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-13-21	Черномуров С.А.
Руководитель практики	Ассистент	Асадова Ю.С.
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 решения задачи используются:

- Объекты стандартных потоков ввода и вывода cin и cout соответственно. Используются для ввода с клавиатуры и вывода на экран.
- Оператор цикла с предусловием while. Используется для инициализации стеков.
- Условный оператор if .. else. Используется для ветвления алгоритма.
- Оператор цикла со счетчиком for. Используется для вывода стеков на экран.
- Манипулятор потока ввода/вывода setw, принадлежит библиотеке iomanip. Используется для форматирования вывода.
- Функция abs для вычисления модуля числа, принадлежит библиотеке cmath.
- Функции min для вычисления минимального из двух чисел, принадлежит библиотеке algorithm.
- Объект ob1 класса Stack. Используется для создания стека 1.
- Объект ob2 класса Stack. Используется для создания стека 2.
- **Класс Stack:**
    - Свойства поля:
        - Поле:
            - Наименование - name;
            - Тип - строковый;
            - Модификатор доступа - закрытый.
        - Поле:
            - Наименование - size;
            - Тип - целочисленный;

        - Модификатор доступа - закрытый.
    - Поле:
        - Наименование - a;
        - Тип - указатель на динамический целочисленный массив;
        - Модификатор доступа - закрытый.
    - Поле:
        - Наименование - curlen;
        - Тип - целочисленный;
        - Модификатор доступа - закрытый.
- Методы:
    - Метод Stack:
        - Функционал - параметризированный конструктор, содержащий имя и размер стека.
    - Метод AddEl:
        - Функционал - параметризированный метод, содержащий добавляемый элемент. Добавляет элемент в стек и возвращает признак успеха.
    - Метод TakeEl:
        - Функционал - параметризированный метод, извлекающий элемент стека и возвращающий признак успеха.
    - Метод StackName:
        - Функционал - параметризированный метод, возвращающий имя стека.
    - Метод StackSize:

    - Функционал - параметризированный метод, возвращающий размер стека.
- Метод CurrentLength:
    - Функционал - параметризированный метод, возвращающий 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.

Функция: main

Функционал: Основной алгоритм программы

Параметры: Отсутствуют

Возвращаемое значение: Целочисленный тип данных - код возврата

Алгоритм функции представлен в таблице 1.

Таблица 1. Алгоритм функции main

№	Предикат	Действия	№ перехода	Комментарий
1		Объявление строковой переменной name	2	
2		Объявление переменной len целочисленного типа данных	3	
3		Считывание с клавиатуры значений переменных name, len	4	
4		Создание объекта ob1 класса Stack путем вызова параметризированного конструктора со строковым параметром name, целочисленным	5	

		параметром len		
5		Считывание с клавиатуры значений переменных name, len	6	
6		Создание объекта ob2 класса Stack путем вызова параметризированного конструктора со строковым параметром name, целочисленным параметром len	7	
7		Объявление переменной el целочисленного типа данных	8	
8	Значение переменной len считано с клавиатуры		9	
			11	Выход из цикла
9	Значение, возвращенное методом CurrentLength объекта ob1 меньше значения, возвращенного методом StackSize объекта ob1	Вызов метода AddEl объекта ob1 с целочисленным параметром el	10	
			11	
10	Значение, возвращенное методом CurrentLength объекта ob2 меньше значения, возвращенного методом StackSize объекта ob2	Вызов метода AddEl объекта ob2 с целочисленным параметром el	8	
			11	
11		Вывод на экран значения, возвращенного методом StackName	12	

		объекта ob1, " ", значение, возвращенное методом StackSize объекта ob1 с последующим переносом на новую строку		
12		Вывод на экран значения, возвращенного методом StackName объекта ob2, " ", значение, возвращенное методом StackSize объекта ob2 с последующим переносом на новую строку	13	
13		Вывод на экран значения, возвращенного методом StackName объекта ob1, с размером поля 15 позиций и прижатием к левому краю	14	
14		Вывод на экран значения, возвращенного методом StackName объекта ob2, с размером поля 15 позиций и прижатием к левому краю с последующим переносом на новую строку	15	
15		Объявление	16	

		целочисленных переменных m, len1, len2		
16		Присвоение переменной len1 значения, возвращенного методом CurrentLength объекта ob1	17	
17		Присвоение переменной len2 значения, возвращенного методом CurrentLength объекта ob2	18	
18		Объявление целочисленной переменной с инициализацией i=0	19	Использование переменной i для счетчика
19	i меньше значения возвращенного функцией min с параметрами len1, len2		20	
			26	Выход из цикла
20	Значение, возвращенное методом CurrentLength объекта ob1 больше нуля	Вызов метода TakeEl объекта ob1 с передачей в него ссылки на целочисленное значение m	21	
			22	
21		Вывод на экран m с размером поля 15 позиций и прижатием к правому краю	22	
22	Значение, возвращенное методом CurrentLength объекта ob2 больше нуля	Вызов метода TakeEl объекта ob2 с передачей в него ссылки на целочисленное	23	

		значение m		
			24	
23		Вывод на экран m с размером поля 15 позиций и прижатием к правому краю	24	
24	Значение i не равно декрементированному значению, возвращенному функцией min с целочисленными параметрами len1, len2	Вывод на экран переноса на новую строку	25	
			25	
25		Инкрементирование i	19	
26	Значение len1 больше значения len2	Вызов метода TakeEl объекта ob1 с передачей в него ссылки на целочисленное значение m	27	
			∅	
27		Вывод на экран переноса на новую строку, значения m с размером поля 15 позиций и прижатием к правому краю	∅	

Конструктор класса: Stack

Модификатор доступа: public

Функционал: Параметризированный конструктор, содержащий имя и размер стека

Параметры: Строковое значение _name, Целочисленное значение _size

Алгоритм конструктора представлен в таблице 2.

Таблица 2. Алгоритм конструктора класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1		Присвоение значению поля name значения переменной _name	2	
2		Присвоение значению поля size значения переменной _size	3	
3		Динамическое выделение памяти для одномерного массива размером size	4	
4		Присвоение значению поля curlen значения -1	∅	

Класс объекта: Stack

Модификатор доступа: public

Метод: StackSize

Функционал: Параметризированный метод, возвращающий размер стека

Параметры: Отсутствуют

Возвращаемое значение: Целочисленный тип данных - значение поля size

Алгоритм метода представлен в таблице 3.

Таблица 3. Алгоритм метода StackSize класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1		Возврат значения поля size	∅	

Класс объекта: Stack

Модификатор доступа: public

Метод: CurrentLength

Функционал: Параметризированный метод, возвращающий количество элементов в стеке

Параметры: Отсутствуют

Возвращаемое значение: Целочисленный тип данных - текущее количество элементво в стеке

Алгоритм метода представлен в таблице 4.

Таблица 4. Алгоритм метода CurrentLength класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1		Возврат инкрементированного значения поля curlen	∅	

Класс объекта: Stack

Модификатор доступа: public

Метод: StackName

Функционал: Параметризированный метод, возвращающий имя стека

Параметры: Отсутствуют

Возвращаемое значение: Строковый тип данных - имя стека

Алгоритм метода представлен в таблице 5.

Таблица 5. Алгоритм метода StackName класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1		Возврат значения поля name	∅	

Класс объекта: Stack

Модификатор доступа: public

Метод: AddEl

Функционал: Параметризированный метод, содержащий добавляемый элемент. Добавляет элемент в стек и возвращает признак успеха

Параметры: Целочисленное значение _el

Возвращаемое значение: Логический тип данных - признак успеха (неудачи) добавления элемента в стек

Алгоритм метода представлен в таблице 6.

Таблица 6. Алгоритм метода AddEl класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1		Инкрементирование значения поля curlen	2	
2	Значение поля curlen меньше или равно декрементированному значению поля size	Присвоение элементу массива a с индексом curlen значения переменной _el	3	
		Возврат логического значения "Ложь"	∅	
3		Возврат логического значения "Истина"	∅	

Класс объекта: Stack

Модификатор доступа: public

Метод: TakeEl

Функционал: Параметризированный метод, извлекающий элемент стека и возвращающий признак успеха

Параметры: Ссылка на целочисленный параметр m

Возвращаемое значение: Логический тип данных - признак успеха (неудачи) извлечения элемента из стека

Алгоритм метода представлен в таблице 7.

Таблица 7. Алгоритм метода TakeEl класса Stack

№	Предикат	Действия	№ перехода	Комментарий
1	Значение, возвращенное методом CurrentLength > 0	Присвоение значению переменной m значения элемента массива a с индексом, равным декрементированному значению, возвращенному методом CurrentLength	2	
		Возврат логического значения "Ложь"	∅	
2		Декрементирование поля curlen	3	
3		Возврат логического значения "Истина"	∅	

Блок-схема алгоритма

Представим описание алгоритмов в графическом виде на рисунках ниже.

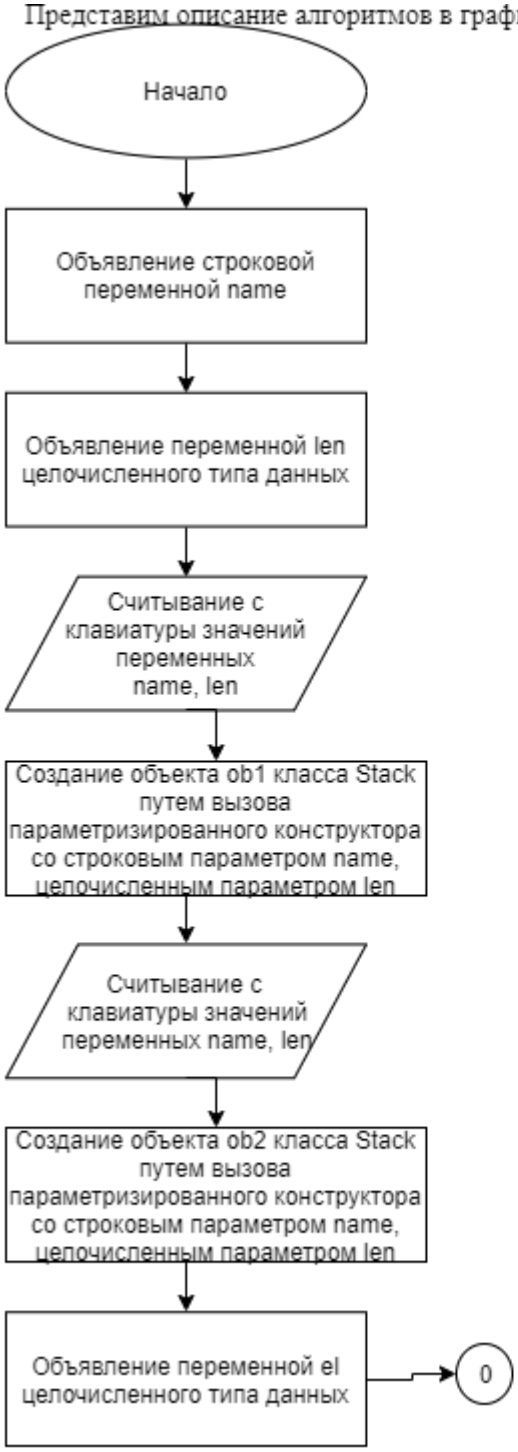

Рис. 1. Блок-схема алгоритма.

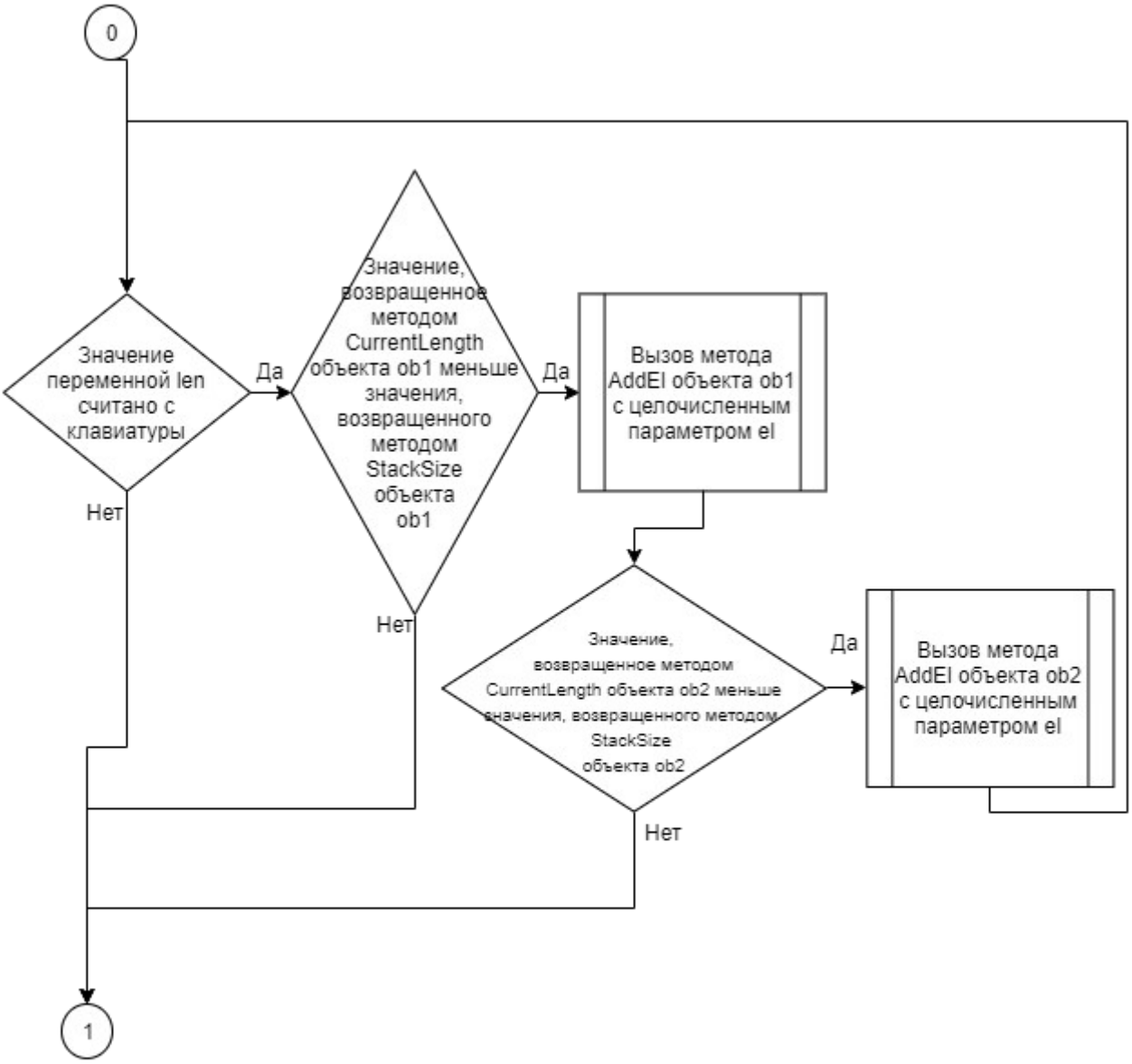

Рис. 2. Блок-схема алгоритма.

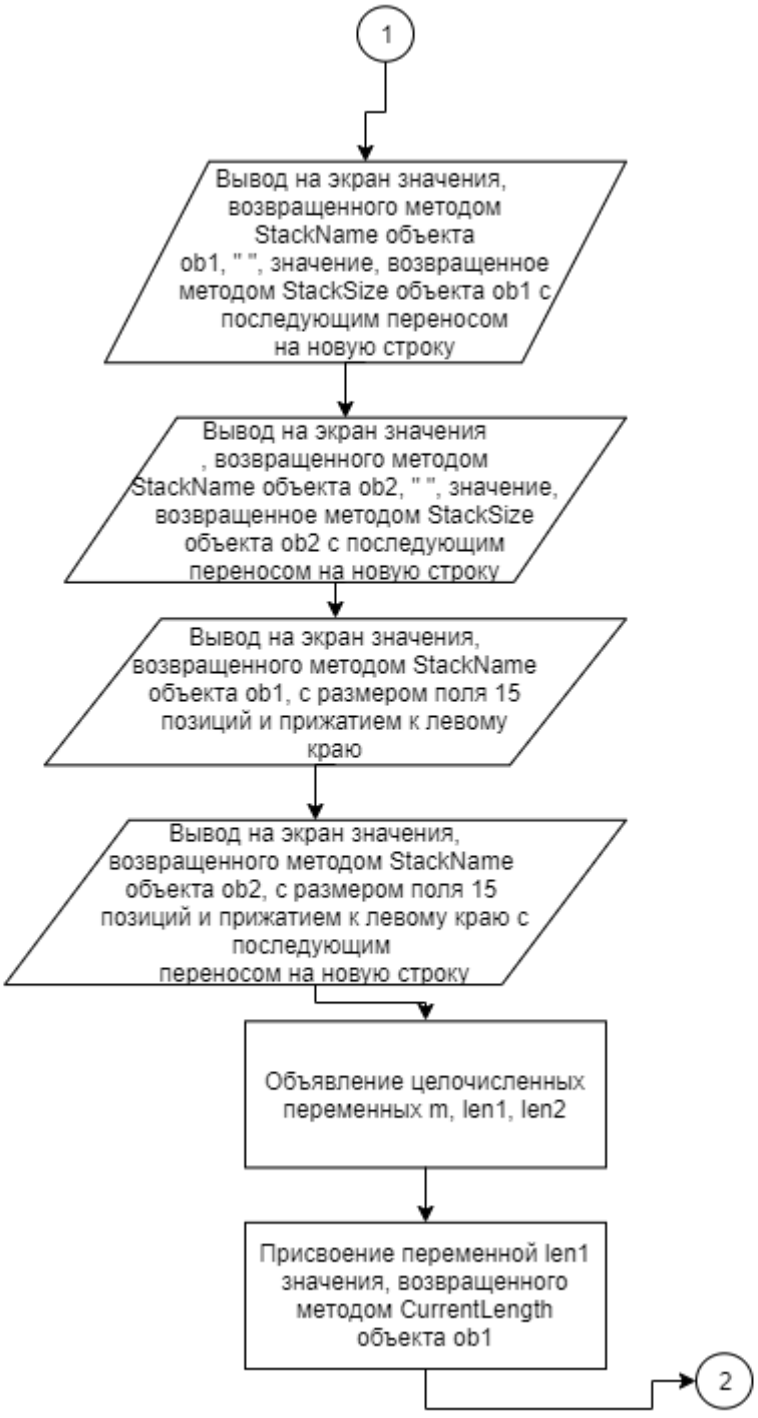

Рис. 3. Блок-схема алгоритма.

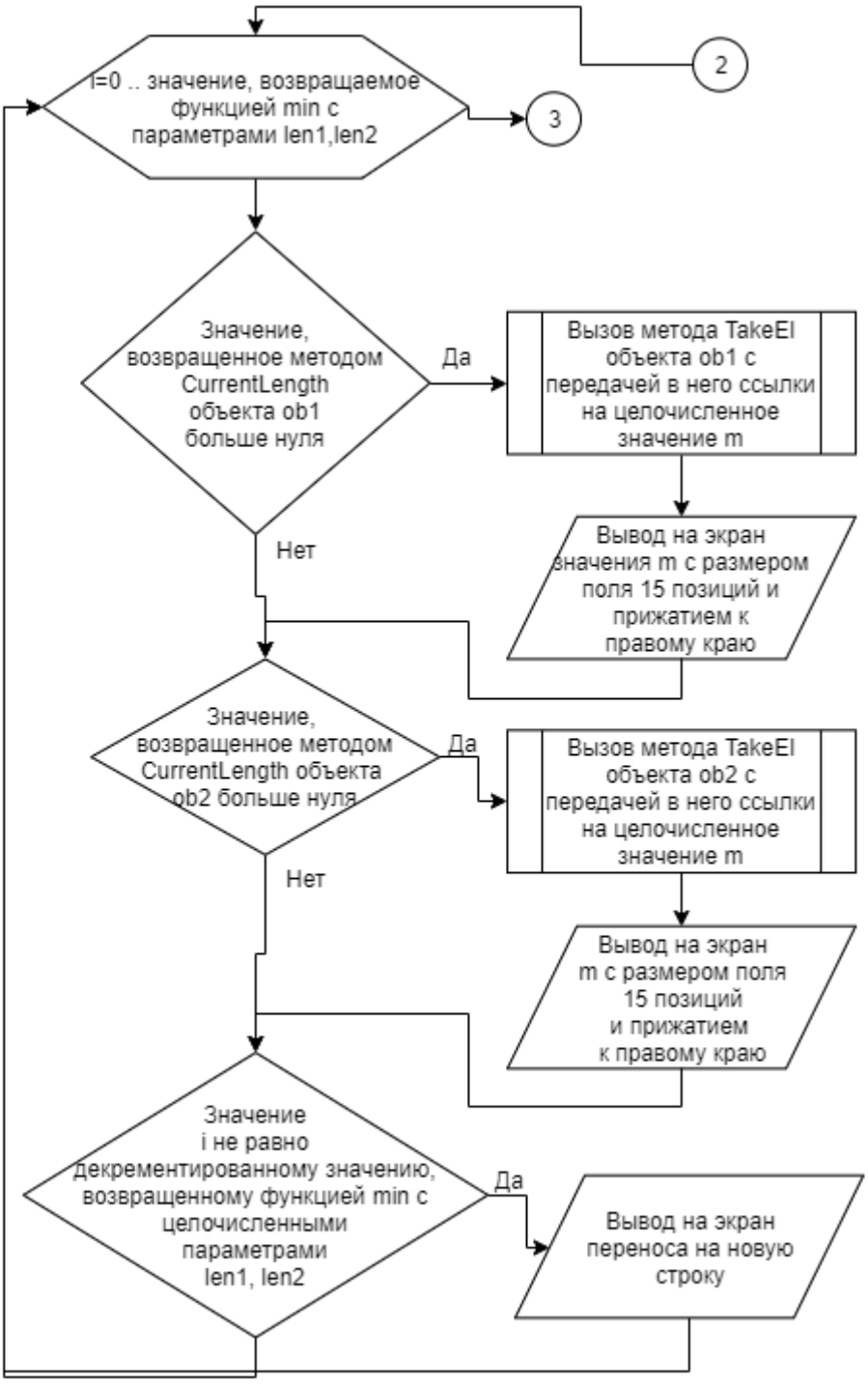

Рис. 4. Блок-схема алгоритма.

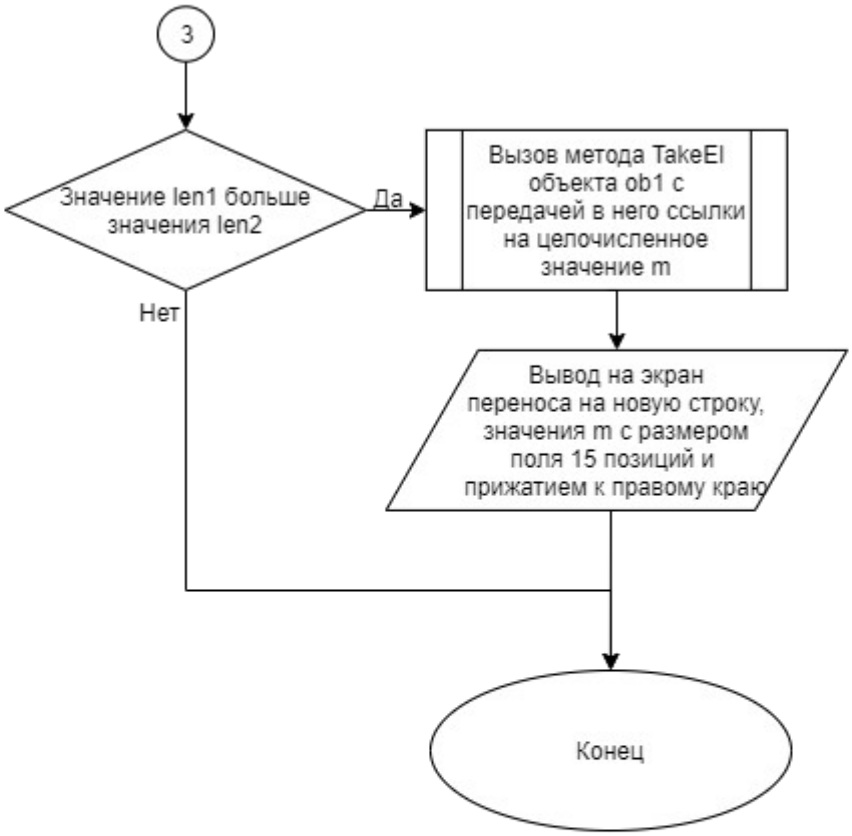

Рис. 5. Блок-схема алгоритма.

Рис. 6. Блок-схема алгоритма.

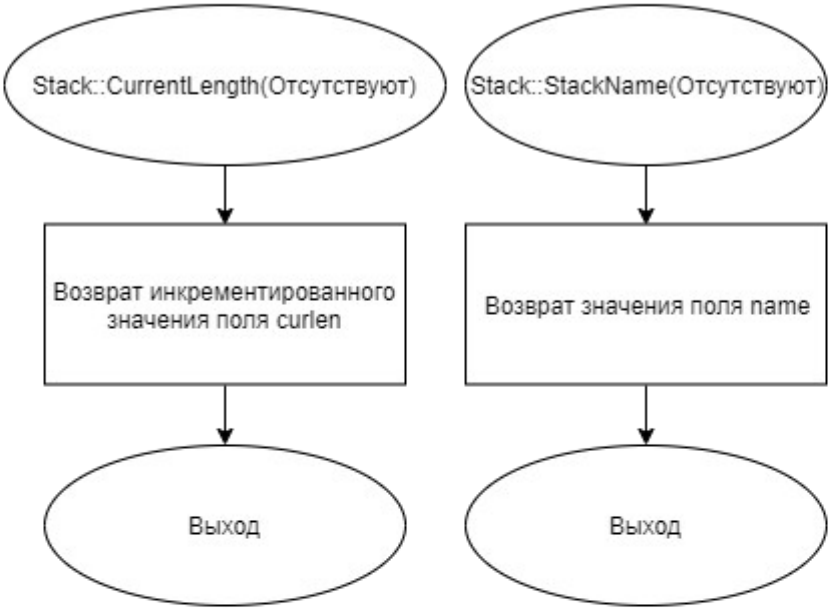

Рис. 7. Блок-схема алгоритма.

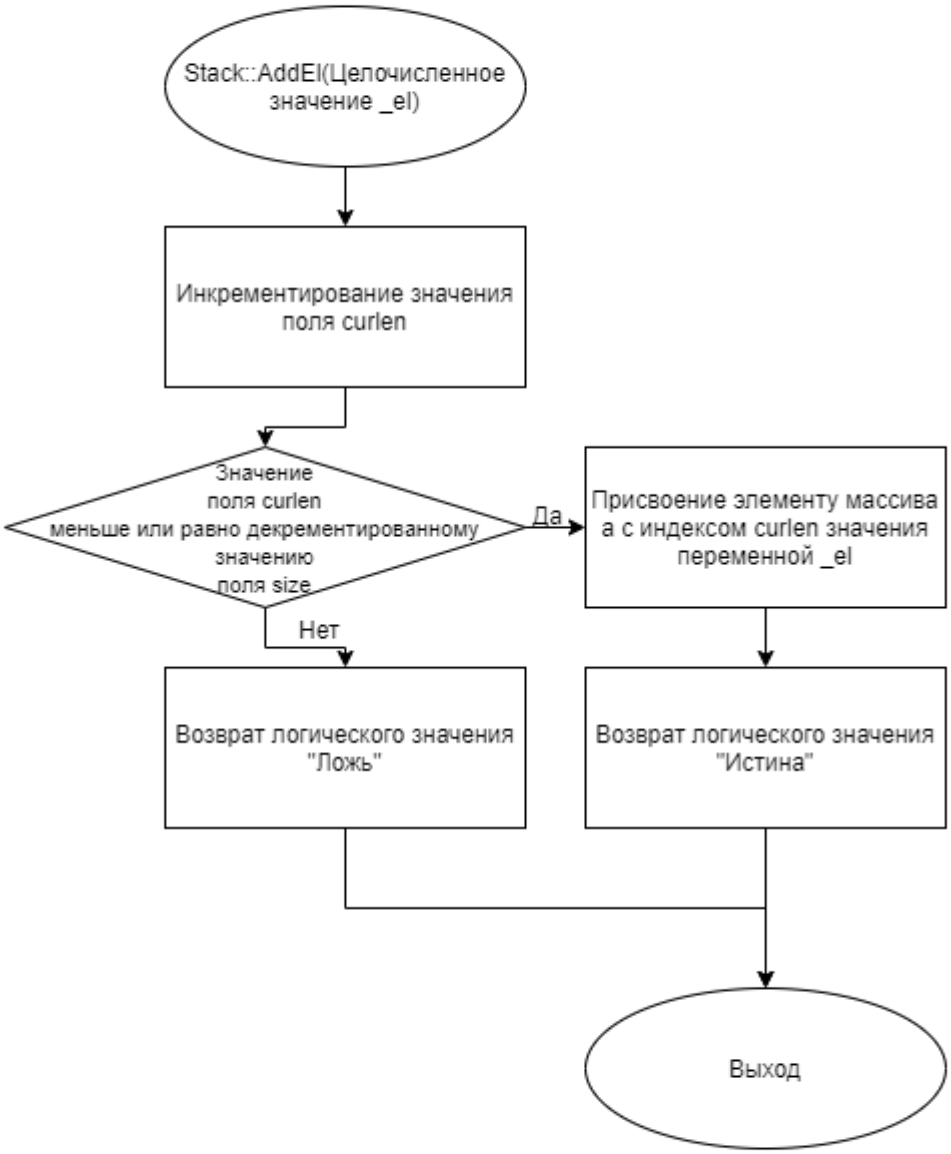

Рис. 8. Блок-схема алгоритма.

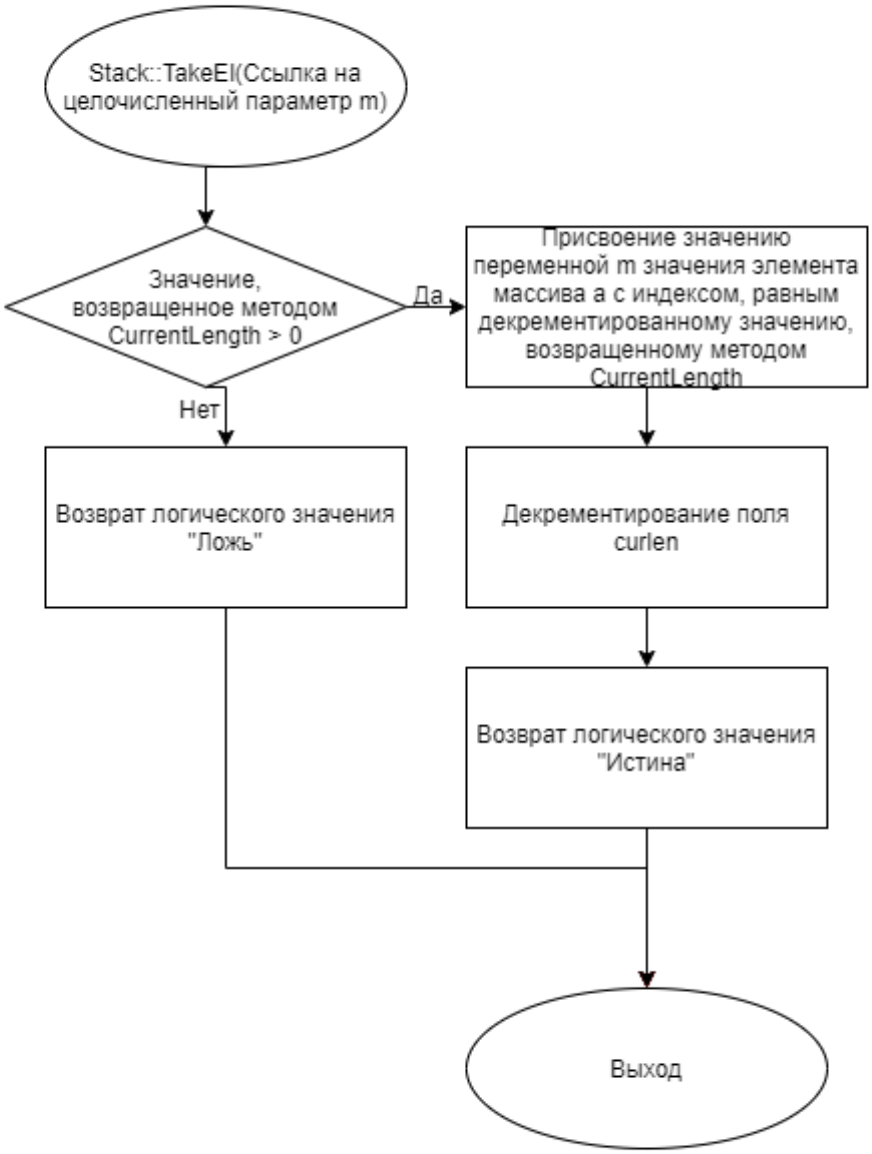

Рис. 9. Блок-схема алгоритма.

Код программы

Программная реализация алгоритмов для решения задачи представлена ниже.

Файл main.cpp

```
#include "Stack.h"
#include <iostream>
#include <string>
#include <iomanip>
#include <algorithm>
#include <cmath>
using namespace std;

int main()
{
        string name; // объявление переменной для имени стека
        int len; // объявление переменной для длины стека

        cin >> name >> len; // ввод значений
        Stack ob1(name, len); // создание первого стека

        cin >> name >> len; // ввод значений
        Stack ob2(name, len); // создание второго стека

        int el; // объявление переменной для элемента стека
        while (cin >> el){ // инициализация значений стеков
                        if (ob1.CurrentLength() < ob1.StackSize())
ob1.AddEl(el);
                        else break;
                        if (ob2.CurrentLength() < ob2.StackSize())
ob2.AddEl(el);
                        else break;
}
        cout << ob1.StackName() << " " << ob1.StackSize() << "\n"; // вывод
параметров первого стека
        cout << ob2.StackName() << " " << ob2.StackSize() << "\n"; // вывод
параметров второго стека

        cout << left << setw(15) << ob1.StackName(); // вывод имени первого
стека
        cout << left << setw(15) << ob2.StackName() << "\n"; // вывод имени
второго стека
        int m, len1, len2; // объявление вспомогательных переменных
        len1 = ob1.CurrentLength(); // присвоение текущей длины стека 1
        len2 = ob2.CurrentLength(); // присвоение текущей длины стека 2

                for (int i = 0;i < min(len1, len2); i++){ // вывод значений
стеков
```

```
                if (ob1.CurrentLength() > 0) {ob1.TakeEl(m); cout << right <<
setw(15) << m;}
                if (ob2.CurrentLength() > 0) {ob2.TakeEl(m); cout << right <<
setw(15) << m;}
                if (i != min(len1, len2)-1) cout << "\n";
                }
        if (len1>len2) {cout<<"\n"; ob1.TakeEl(m); cout << right << setw(15)
<< m;} // вывод последнего значения первого стека (если нужно)
        return 0;
}
```

Файл Stack.cpp

```
#include "Stack.h"
#include <iostream>
#include <string>
using namespace std;

Stack :: Stack(string _name, int _size){
        name = _name; // присвоение имени стека
        size = _size; // присвоение размера стека
        a = new int[size]; // выделение памяти для массива
        curlen = -1; // инициализация текущей длины -1
}

int Stack :: StackSize(){ // возврат размера стека
        return size;
}

int Stack :: CurrentLength(){ // возврат текущей длины стека
        return curlen+1;
}

string Stack :: StackName(){ // возврат имени стека
        return name;
}

bool Stack :: AddEl(int _el){
        curlen++; // инкрементирование текущей длины
        if (curlen <= size-1) {a[curlen] = _el; return true;} // инициализация
элемента стека
        else
                return false;
}

bool Stack :: TakeEl(int &m){
        if (CurrentLength()>0){ // извлечение элемента стека
                m = a[CurrentLength()-1];
                curlen--; // декрементирование текущей длины стека
                return true;
        }
        else return false;

}
```

Файл Stack.h

```
#ifndef _ST_H
#define _ST_H

#include <string>
#include <iostream>
using namespace std;

class Stack{
        private:
                string name;
                int size;
                int curlen;
                int *a;
        public:

                Stack(string _name,int _size);
                bool AddEl(int _el);
                bool TakeEl(int &m);
                string StackName();
                int StackSize();
                int CurrentLength();
};
#endif
```

Тестирование

Результат тестирования программы представлен в следующей таблице.

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
s1 3 s2 2 2 3 4	s1 3 s2 2 s1 s2 4 3 3 2 2	s1 3 s2 2 s1 s2 4 3 3 2 2
s1 2 s2 2 2 3 4	s1 2 s2 2 s1 s2 3 3 2 2	s1 2 s2 2 s1 s2 3 3 2 2
s1 3 s2 4 3 4 5 6	s1 3 s2 4 s1 s2 5 5 4 4 3 3	s1 3 s2 4 s1 s2 5 5 4 4 3 3

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).