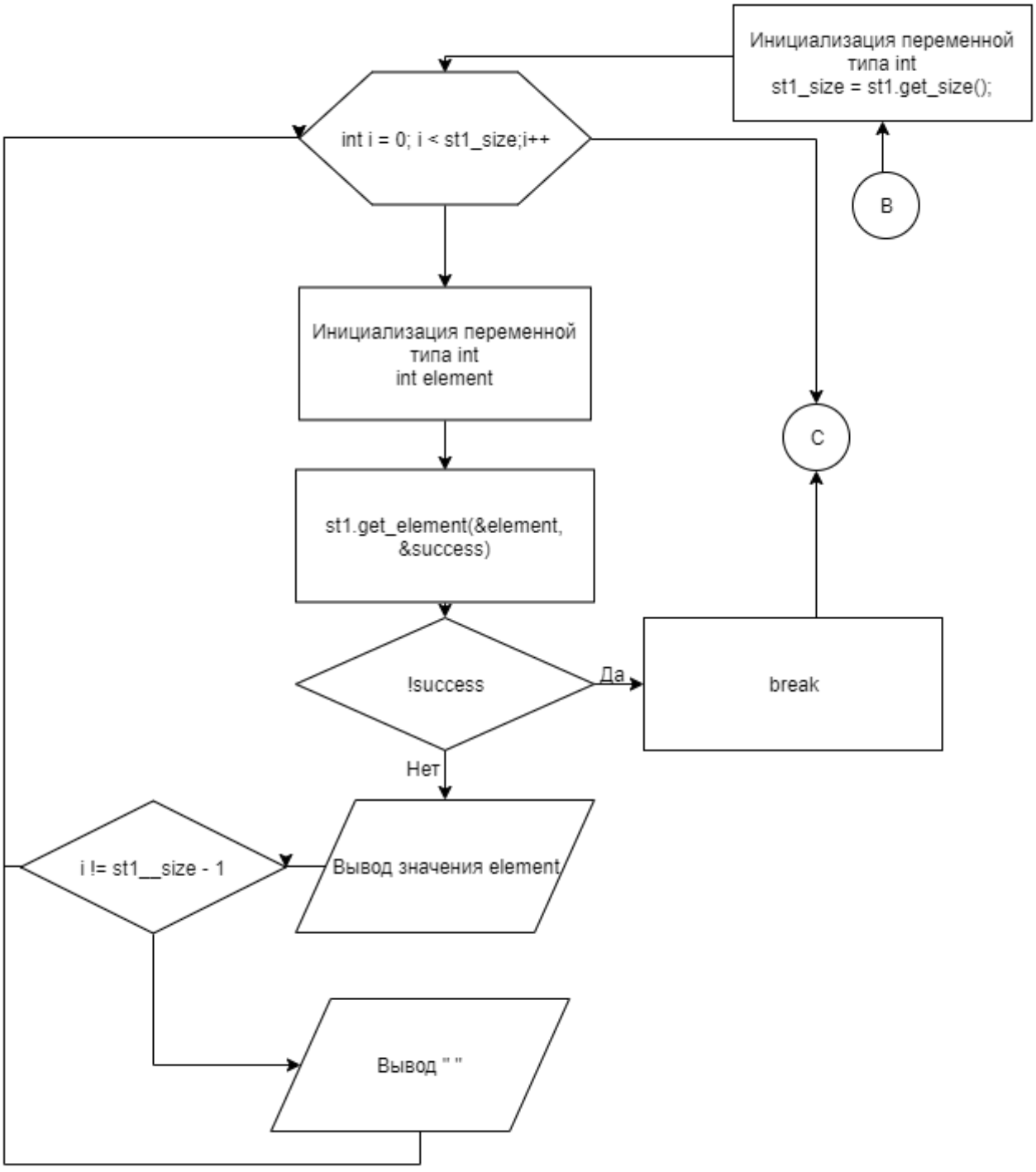

Рис. 7. Блок-схема алгоритма.

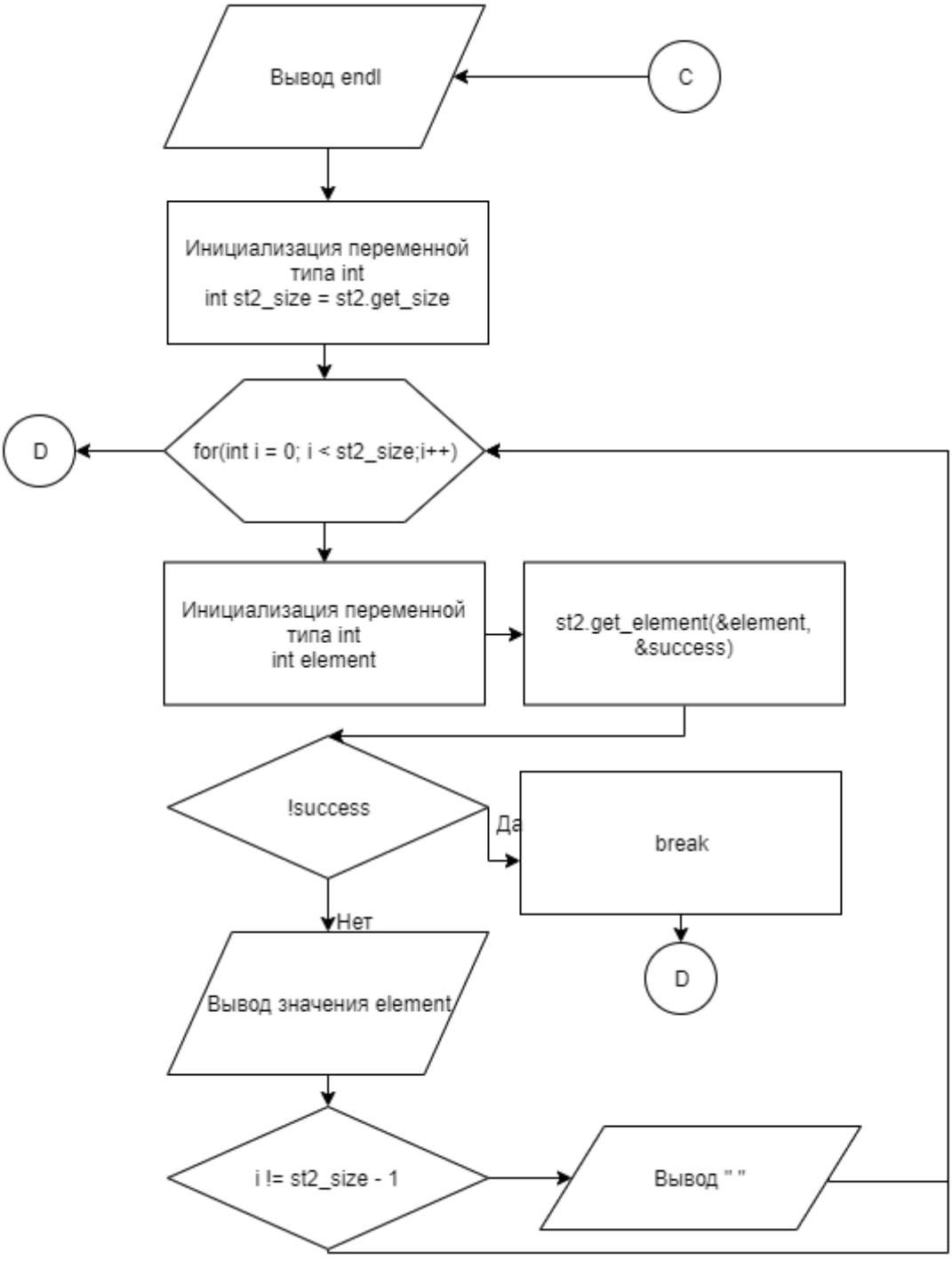

Рис. 8. Блок-схема алгоритма.

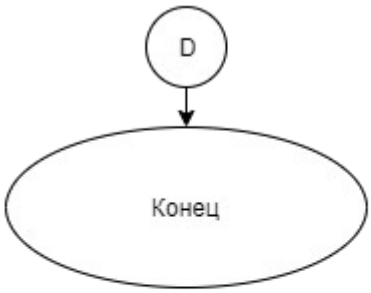

Рис. 9. Блок-схема алгоритма.

Код программы

Программная реализация алгоритмов для решения задачи представлена ниже.

Файл main.cpp

```
#include <iostream>
#include <string>
#include "Stack.h"

using namespace std;

void print(string a, string b){
        string strip = "";
        if(a != ""){
        for(int i = 0; i < 15;i++){
                if(i < 15 - a.length()){
                        strip += " ";
                }else{
                        strip += a[i - (15 - a.length())];
                }
        }
        }
        if(b != ""){
        for(int i = 0; i < 15;i++){
                if(i < 15 - b.length()){
                        strip += " ";
                }else{
                        strip += b[i - (15 - b.length())];
                }
        }
        }

        cout << strip;
}

int main()
{
        string name;
        int capacity;
        cin >> name >> capacity;
        Stack st1(name, capacity);
        cin >> name >> capacity;
        Stack st2(name, capacity);
        bool success;
        while(true){
                int element;
                cin >> element;
                st1.add_element(element, &success);
```

```
                if(!success)break;
                //cout << "add to st1\n";
                st2.add_element(element, &success);
                if(!success)break;
                //cout << "add to st2\n";
        }
        cout << st1.get_name() << " " << st1.get_capacity() << endl;
        cout << st2.get_name() << " " << st2.get_capacity() << endl;

        string st1_n,st2_n;
        string names = "";

        st1_n = st1.get_name();
        st2_n = st2.get_name();

        for(int i = 0; i < 15;i++){
                if(i < (st1_n.length())){
                        names += st1_n[i];
                }else{
                        names += " ";
                }
        }
        for(int i = 0; i < 15;i++){
                if(i < (st2_n.length())){
                        names += st2_n[i];
                }else{
                        names += " ";
                }
        }

        cout << names << endl;

        int max_l = (st1.get_size() > st2.get_size())? st1.get_size() :
st2.get_size();

        for(int i = 0; i < max_l; i++){
                int buffer;
                int el1,el2;
                string els1,els2;
                st1.get_element(&el1,&success);
                if(success)els1 = to_string(el1);
                else els1 = "";
                st2.get_element(&el2,&success);
                if(success)els2 = to_string(el2);
                else els2 = "";

                //cout << els1 << " " << els2 << endl;

                print(els1,els2);

                if(i != max_l - 1){
                        cout << "\n";
                }
        }

        return(0);
}
```

Файл Stack.cpp

```
#include<iostream>
#include<string>
#include "Stack.h"

using namespace std;

Stack::Stack(string name, int capacity){
        this->name = name;
        this->capacity = capacity;
        this->last_index = 0;
        this->stack = new int[capacity];
}

Stack::~Stack(){
        delete this->stack;
}

void Stack::add_element(int element, bool*success){
        if(last_index < this->get_capacity()){
                if(last_index < 0)last_index = 0;
                this->stack[this->last_index] = element;
                this->last_index++;
                *success = true;
        }else *success = false;
}

void Stack::get_element(int* element, bool*success){
        if(last_index - 1 >= 0){
                last_index--;
                *element = this->stack[this->last_index];
                *success = true;
        }else *success = false;
}
```

Файл Stack.h

```
#ifndef STACK_H
#define STACK_H
#include <string>

class Stack{
private:
        std::string name;
        int capacity;
        int last_index;
        int *stack;
```

```
public:
        Stack(std::string name, int capacity);
        ~Stack();
        void add_element(int element, bool* success);
        void get_element(int *element, bool* success);
        std::string get_name() {return this->name;}
        int get_capacity() {return this->capacity;}
        int get_size() {return this->last_index;}
};

#endif
```

Тестирование

Результат тестирования программы представлен в следующей таблице.

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
first 4 second 2 1 2 3 4	first 4 second 2 first second 3 2 2 1 1	first 4 second 2 first second 3 2 2 1 1
first 4 second 4 1 2 3 4	first 4 second 4 first second 4 4 3 3 2 2 1 1	first 4 second 4 first second 4 4 3 3 2 2 1 1

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ (ИСТОЧНИКОВ)

1. Васильев А.Н. Объектно-ориентированное программирование на C++. Издательство: Наука и Техника. Санкт-Петербург, 2016г. 543 стр.

2. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2017. — 624 с.

3. Методическое пособие для проведения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [Электронный ресурс] – URL: https://mirea.aco-avrora.ru/student/files/methodichescoe_posobie_dlya_laboratornyh_rabot_3.pdf (дата обращения 05.05.2021).

4. Приложение к методическому пособию студента по выполнению заданий в рамках курса «Объектно-ориентированное программирование» [Электронный ресурс]. URL: https://mirea.aco-avrora.ru/student/files/Prilozheniye_k_methodichke.pdf (дата обращения 05.05.2021).

5. Видео лекции по курсу «Объектно-ориентированное программирование» [Электронный ресурс]. АСО «Аврора».

6. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие /Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).