Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Дан объект следующей конструкции:

В закрытом доступе имеется массив целого типа и поле его длины. Количество элементов массива четное и больше двух. Объект имеет функциональность:

- Конструктор по умолчанию, в начале работы выдает сообщение;
- Параметризированный конструктор, передается целочисленный параметр. Параметр должен иметь значение больше 2 и быть четным. По значению параметра определяется размерность целочисленного массива из закрытой области. В начале работы выдает сообщение;
- Метод деструктор, который выдает сообщение что он отработал;
- Метод ввода данных для созданного массива;
- Метод 1, который суммирует значения очередной пары элементов и сумму присваивает первому элементу пары. Далее суммирует элементы полученного массива и возвращает это значение. Например, пусть массив состоит из элементов {1,2,3,4}. В результате суммирования пар получим массив {3,2,7,4};
- Метод 2, который умножает значения очередной пары элементов и результат присваивает первому элементу пары. Далее суммирует элементы полученного массива и возвращает это значение. Например, пусть массив состоит из элементов {1,2,3,4}. В результате умножения пар получим массив {2,2,12,4};
- Метод который, суммирует значения элементов массива и возвращает это значение.

Разработать функцию, которая в качестве параметра получает объект по значению. Функция вызывается метод 2, далее выводит сумму элементов массива

с новой строки.

В основной функции реализовать алгоритм:

1. Ввод размерности массива.
2. Если размерность массива некорректная, вывод сообщения и завершить работу алгоритма.
3. Вывод значения размерности массива.
4. Создание объекта с аргументом размерности массива.
5. Вызов метода для ввода значений элементов массива.
6. Вызов функции передача в качестве аргумента объекта.
7. Вызов метода 1 от имени объекта.
8. Вывод суммы элементов массива объекта с новой строки.

Разработать конструктор копии объекта для корректного выполнения вычислений. В начале работы конструктор копии выдает сообщение с новой строки.

1.1 Описание входных данных

Первая строка:

«Целое число»

Вторая строка:

«Целое число» «Целое число» . . .

Пример:

```
8
1 2 3 4 5 6 7 8
```

1.2 Описание выходных данных

Если введенная размерность массива допустима, то в первой строке выводится это значение:

```
«Целое число»
```

Если введенная размерность массива не больше двух или нечетная, то в первой строке выводится некорректное значение и вопросительный знак:

```
«Целое число»?
```

Конструктор по умолчанию в начале работы с новой строки выдает сообщение:

```
Default constructor
```

Параметризированный конструктор в начале работы с новой строки выдает сообщение:

```
Constructor set
```

Конструктор копирования в начале работы с новой строки выдает сообщение:

```
Copy constructor
```

Деструктор в начале работы с новой строки выдает сообщение:

```
Destructor
```

Пример вывода:

```
8
Constructor set
Copy constructor
120
Destructor
56
Destructor
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект cin класса istream предназначен для ввода данных с клавиатуры;
- объект cout класса ostream предназначен для вывода текста;
- for - оператор цикла со счетчиком;
- new - оператор резервирования памяти;
- delete - оператор освобождения памяти;
- if..else - условный оператор.

Класс Zrada:

- свойства/поля:
    - поле массив целого типа:
        - наименование — array;
        - тип — *int;
        - модификатор доступа — private;
    - поле длинна массива:
        - наименование — size;
        - тип — int;
        - модификатор доступа — private;
- функционал:
    - метод Zrada — параметризированный конструктор;
    - метод Zrada — конструктор копирования;
    - метод ~Zrada — деструктор;
    - метод fill_array — заполнение массива значениями;
    - метод sum_neighbors — сложение соседних элементов массива;
    - метод multiply_neighbors — умножение соседних элементов массива;

- метод sum — сумма элементов массива.

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 Zrada

Функционал: параметризированный конструктор.

Параметры: arr_size - целочисленный, длинна массива.

Алгоритм конструктора представлен в таблице 1.

Таблица 1 – Алгоритм конструктора класса Zrada

№	Предикат	Действия	№ перехода
1		Вывод "Constructor set"	2
2		Присвоение свойству size параметра arr_size	3
3		Выделение памяти под массив array	∅

3.2 Алгоритм деструктора класса Zrada

Функционал: деструктор.

Параметры: нет.

Алгоритм деструктора представлен в таблице 2.

Таблица 2 – Алгоритм деструктора класса Zrada

№	Предикат	Действия	№ перехода
1		Освобождение выделенной памяти под динамический массив array	2
2			∅

3.3 Алгоритм метода fill_array класса Zrada

Функционал: заполнение массива значениями.

Параметры: нет.

Возвращаемое значение: ничего не возвращает.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода fill_array класса Zrada

№	Предикат	Действия	№ перехода
1		Объявление целочисленной переменной value	2
2		Инициализация целочисленной переменной i	3
3	i < size	Ввод значения i-той ячейки массива array; инкрементация i	3
			∅

3.4 Алгоритм метода sum_neighbors класса Zrada

Функционал: сложение соседних элементов массива.

Параметры: нет.

Возвращаемое значение: целочисленное - сумма элементов модифицированного массива.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода sum_neighbors класса Zrada

№	Предикат	Действия	№ перехода
1		Инициализация целочисленной переменной i	2
2	i < size	Присваивание i-тому элементу массива суммы текущего и следующего элемента; увеличение i на 2	3
			3

№	Предикат	Действия	№ перехода
3		Вызов метода sum() и возврат результата работы	∅

3.5 Алгоритм метода multiply_neighbors класса Zrada

Функционал: умножение соседних элементов массива.

Параметры: нет.

Возвращаемое значение: целочисленное - сумма элементов модифицированного массива.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода multiply_neighbors класса Zrada

№	Предикат	Действия	№ перехода
1		Инициализация целочисленной переменной i	2
2	i < size	Присваивание i-тому элементу массива произведения текущего и следующего элемента; увеличение i на 2	3
			3
3			4
4		Вызов метода sum() и возврат результата работы	∅

3.6 Алгоритм метода sum класса Zrada

Функционал: сумма элементов массива.

Параметры: нет.

Возвращаемое значение: целочисленное - сумма элементов массива.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода sum класса Zrada

№	Предикат	Действия	№ перехода
1		Инициализация целочисленной переменной sum = 0	2
2		Инициализация целочисленной переменной i = 0	3
3	i < size	Присваивание переменной sum значения суммы sum и i-того элемента массива; инкрементация i	4
			4
4		Возврат значения переменной sum	∅

3.7 Алгоритм конструктора класса Zrada

Функционал: конструктор копирования.

Параметры: ссылка - объект класса Zrada.

Алгоритм конструктора представлен в таблице 7.

Таблица 7 – Алгоритм конструктора класса Zrada

№	Предикат	Действия	№ перехода
1		Вывод "Copy constructor"	2
2		Присвоение свойству size создаваемого объекта значения поля size объекта копирования	3
3		Выделение памяти под динамический массив array длинной size	4
4		Инициализация целочисленной переменной i	5
5	i < size	Присваивание i-той ячейке массива создаваемого объекта значения i-той ячейки объекта копирования; инкрементация i	5
			∅

3.8 Алгоритм конструктора класса Zrada

Функционал: конструктор по умолчанию.

Параметры: нет.

Алгоритм конструктора представлен в таблице 8.

Таблица 8 – Алгоритм конструктора класса Zrada

№	Предикат	Действия	№ перехода
1		Вывод "Default constructor"	∅

3.9 Алгоритм функции main

Функционал: точка входа в программу.

Параметры: нет.

Возвращаемое значение: целочисленное - индикатор корректности выполнения программы.

Алгоритм функции представлен в таблице 9.

Таблица 9 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление целочисленной переменной arr_size	2
2		Ввод значения переменной с клавиатуры	3
3	!(arr_size > 2 && arr_size % 2 == 0)	Вывод «значение переменной»?	4
			∅
4		Вывод arr_size	5
5		Объявление объекта класса Zrada с помощью параметризированного конструктора	6
6		Вызов метода fill_array()	7
7		Вызов функции turn_on()	8

№	Предикат	Действия	№ перехода
8		Вызов метода sum_neighbors()	9
9		Вызов метода sum() и вывод результата вызова	∅

3.10 Алгоритм функции turn_on

Функционал: вызов второго метода у экземпляра класса.

Параметры: Zrada - объект класса.

Возвращаемое значение: ничего не возвращает.

Алгоритм функции представлен в таблице 10.

Таблица 10 – Алгоритм функции turn_on

№	Предикат	Действия	№ перехода
1		Вызов метода multiply_neighbors()	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-6.

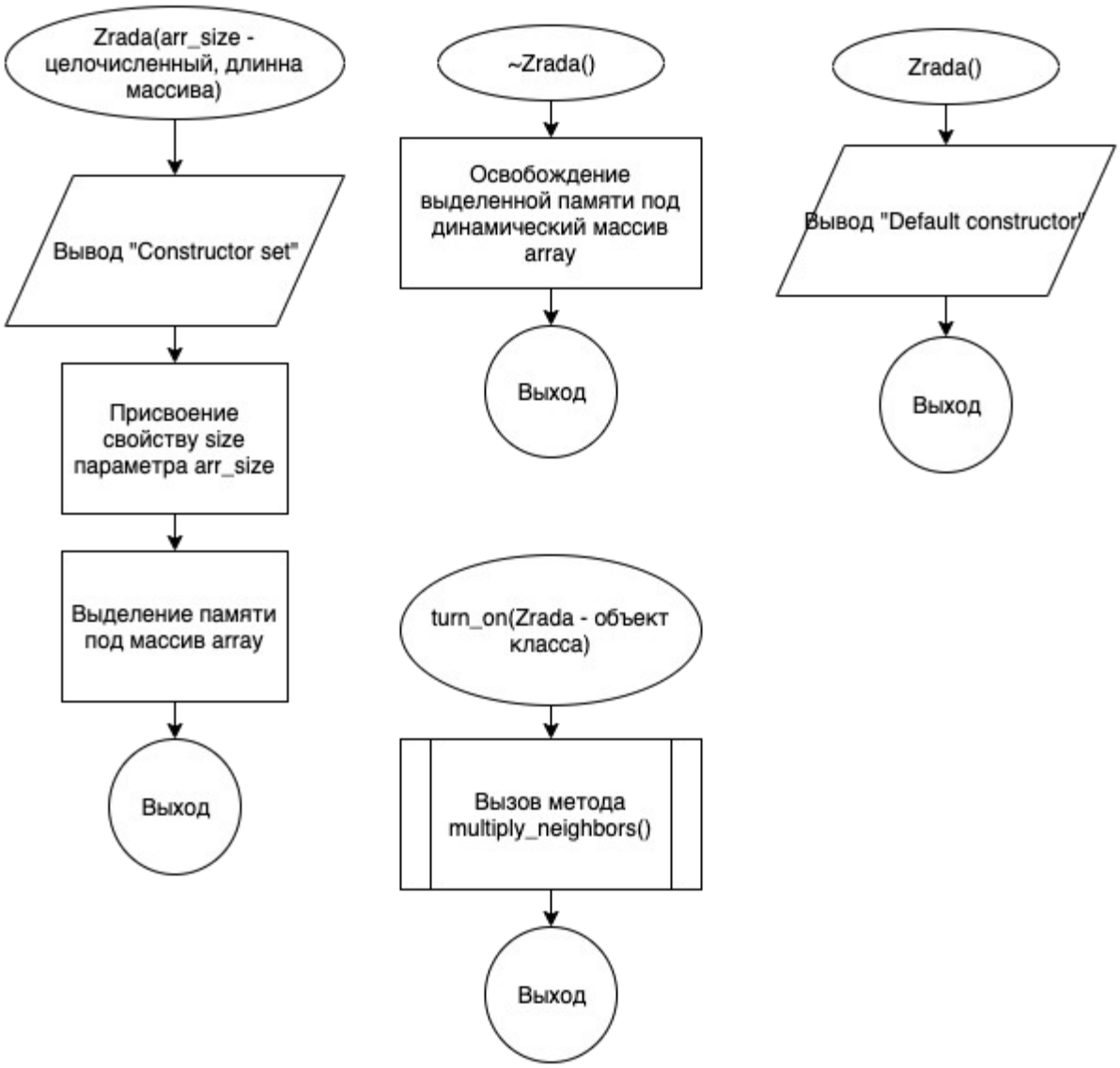

Рисунок 1 – Блок-схема алгоритма

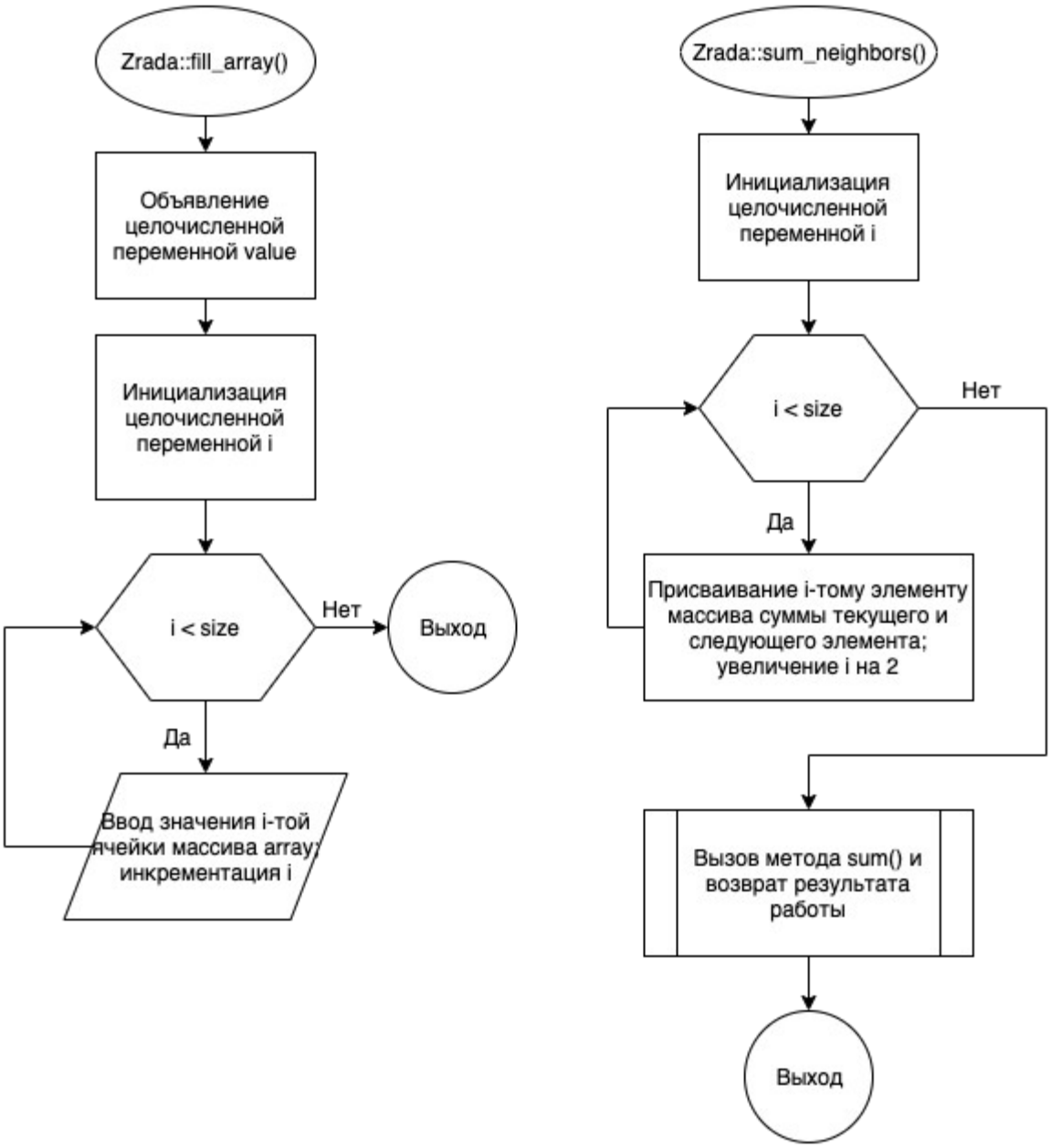

Рисунок 2 – Блок-схема алгоритма

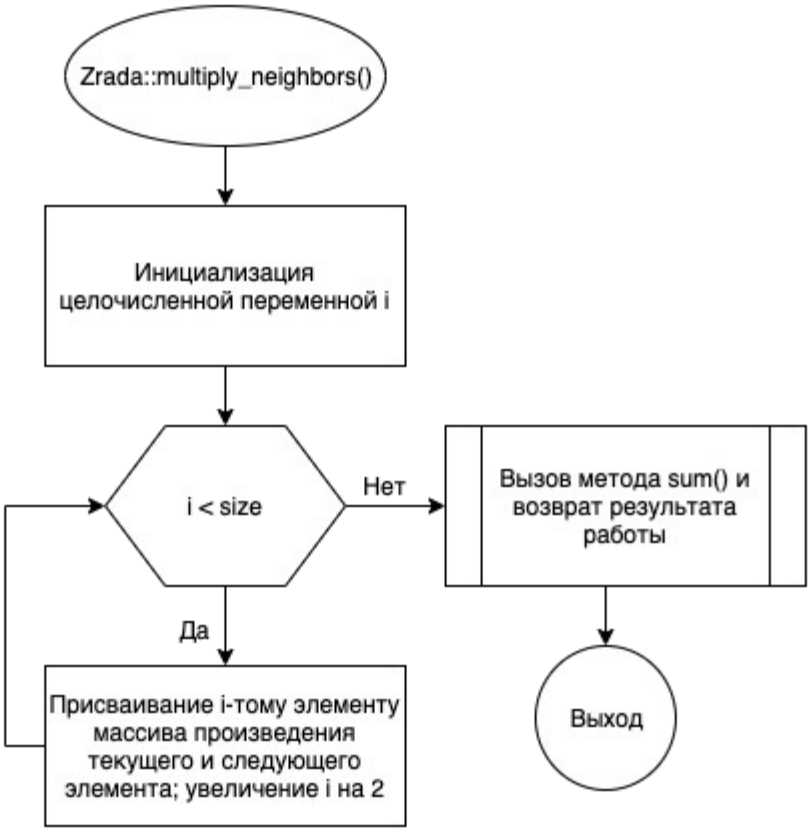

Рисунок 3 – Блок-схема алгоритма

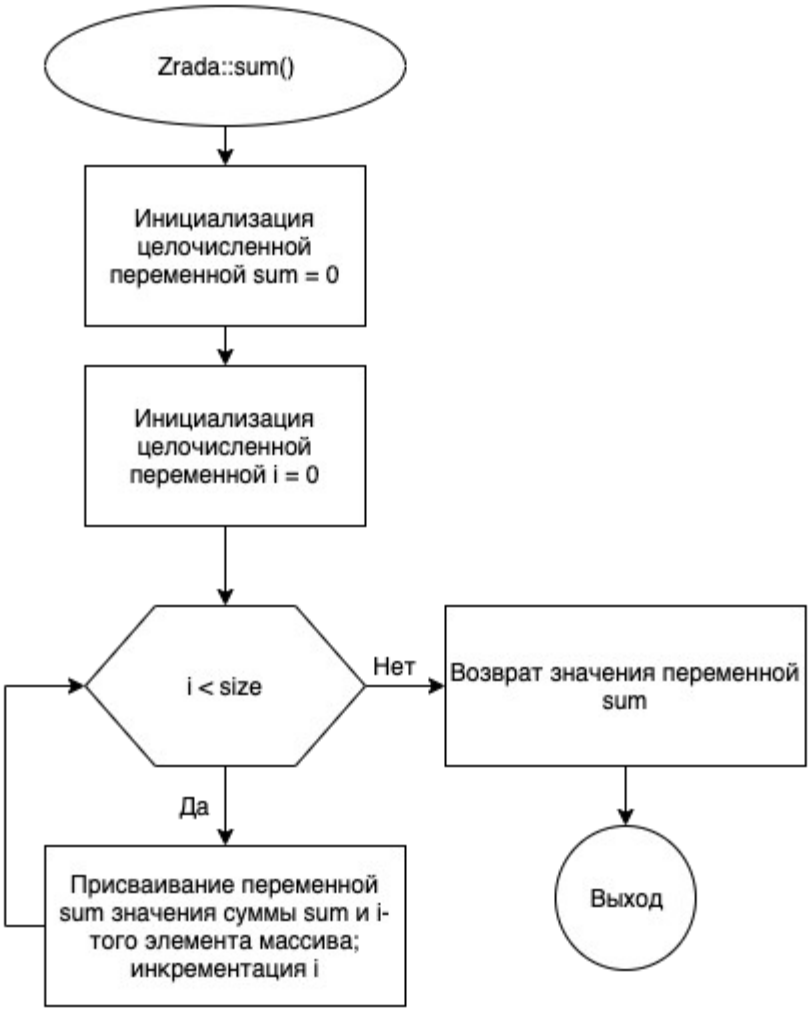

Рисунок 4 – Блок-схема алгоритма

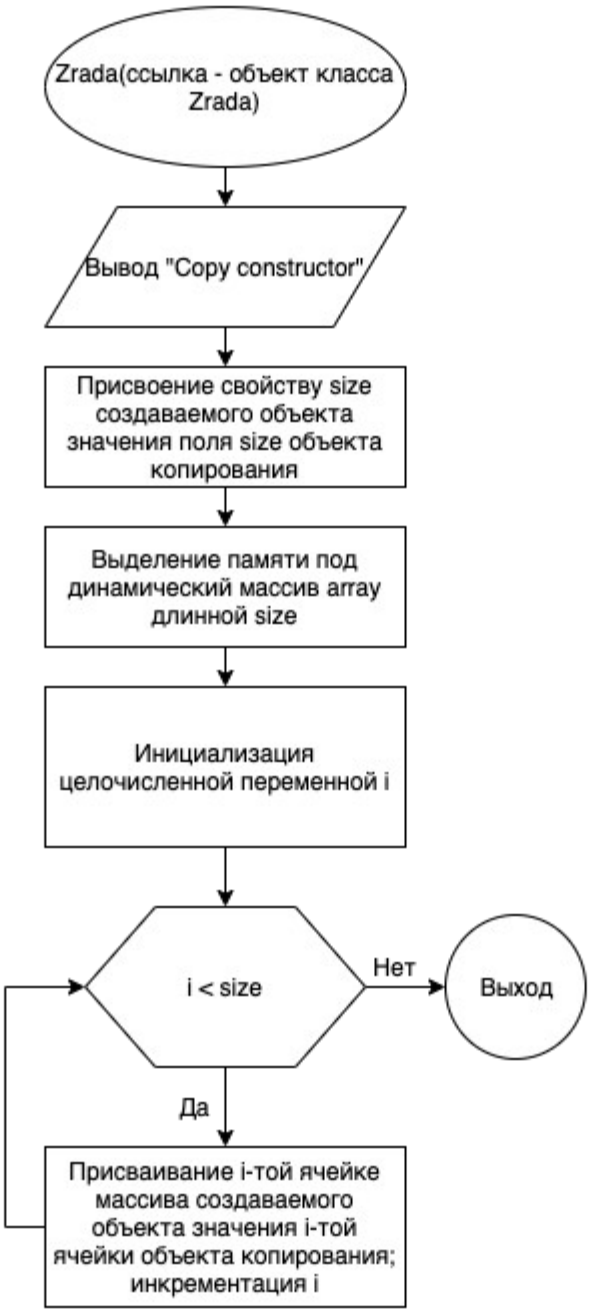

Рисунок 5 – Блок-схема алгоритма

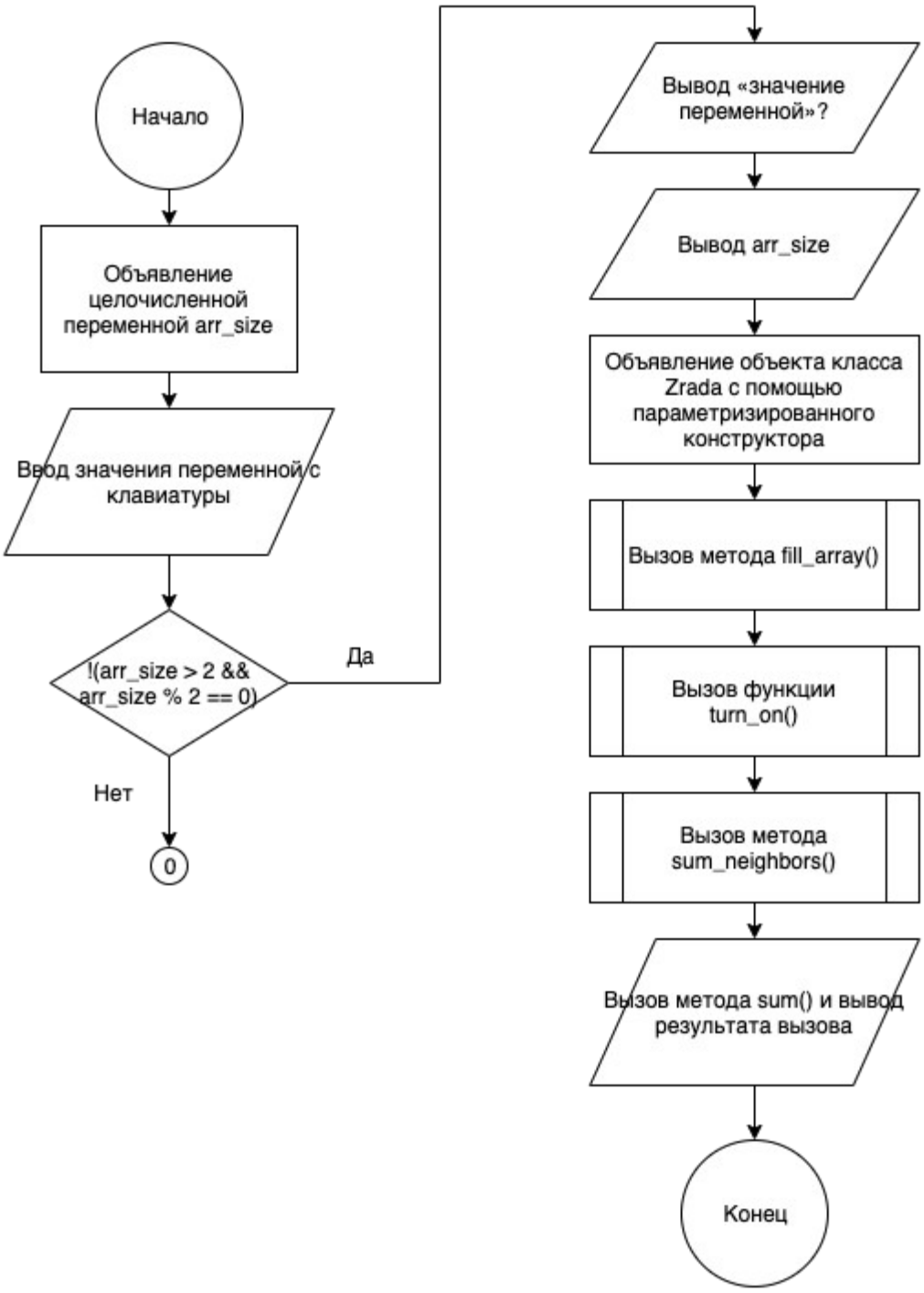

Рисунок 6 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

```
#include <iostream>
#include "Zrada.h"

void turn_on(Zrada peremoga) {
   std::cout << peremoga.multiply_neighbors() << std::endl;
}

int main() {

   int arr_size;
   std::cin >> arr_size;

   if (!(arr_size > 2 && arr_size % 2 == 0)) {
      std::cout << arr_size << "?";
      return 0;
   }

   std::cout << arr_size << std::endl;

   Zrada zrada(arr_size);

   zrada.fill_array();
   turn_on(zrada);
   zrada.sum_neighbors();
   std::cout << zrada.sum() << std::endl;

   return 0;
}
```

5.2 Файл Zrada.cpp

Листинг 2 – Zrada.cpp

```
#include "Zrada.h"
```

```
#include <iostream>

Zrada::Zrada(int arr_size) {
   std::cout << "Constructor set" << std::endl;
   size = arr_size;
   array = new int[size];
}

Zrada::Zrada() {
   std::cout << "Default constructor" << std::endl;
}

Zrada::Zrada(const Zrada &p_zrada) {
   std::cout << "Copy constructor" << std::endl;
   size = p_zrada.size;
   array = new int[size];

   for (int i = 0; i < size; i++) {
      array[i] = p_zrada.array[i];
   }
}

int Zrada::sum_neighbors() {

   for (int i = 0; i < size; i += 2) {
      array[i] += array[i + 1];
   }

   return sum();
}

int Zrada::multiply_neighbors() {
   for (int i = 0; i < size; i += 2) {
      array[i] *= array[i + 1];
   }
   return sum();
}

void Zrada::fill_array() {
   int value;
   for (int i = 0; i < size; i++) {
      std::cin >> value;
      array[i] = value;
   }
}

Zrada::~Zrada() {
   std::cout << "Destructor" << std::endl;
   delete[] array;
}

int Zrada::sum() {
   int sum = 0;
   for (int i = 0; i < size; i++) {
      sum += array[i];
```

```
    }
    return sum;
}
```

5.3 Файл Zrada.h

Листинг 3 – Zrada.h

```
#ifndef __ZRADA_H
#define __ZRADA_H


class Zrada {
private:
   int *array, size;

public:
   Zrada(int arr_size);

   Zrada(const Zrada &p_zrada);

   Zrada();

   ~Zrada();

   void fill_array();

   int sum_neighbors();

   int multiply_neighbors();

   int sum();
};


#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 11.

Таблица 11 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
4 1 2 3 4	4 Constructor set Copy constructor 20 Destructor 16 Destructor	4 Constructor set Copy constructor 20 Destructor 16 Destructor
3	3?	3?
8 1 2 3 4 5 6 7 8	8 Constructor set Copy constructor 120 Destructor 56 Destructor	8 Constructor set Copy constructor 120 Destructor 56 Destructor

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).