Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Сконструировать систему, которая демонстрирует возможность создания объектов одного класса с отличимыми значениями свойств. При этом, появление этих отличий алгоритмически реализовано в параметризованном конструкторе.

Спроектировать объект, который реализует работу стека. Стек предназначен для работы с целыми числами.

Объект имеет характеристики:

- - наименование (строка, не более 10 символов);
- - максимальный размер стека (максимально допустимое количество элементов в стеке). Минимальный размер стека больше или равно 1.

Функционал объекта:- добавить элемент в стек и вернуть признак успеха выполнения действия (логическое). Значение элемента передается посредством параметра. Если стек заполнен, очередной элемент не добавляется и возвращается логическое значение, означающее невозможность выполнения действия.- извлечь элемент из стека и вернуть: признак успеха выполнения действия (логическое). Получение значения элемента стека организовать посредством параметра. Если стек пуст, то возвращается логическое значение, означающее невозможность выполнения действия;- вернуть значение текущего количество элементов в стеке (целое);- вернуть наименование объекта.

Для объекта разработать параметризованный конструктор, которому передается имя объекта и максимальный размер стека.

Алгоритм отработки системы:1. Вводит имя и размер для первого стека.2. Создает объект первого стека с помощью оператора new.3. Выводит имя и размер первого стека.4. В цикле вводит значения элементов первого стека. Ввод осуществляется пока не встретится значение 0.5. Вводит имя и размер для второго стека.6. Объявляет объект второго стека.7. Выводит имя и размер

второго стека.8. В цикле вводит значения элементов второго стека. Ввод осуществляется пока не встретится значение 0.9. В цикле построчно выводит содержимое стеков. Цикл прекращается, когда размер любого из стеков станет равен 0. Проверка происходит после каждого вывода, начиная с 1го стека.

1.1 Описание входных данных

Первая строка:

```
«имя стека 1» «размер стека»
```

Вторая строка:

Последовательность целых чисел, разделенных пробелами, последнее число 0.

Третья строка:

```
«имя стека 2» «размер стека»
```

Четвертая строка:

Последовательность целых чисел, разделенных пробелами, последнее число 0.

1.2 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 занимает поле длины 15 позиции и прижата к левому краю.

Четвертая строка и далее построчно:

```
«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. Вывод прекращается после извлечения и вывода последнего элемента в каком-нибудь стеке.

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект st1 класса Stack предназначен для реализации работы стека для первого случая;
- объект st2 класса Stack предназначен для реализации работы стека для второго случая;
- функция main для выполнения основного алгоритма программы;
- библиотека iostream;
- оператор стандартного потока ввода данных cin;
- оператор стандартного потока вывода данных cout;
- оператор смещения влево вывода в консоль - left;
- оператор перехода на следующую строку в консоли - endl;
- функция манипуляции смещением вывода значения в консоль - setw(int);
- операторы логического ветвления - if/else;
- оператор цикла с постусловием - while.

Класс Stack:

- свойства/поля:
    - поле имя стека:
        - наименование — stackName;
        - тип — string;
        - модификатор доступа — private;
    - поле максимальный размер стека:
        - наименование — maxSize;
        - тип — int;
        - модификатор доступа — private;
    - поле вектор для хранения данных:

        - наименование — myStack;
        - тип — vector<int>;
        - модификатор доступа — private;
- функционал:
    - метод Stack — выделение памяти под объект и инициаллизация полей аргументами конструктора;
    - метод get_name — вовзвращает имя стека;
    - метод get_size — вовзвращает текущий размер стека;
    - метод push_item — вставляет значение элемента в стек;
    - метод pop_item — вовзвращает значение последнего элемента, введенного в стек;
    - метод get_max_size — возвращает максимальный размер стека.

3 ОПИСАНИЕ АЛГОРИТМОВ

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

3.1 Алгоритм конструктора класса Stack

Функционал: выделение памяти под объект и инициаллизация полей аргументами конструктора.

Параметры: string name - название стека, int length - максимальный размер стека.

Алгоритм конструктора представлен в таблице 1.

Таблица 1 – Алгоритм конструктора класса Stack

№	Предикат	Действия	№ перехода
1		инициализация поля stackName значением подстроки name размером в 10 символов	2
2		инициализация поля maxSize значением переменной length	3
3	length не натуральное число?	определение поля maxSize значением 1	∅
			∅

3.2 Алгоритм метода get_name класса Stack

Функционал: вовзвращает имя стека.

Параметры: нет.

Возвращаемое значение: string - имя стека.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода get_name класса Stack

№	Предикат	Действия	№ перехода
1		вывод значения поля stackName	∅

3.3 Алгоритм метода get_size класса Stack

Функционал: вовзвращает текущий размер стека.

Параметры: нет.

Возвращаемое значение: int - текущий размер стека.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода get_size класса Stack

№	Предикат	Действия	№ перехода
1		вывод значения размера стека myStack.size()	∅

3.4 Алгоритм метода push_item класса Stack

Функционал: вставляет значение элемента в стек.

Параметры: int item - значение элемента.

Возвращаемое значение: bool - идентификатор корректности выполнения метода.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода push_item класса Stack

№	Предикат	Действия	№ перехода
1	стек переполнен?	вывод значения false	∅
		вывод значения true	∅

3.5 Алгоритм метода pop_item класса Stack

Функционал: вовзвращает значение последнего элемента, введенного в стек.

Параметры: int& item - определение последнего элемента стека.

Возвращаемое значение: bool - идентификатор корректности выполнения метода.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода pop_item класса Stack

№	Предикат	Действия	№ перехода
1	стек не пустой?	вывод значения false	∅
		определение переменной по ссылке item значением последнего элемента вектора myStack	2
2		удаление последнего элемента вектора myStack	3
3		вывод значения true	∅

3.6 Алгоритм метода get_max_size класса Stack

Функционал: возвращает максимальный размер стека.

Параметры: нет.

Возвращаемое значение: int - максимальный размер стека.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода get_max_size класса Stack

№	Предикат	Действия	№ перехода
1		вывод значения поля maxSize	∅

3.7 Алгоритм функции main

Функционал: главная функция программы.

Параметры: нет.

Возвращаемое значение: целое, идентификатор работоспособности программы.

Алгоритм функции представлен в таблице 7.

Таблица 7 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		объявление переменной name типа string	2
2		объявление переменной digit типа int	3
3		инициализация переменной flag типа bool значением true	4
4		определение значений переменных name и digit	5
5		инициализация объекта st1 класса Stack конструктором с аргументами name и digit	6
6		ввод значения в переменную digit	7
7	digit != 0 и в стеке есть место?	определение flag значением вызова у объекта st1 метода push_item с аргументом digit	8
			9
8		ввод значения в переменную digit	7
9		ввод значения в переменную name	10
10	предыдущий цикл обработал все числа до 0?		11
		ввод значения в переменную name	10
11		определение переменной flag значением true	12
12		определение значений переменной digit	13
13		инициализация объекта st2 класса Stack конструктором с аргументами name и digit	14
14		ввод значения в переменную digit	15
15	digit != 0 и в стеке есть место?	определение flag значением вызова у объекта st2 метода push_item с аргументом digit	16

№	Предикат	Действия	№ перехода
			17
16		ввод значения в переменную digit	15
17		вывод имени и текущего размера объекта st1 через пробел	18
18		вывод имени и текущего размера объекта st2 через пробел	19
19		вывод имени объекта st1 с центрированием влево и дополнением пробелов	20
20		вывод имени объекта st2 с центрированием влево и дополнением пробелов	21
21	в стеках существуют элементы?	инициализация переменных val1 и val2 значением 0	22
			∅
22		инициализация переменной s1 вызовом у объекта st1 метода pop_item с аргументом val1	23
23		инициализация переменной s2 вызовом у объекта st2 метода pop_item с аргументом val2	24
24	из стека st1 есть, что извлечь?	вывод элемента объекта st1 с центрированием вправо и дополнением пробелов	25
		вывод пропуска	25
25	стек st1 закончился?		∅
			26
26	из стека st2 есть, что извлечь?	вывод элемента объекта st2 с центрированием вправо и дополнением пробелов	27
		вывод пропуска	27
27		вывод перехода на следующую строку	21

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-9.

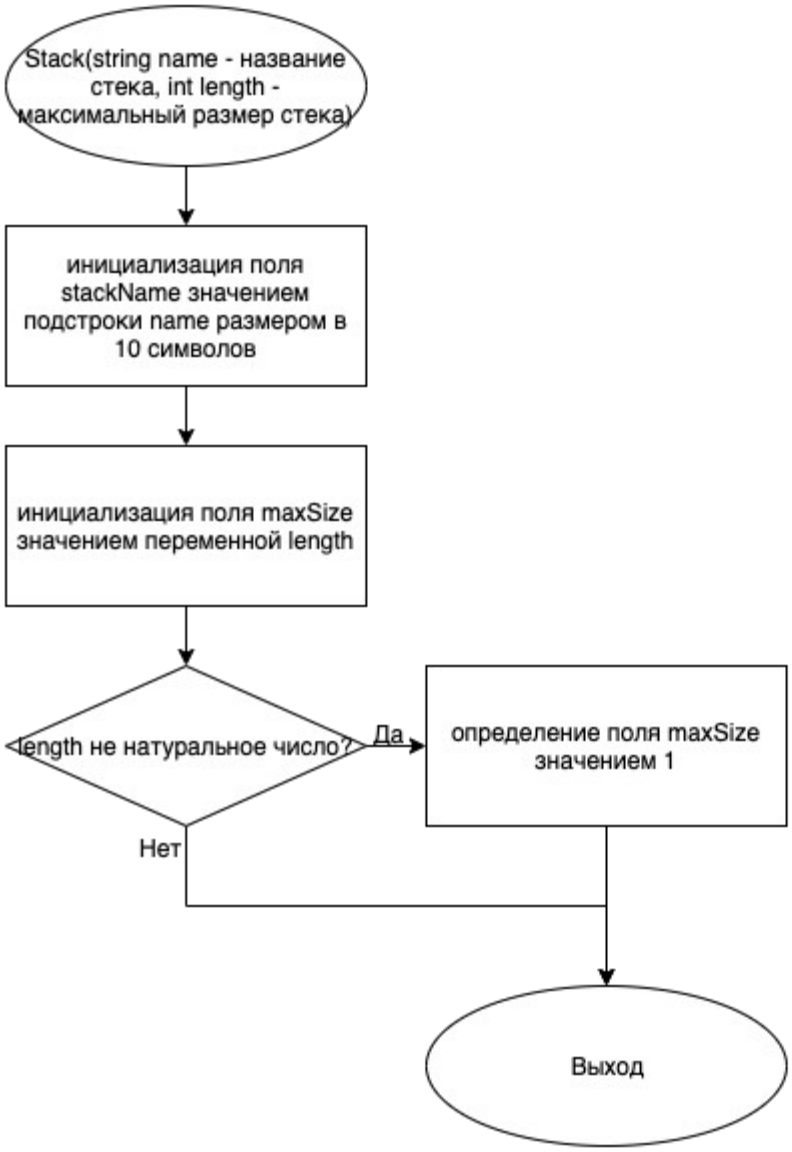

Рисунок 1 – Блок-схема алгоритма

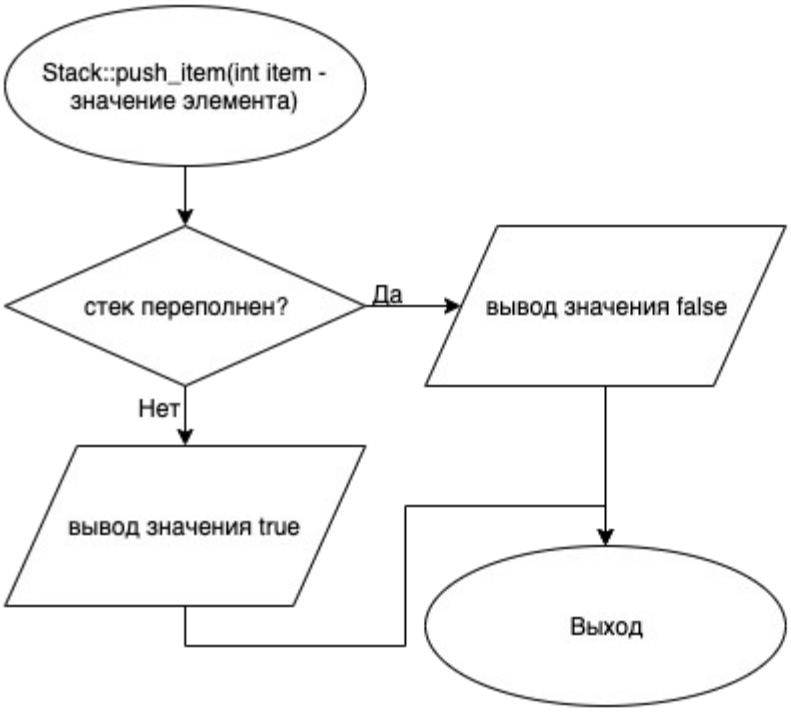

Рисунок 2 – Блок-схема алгоритма

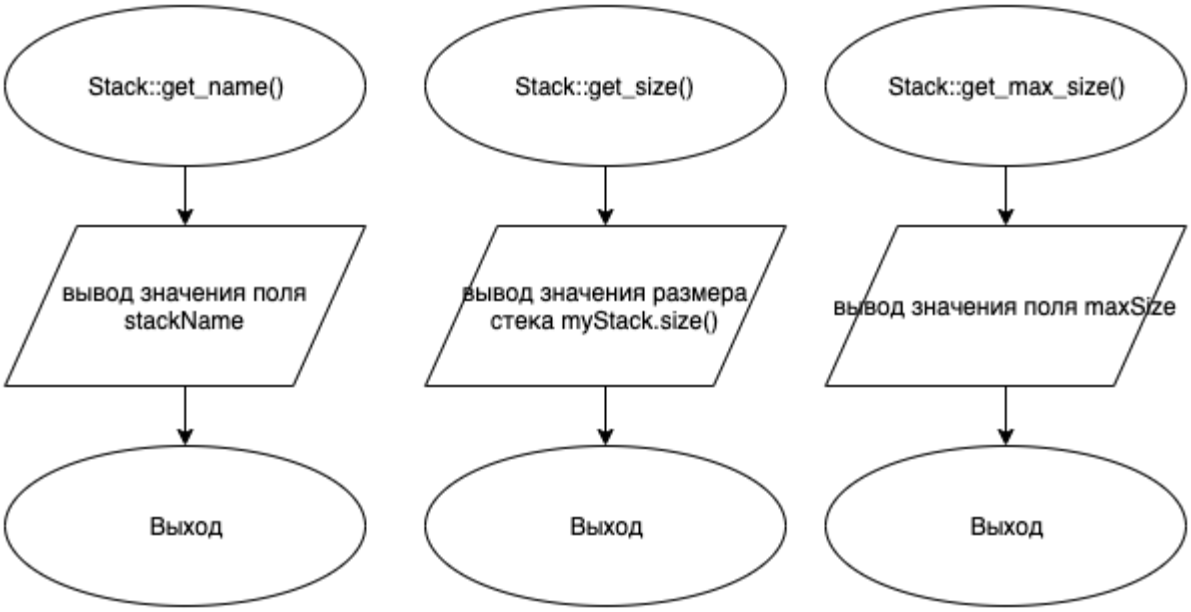

Рисунок 3 – Блок-схема алгоритма

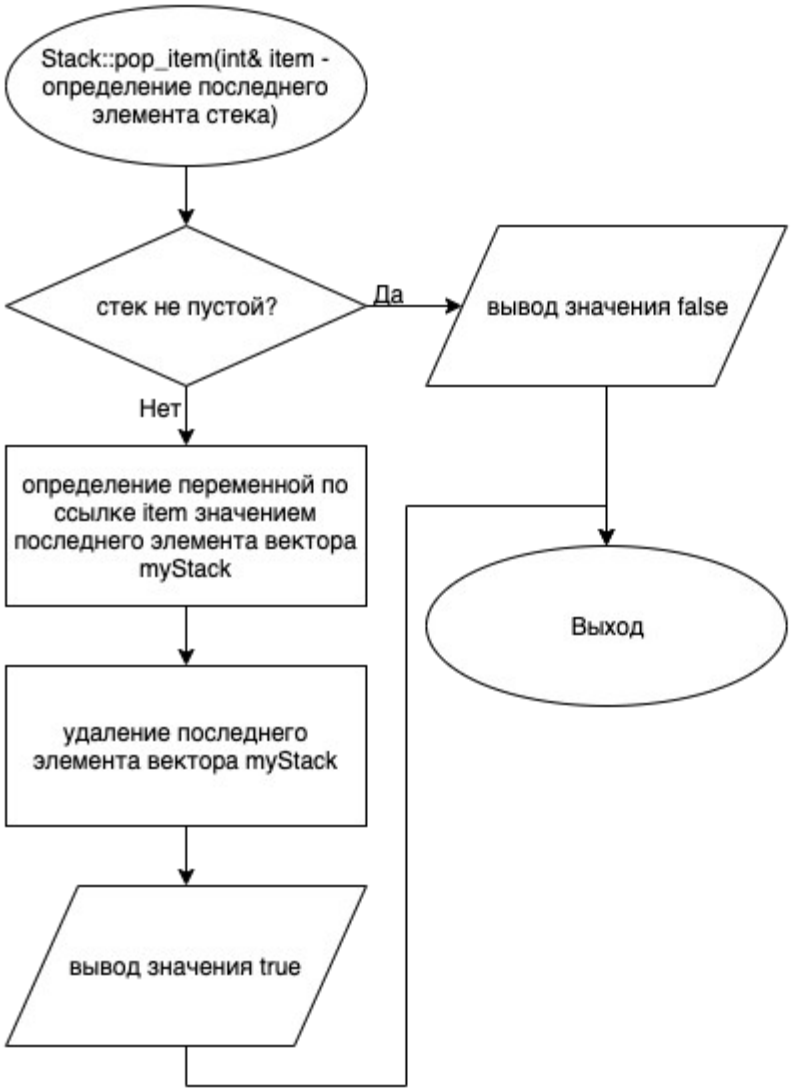

Рисунок 4 – Блок-схема алгоритма

Рисунок 5 – Блок-схема алгоритма

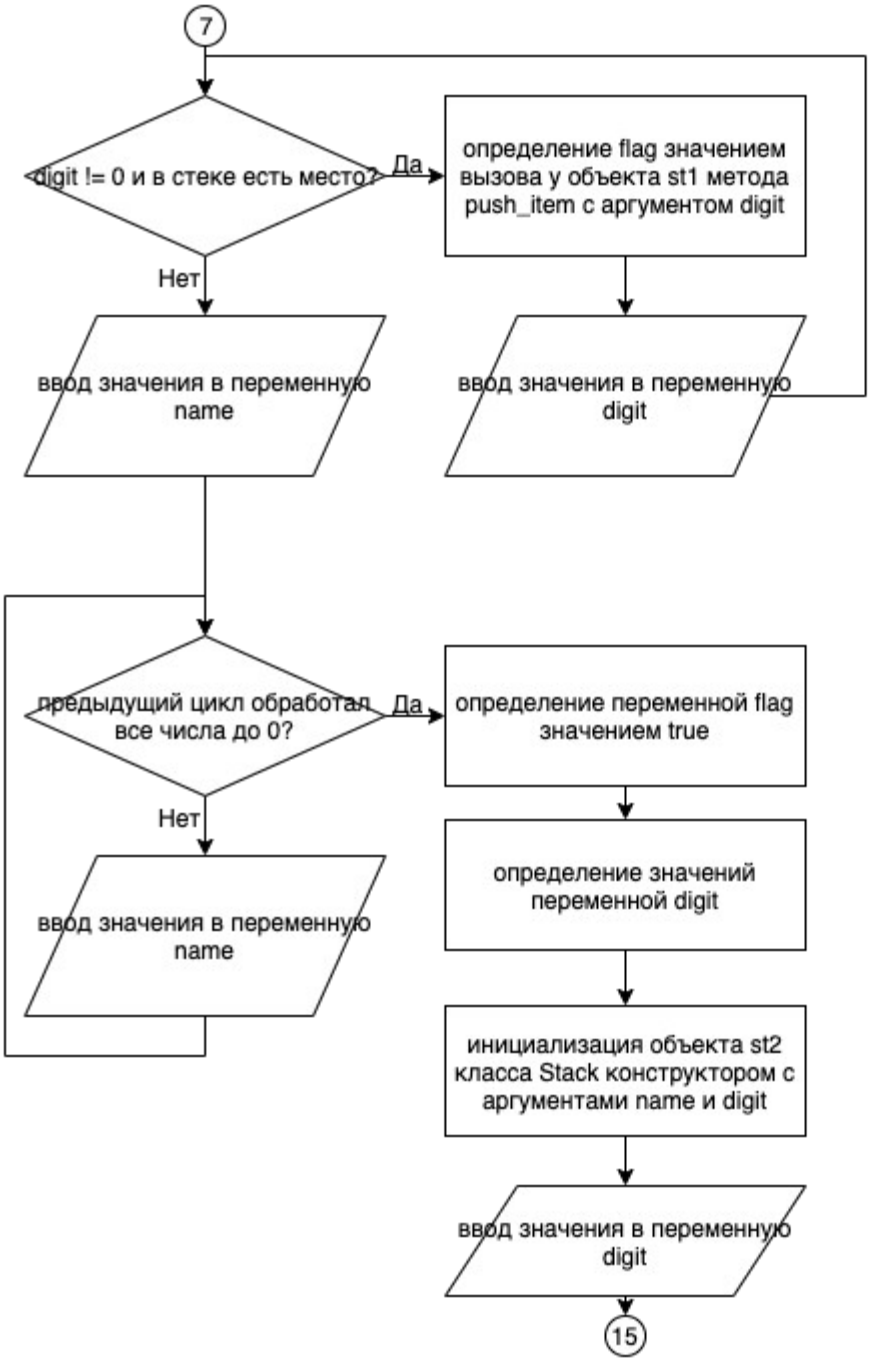

Рисунок 6 – Блок-схема алгоритма

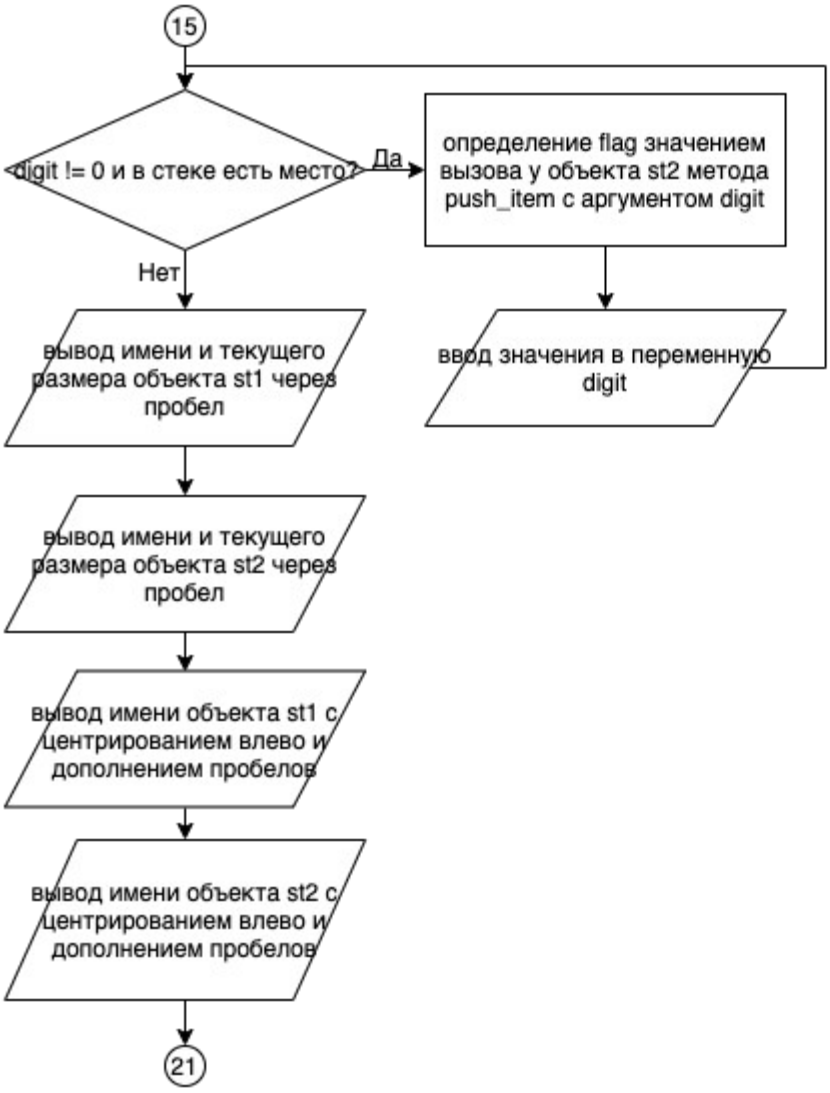

Рисунок 7 – Блок-схема алгоритма

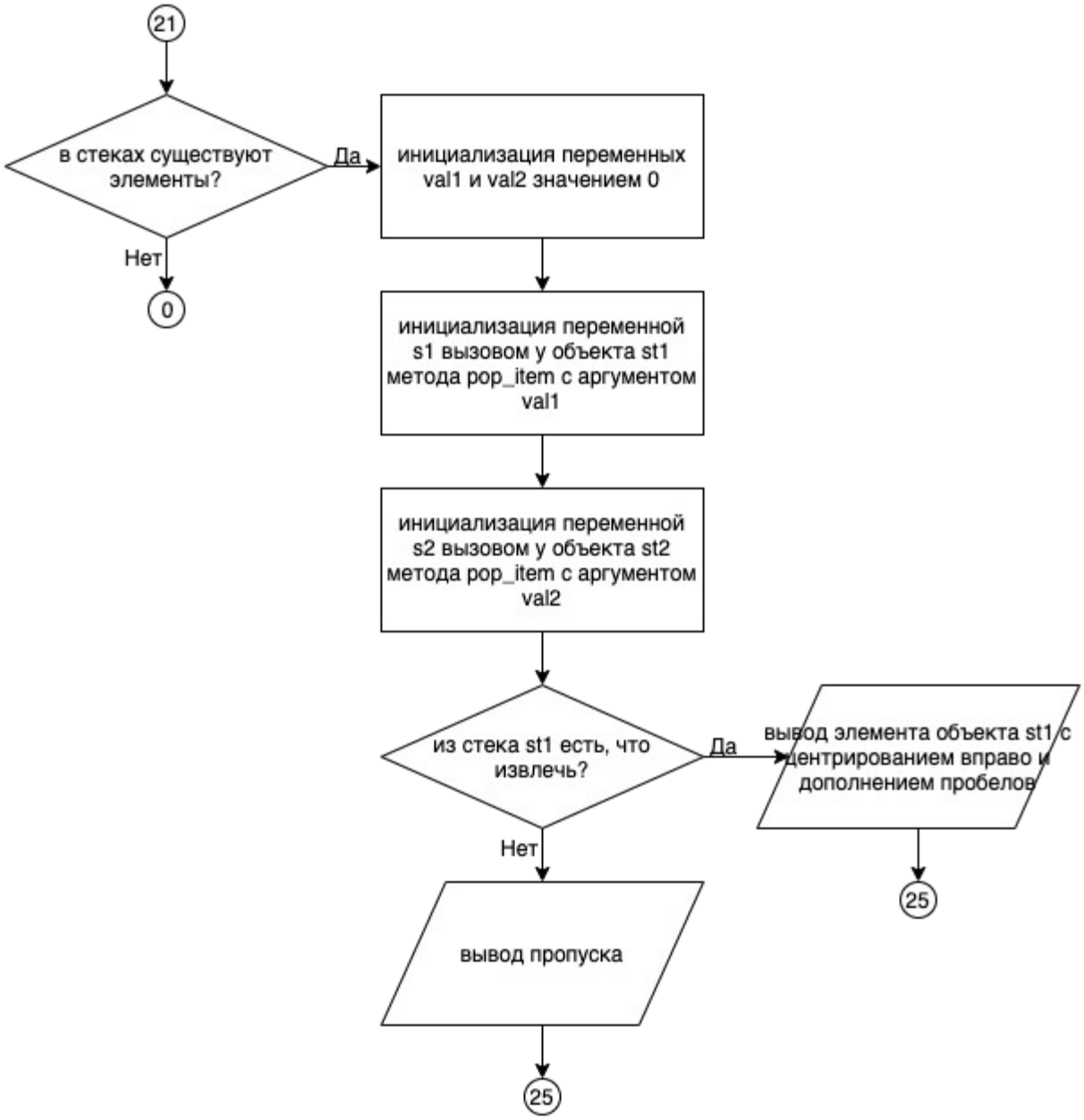

Рисунок 8 – Блок-схема алгоритма

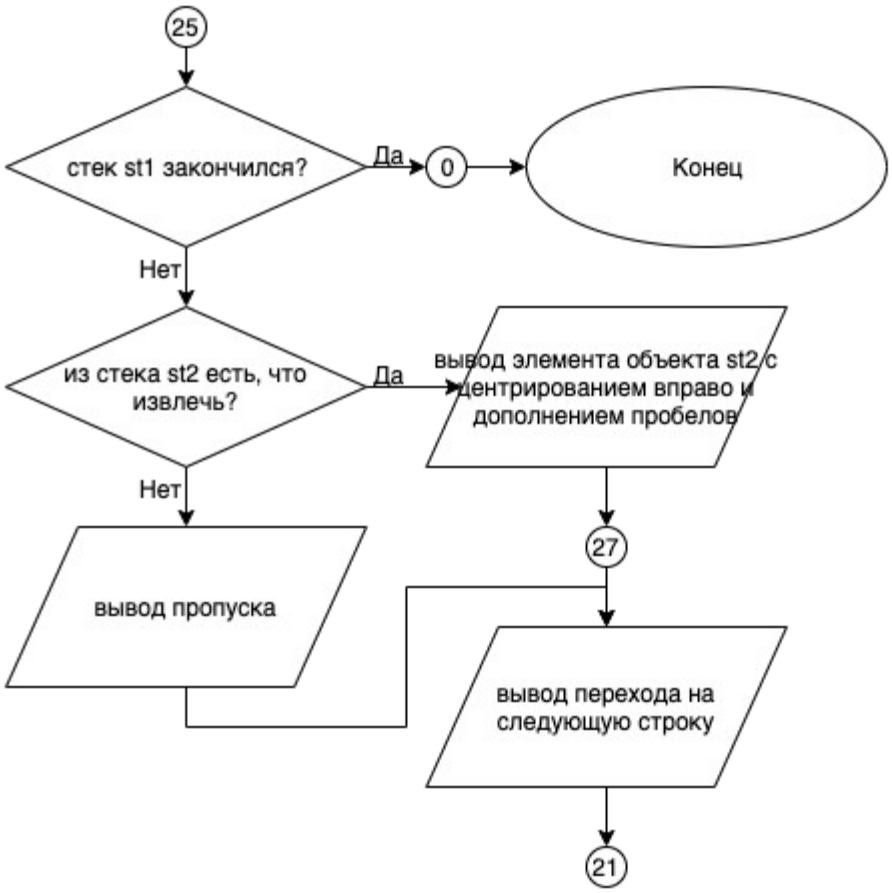

Рисунок 9 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

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

int main()
{
   string name;
   int digit;

   bool flag = true;

   cin >> name >> digit;
   Stack st1(name, digit);

   cin >> digit;
   while (digit != 0){
      st1.push_item(digit);
      cin >> digit;
   }

   flag = true;

   cin >> name >> digit;
   Stack st2(name, digit);

   cin >> digit;
   while (digit != 0 && flag == true){
      flag = st2.push_item(digit);
      cin >> digit;
   }

   cout << st1.get_name() << " " << st1.get_max_size() << endl;
   cout << st2.get_name() << " " << st2.get_max_size() << endl;

   cout << left << setw(15) << st1.get_name();
   cout << setw(15) << st2.get_name() << endl;
```

```
        int val1 = 0, val2 = 0;
        while (true){

           bool s1 = st1.pop_item(val1);
           bool s2 = st2.pop_item(val2);

           if (s1){
              cout << right << setw(15) << val1;
           } else {
              cout << setw(15) << " ";
           }

           if (st1.get_size() == 0) break;

           if (s2){
              cout << right << setw(15) << val2;
           } else {
              cout << setw(15) << " ";
           }

           cout << endl;

           if (st1.get_size()== 0 || st2.get_size() == 0) break;
        }
        return(0);
     }
```

5.2 Файл Stack.cpp

Листинг 2 – Stack.cpp

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

     Stack::Stack(string name, int lenght): maxSize(lenght){
        stackName = name.substr(0, 10);
        if (lenght < 1){
           maxSize = 1;
        }
     }

     string Stack::get_name(){
        return stackName;
     }

     int Stack::get_max_size(){
        return maxSize;
     }
```

```
int Stack::get_size(){
   return myStack.size();
}

bool Stack::push_item(int item){
   if (myStack.size() == maxSize){
      return false;
   }
   myStack.push_back(item);
   return true;
}

bool Stack::pop_item(int& item){
   if (!myStack.size()){
      return false;
   }
   item = myStack.back();
   myStack.pop_back();
   return true;
}
```

5.3 Файл Stack.h

Листинг 3 – Stack.h

```
#ifndef __STACK__H
#define __STACK__H
using namespace std;
#include <string>
#include <vector>

class Stack{
private:
   string stackName;
   int maxSize;
   vector<int> myStack;
public:
   Stack(string, int);
   string get_name();
   int get_max_size();
   int get_size();
   bool push_item(int);
   bool pop_item(int&);
};

#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 8.

Таблица 8 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
first 1 4 5 6 second 2 0 8	first 1 first 1 first first 4	first 1 first 1 first first 4
first 5 3 5 6 7 0 second 3 1 2 0	first 5 second 3 first second 7 2 6 1	first 5 second 3 first second 7 2 6 1

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19 Единая система программной документации.

2. Методическое пособие студента для выполнения 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).

3. 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).

4. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2019. — 624 с.

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).