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. По значению параметра определяется размерность целочисленного массива из закрытой области. Массив создается. 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. Размер должен иметь значение больше 2 и быть четным.
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 класса obj класса ArrayObject предназначен для хранения данных массива и выполнения операций с ними;
- объект cin класса потокового ввода предназначен для функционирования системы;
- объект cout класса потокового вывода предназначен для функционирования системы;
- функция main для основной алгоритм работы программы;
- функция array_func для вызов метода 'function_two()' объекта 'ArrayObject';
- оператор цикла;
- условный оператор;
- оператор выделения памяти new;
- оператор освобождения памяти delete[].

Класс ArrayObject:

- свойства/поля:
    - поле поле, отвечающие за хранение массива:
        - наименование — size_array_data;
        - тип — указатель на целочисленный массив;
        - модификатор доступа — private;
    - поле поле, отвечающие за хранение размера array_data:
        - наименование — size_array_data;
        - тип — целое число;
        - модификатор доступа — private;
- функционал:

- метод ArrayObject — конструктор;
- метод ArrayObject(int _size_array_data) : size_array_data(_size_array_data) — конструктор параметризированный конструктор создающий объект с указанной размерностью массива;
- метод ArrayObject(const ArrayObject& temporary_object) — конструктор копирования;
- метод ~ArrayObject — деструктор;
- метод read_elements — метод для ввода элементов массива;
- метод function_one — метод, который суммирует значения каждой пары элементов массива и возвращает сумму элементов массива после этой операции;
- метод function_two — метод, который умножает значения каждой пары элементов массива и возвращает сумму элементов массива после этой операции;
- метод calculate_sum_elements — метод, который вычисляет сумму всех 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 функции main

Функционал: основной алгоритм работы программы.

Параметры: отсутствуют.

Возвращаемое значение: int - индикатор корректности завершения работы программы.

Алгоритм функции представлен в таблице 1.

Таблица 1 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		объявление целочисленной переменной _size	2
2		ввод значения переменной _size с клавиатуры	3
3	_size <= 2 || _size % 2 != 0	вывод значения переменной '_size', вывод '?'	∅
			4
4		выводим значение переменной _size на экран	5
5		создание объекта класса 'ArrayObject' с именем 'obj' и размером массива, переданным в качестве параметра '_size'	6
6		вызов метода 'read_elements()' объекта 'obj', который позволяет пользователю ввести значение элементов массива	7
7		вызов функции 'array_func()', передавая ей объект 'obj' класса 'ArrayObject' в качестве аргумента	8
8		вывод на экран суммы элементов массива после	∅

№	Предикат	Действия	№ перехода
		метода 'function_one()' к объекту 'obj'	

3.2 Алгоритм функции array_func

Функционал: вызов метода 'function_two()'.

Параметры: объект класса 'ArrayObject'.

Возвращаемое значение: отсутствует.

Алгоритм функции представлен в таблице 2.

Таблица 2 – Алгоритм функции array_func

№	Предикат	Действия	№ перехода
1		определение функции 'array_func()', которая принимает объект класса 'ArrayObject' в качестве параметра	2
2		вызов метода function_two() для объекта 'temporary_object' класса 'ArrayObject'	∅

3.3 Алгоритм конструктора класса ArrayObject

Функционал: конструктор.

Параметры: отсутствуют.

Алгоритм конструктора представлен в таблице 3.

Таблица 3 – Алгоритм конструктора класса ArrayObject

№	Предикат	Действия	№ перехода
1		вывод на экран "Default constructor"	∅

3.4 Алгоритм деструктора класса ArrayObject

Функционал: деструктор.

Параметры: отсутствуют.

Алгоритм деструктора представлен в таблице 4.

Таблица 4 – Алгоритм деструктора класса ArrayObject

№	Предикат	Действия	№ перехода
1		определение деструктора класса "ArrayObject"	2
2		вывод на экран 'Destructor'	3
3		освобождение памяти, выделенную под массив данных 'data_array_size' используя оператор delete[]	∅

3.5 Алгоритм метода read_elements класса ArrayObject

Функционал: метод для ввода элементов массива.

Параметры: отсутствуют.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода read_elements класса ArrayObject

№	Предикат	Действия	№ перехода
1		определение метода 'read_elements()' класса 'ArrayObject'	2
2		инициализация целочисленной переменной i	3
3	i < size_array_data	++i; считывание значения, введеное с клавиатуры, сохранение его в i-том элемента массива данных 'data_array_size'	3
			∅

3.6 Алгоритм метода function_one класса ArrayObject

Функционал: метод, который суммирует значения каждой пары элементов массива и возвращает сумму элементов массива после этой операции.

Параметры: отсутствуют.

Возвращаемое значение: calculate_sum_elements - сумма элементов массива.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода function_one класса ArrayObject

№	Предикат	Действия	№ перехода
1		определение метода 'function_one()' класса 'ArrayObject'	2
2		инициализация целочисленной переменной i	3
3	i < size_array_data	i += 2; увеличение значение элемента массива на значение следующего элемента и сохранение результата обратно в первом элементе пары	3
			4
4		возврат целочисленного значения - суммы элементов массива после применения операций сложения к каждой паре элементов	∅

3.7 Алгоритм метода function_two класса ArrayObject

Функционал: метод, который умножает значения каждой пары элементов массива и возвращает сумму элементов массива после этой операции.

Параметры: отсутствуют.

Возвращаемое значение: calculate_sum_elements - сумма элементов массива.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода function_two класса ArrayObject

№	Предикат	Действия	№ перехода
1		определение метода 'function_two()' класса 'ArrayObject'	2
2		инициализация целочисленной переменной i	3
3	i < size_array_data	i += 2; умножение значений элемента массива на значение следующего элемента и сохранение результата обратно в первом элементе пары	4
			4
4		возврат целочисленного значения - суммы элементов массива после применения операций умножения к каждой паре элементов	∅

3.8 Алгоритм метода calculate_sum_elements класса ArrayObject

Функционал: метод, который вычисляет сумму всех элементов массива.

Параметры: отсутствуют.

Возвращаемое значение: sum - сумма всех элементов массива.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода calculate_sum_elements класса ArrayObject

№	Предикат	Действия	№ перехода
1		определение метода 'calculate_sum_elements()' класса 'ArrayObject'	2
2		инициализация целочисленной переменной sum; sum = 0	3
3		инициализация целочисленной переменной i	4
4	i < size_array_data	i++; добавление значения элемента массива к текущей сумме, которая хранится в переменной	4

№	Предикат	Действия	№ перехода
		'sum'	
		увеличение значения переменной 'sum' на значение элемента массива с индексом 'i'	5
5		вовзрат значения переменной 'sum', которое представляет собой сумму всех элементов массива	∅

3.9 Алгоритм метода ArrayObject(const ArrayObject& temporary_object) класса ArrayObject

Функционал: конструктор копирования.

Параметры: ArrayObject& temporary_object - ссылка на объект класса.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода ArrayObject(const ArrayObject& temporary_object) класса ArrayObject

№	Предикат	Действия	№ перехода
1		определение конструктора копирования для класса 'ArrayObject', который создает новый объект, идентичный объекту 'temporary_object', путем копирование его полей	2
2		значение поля 'size_array_data' нового объекта 'ArrayObject' устанавливается равным значению поля 'size_array_data' объекта 'temporary_object'	3
3		выделение динамической памяти для нового массива 'data_array_size' с размером, равным 'size_array_data'	4
4		инициализация целочисленной переменной i	5
5	i < size_array_data	i++; копирование значения элементов массива	5

№	Предикат	Действия	№ перехода
		'data_array_size' объекта 'temporary_object' в массив 'data_array_size' нового объекта 'ArrayObject'	
			6
6		вывод на экран "Copy constructor"	∅

3.10 Алгоритм метода ArrayObject(int _size_array_data) : size_array_data(_size_array_data) класса ArrayObject

Функционал: конструктор параметризированный конструктор создающий объект с указанной размерностью массива.

Параметры: int _size_array_data - размерность массива данных.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода ArrayObject(int _size_array_data) : size_array_data(_size_array_data) класса ArrayObject

№	Предикат	Действия	№ перехода
1		инициализация поля 'size_array_data' значением '_size_array_data', переданным в конструктор, используя инициализацию членов	2
2		выделение динамической памяти для массива 'data_array_size' размером 'size_array_data' элементов типа 'int'	3
3		вывод на экран "Constructor set"	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-8.

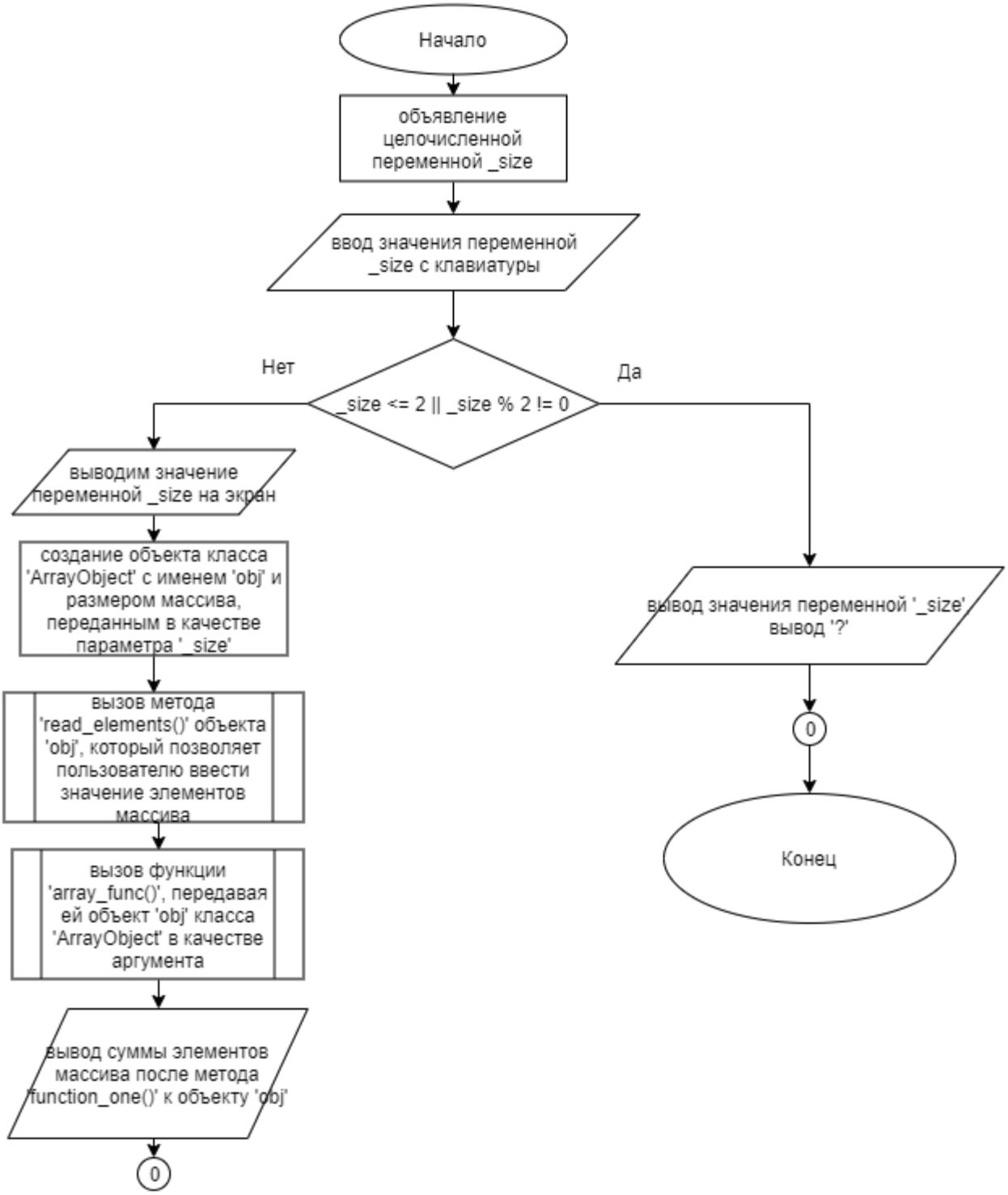

Рисунок 1 – Блок-схема алгоритма

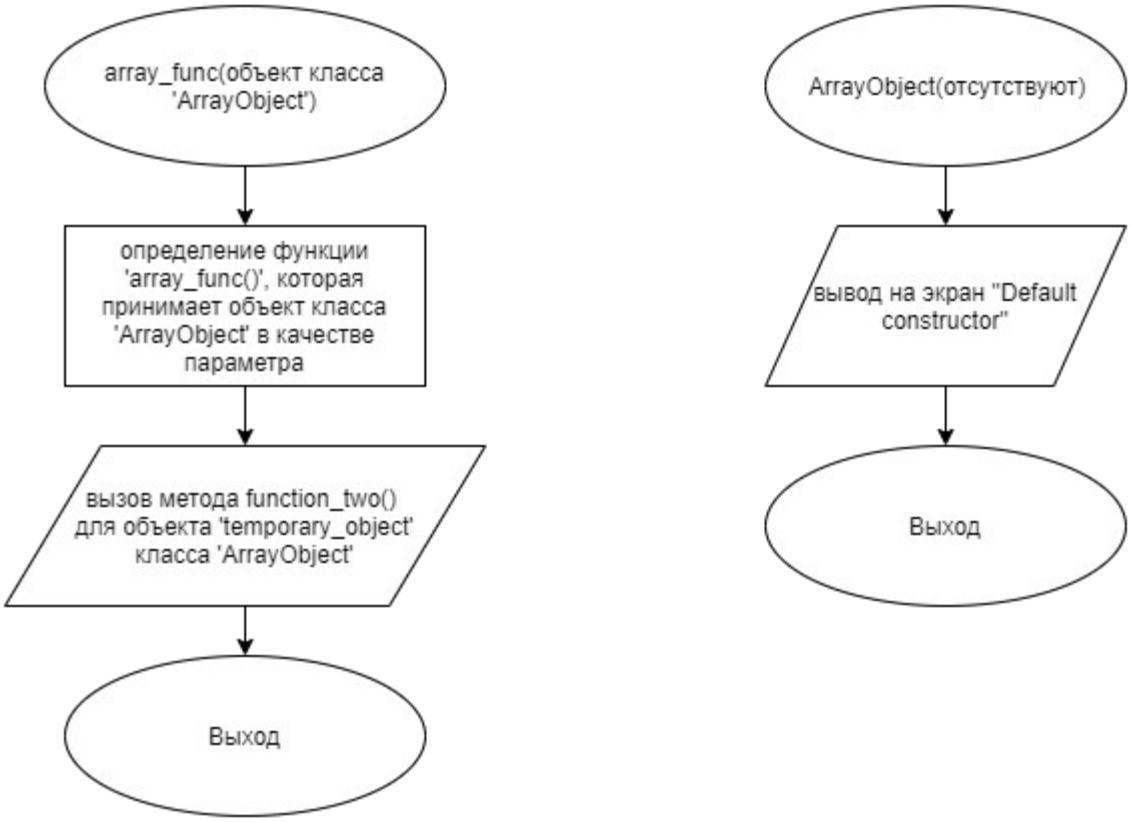

Рисунок 2 – Блок-схема алгоритма

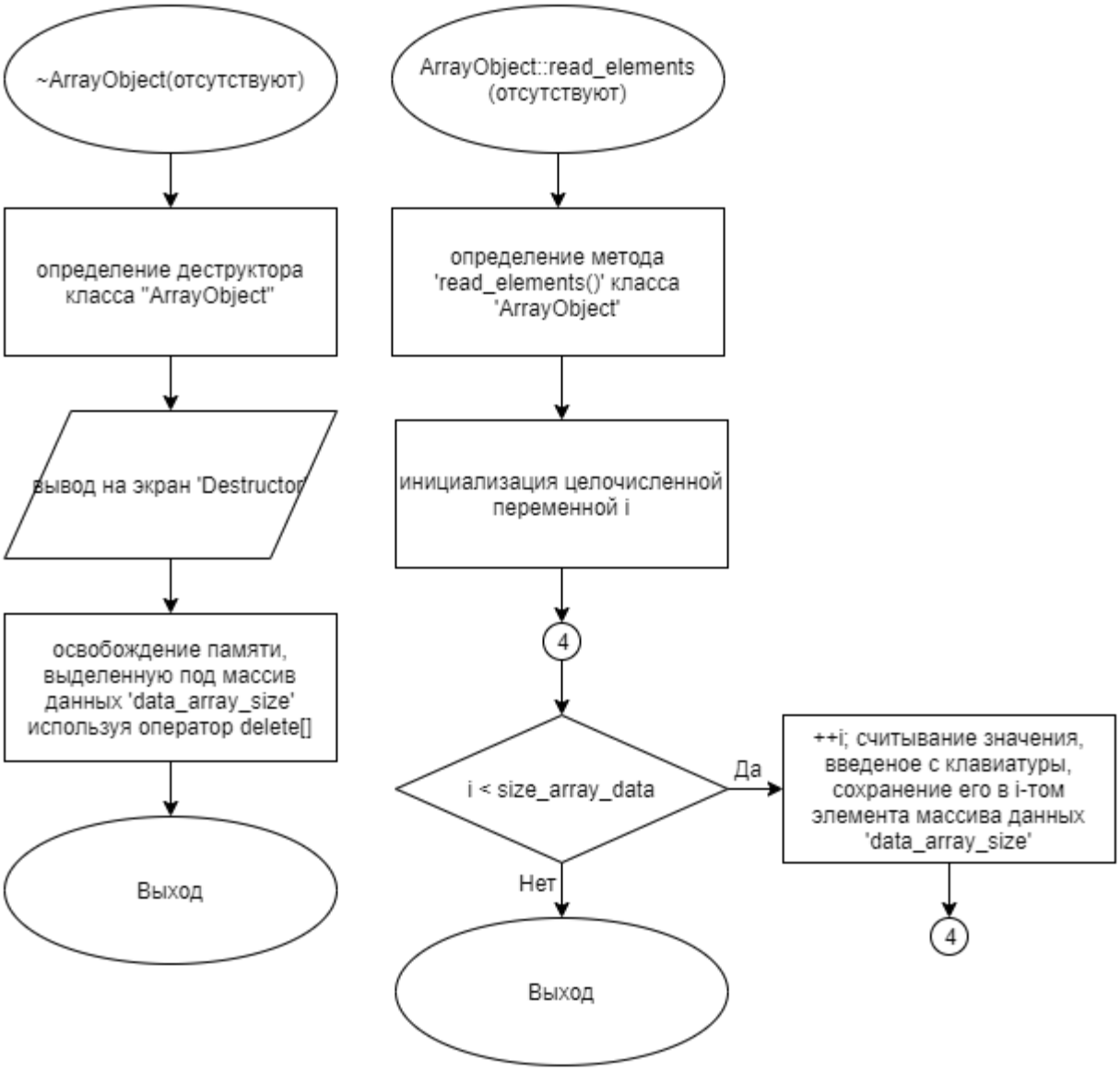

Рисунок 3 – Блок-схема алгоритма

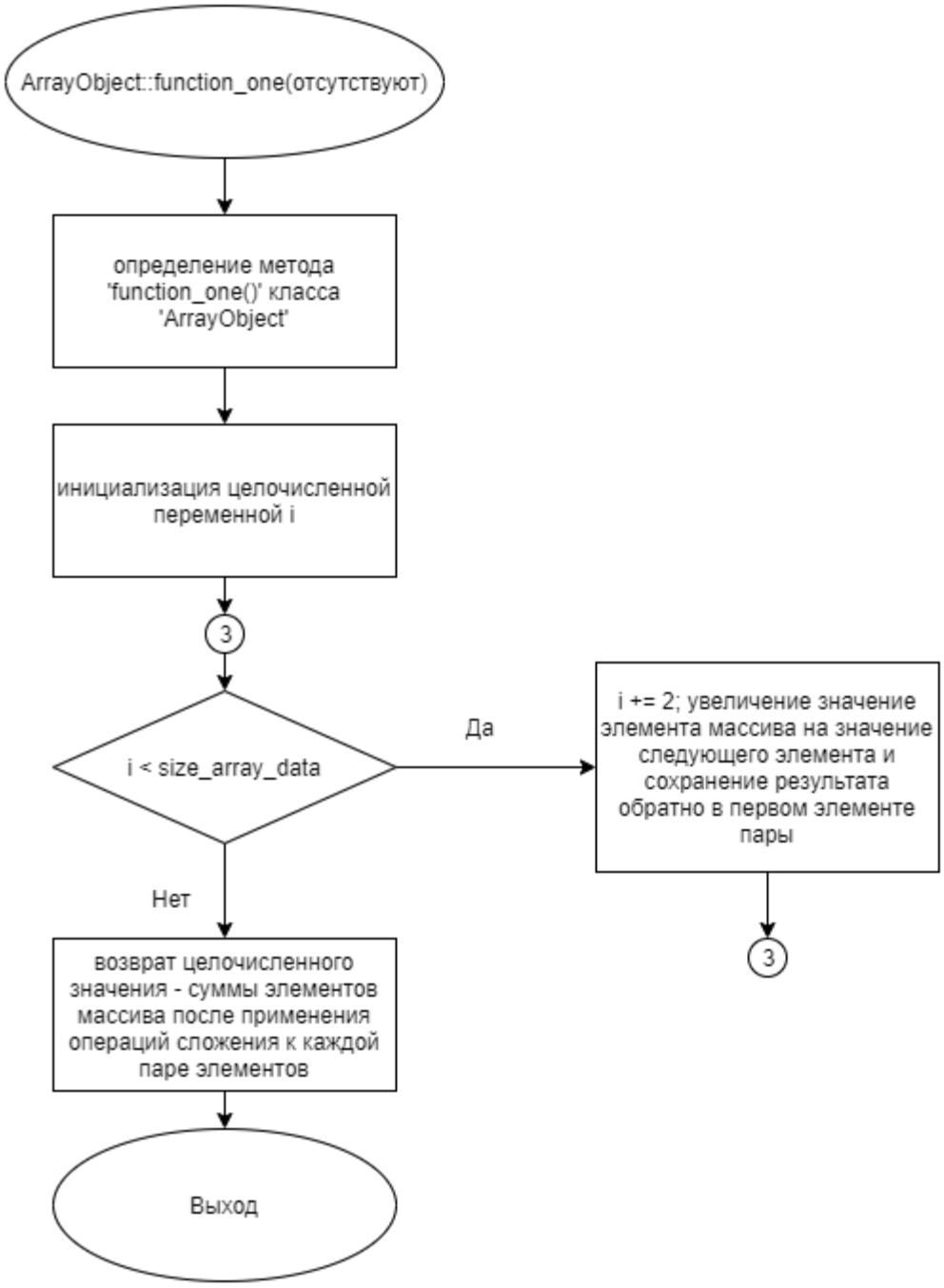

Рисунок 4 – Блок-схема алгоритма

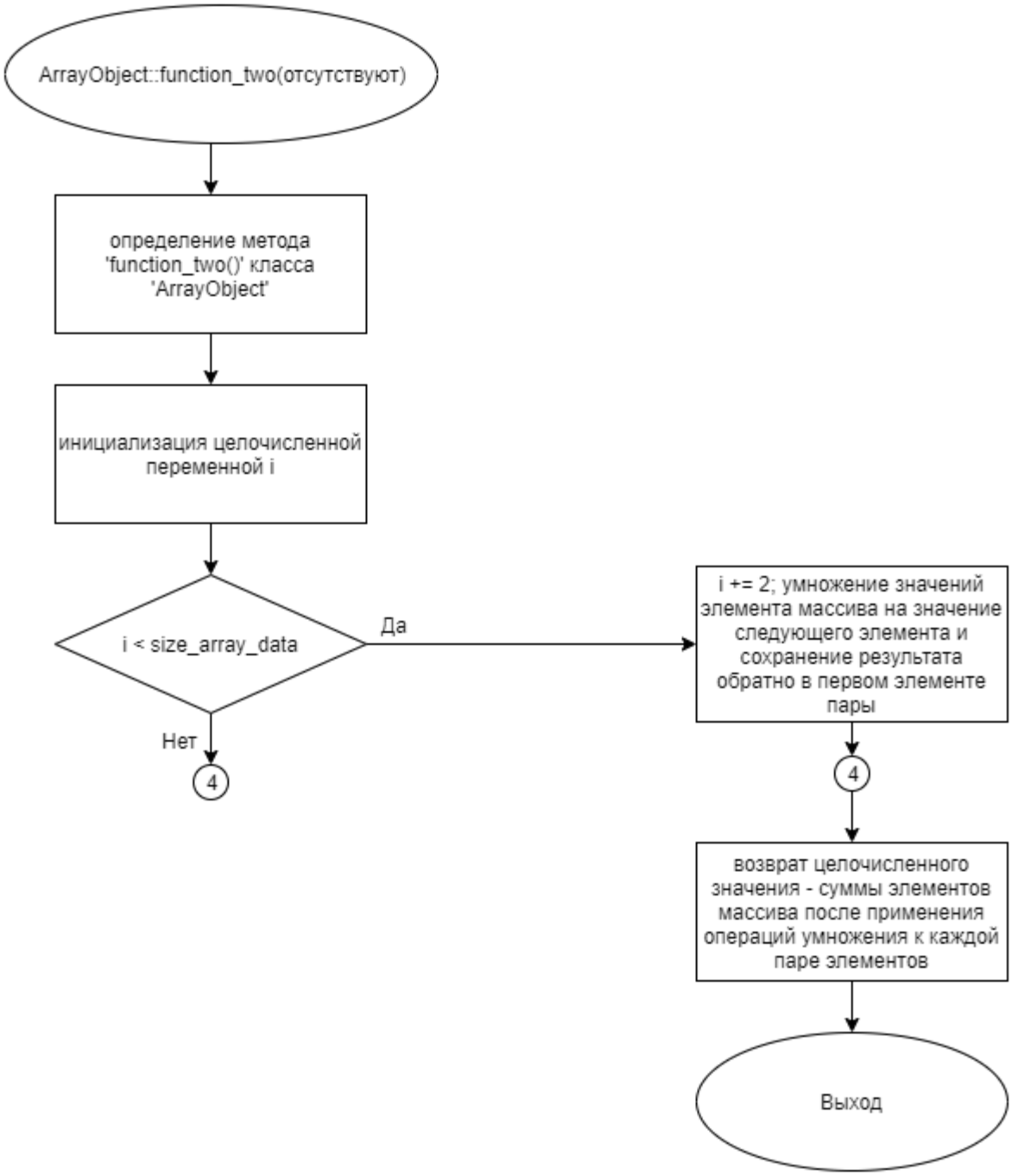

Рисунок 5 – Блок-схема алгоритма

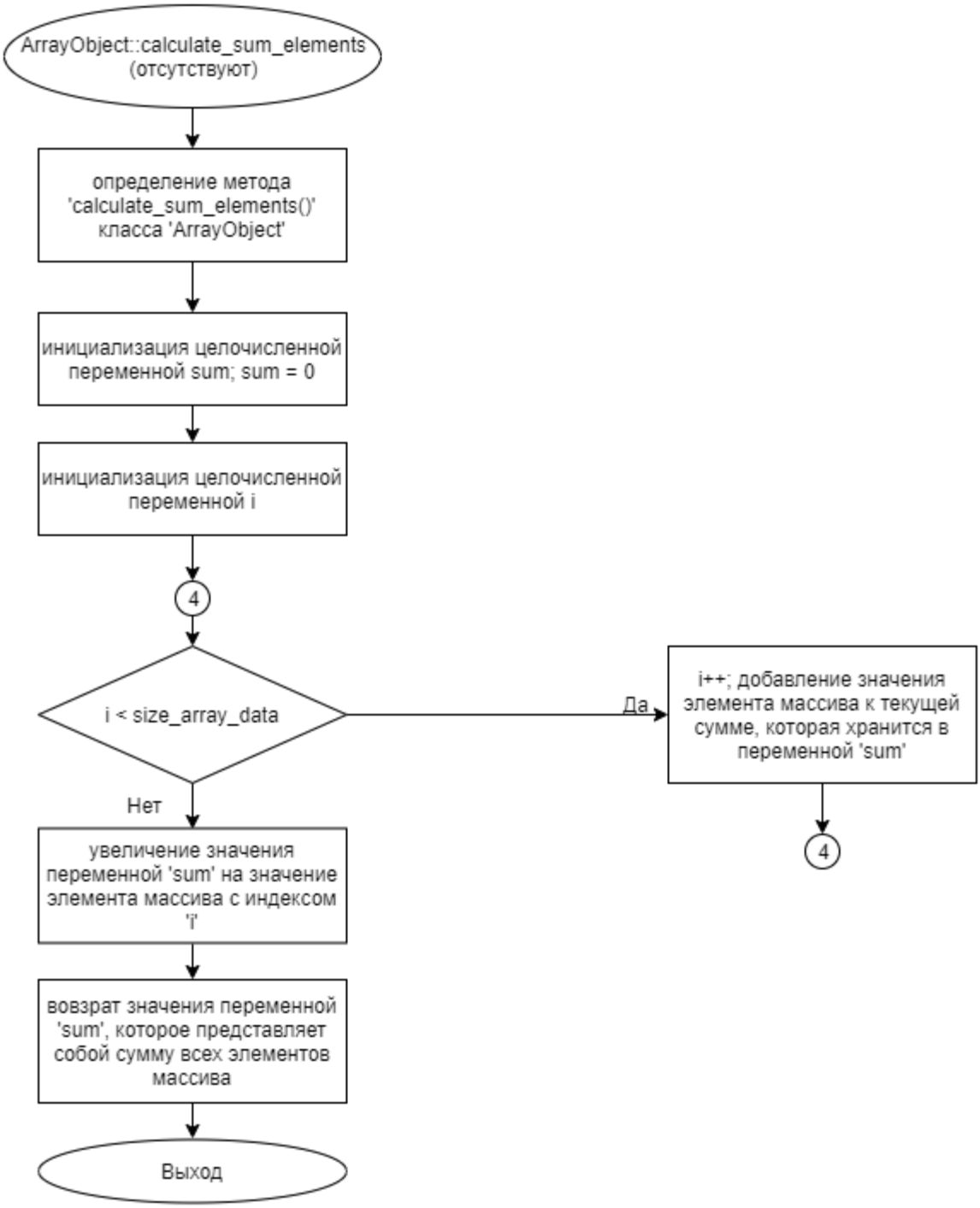

Рисунок 6 – Блок-схема алгоритма

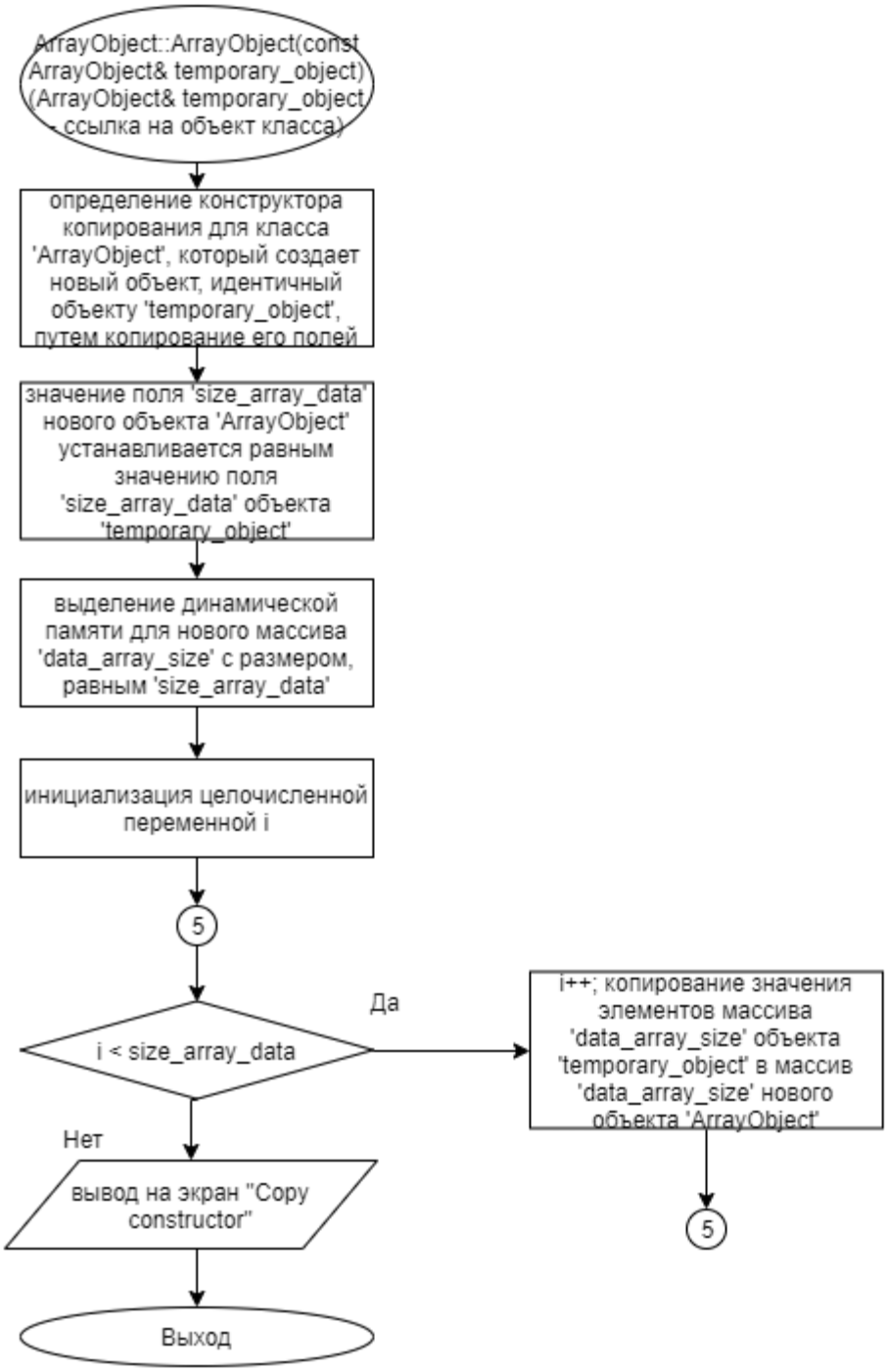

Рисунок 7 – Блок-схема алгоритма

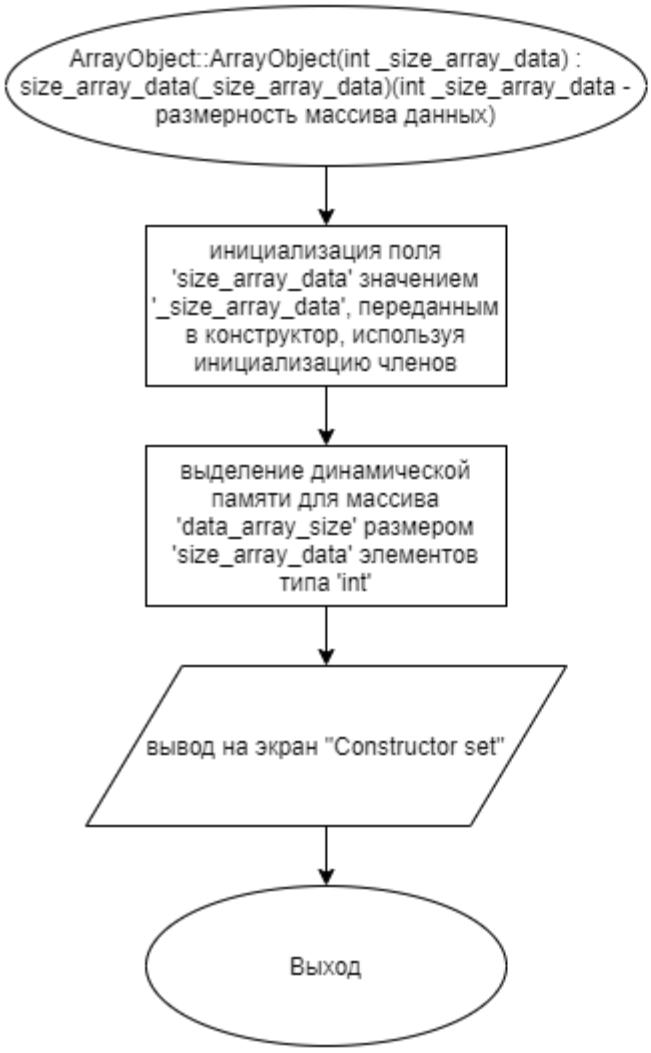

Рисунок 8 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл ArrayObject.cpp

Листинг 1 – ArrayObject.cpp

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

using namespace std;

ArrayObject::ArrayObject(int                _size_array_data)               :
size_array_data(_size_array_data) {
   data_array_size = new int[size_array_data];
   cout << "Constructor set" << endl;
}

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

ArrayObject::ArrayObject(const ArrayObject& temporary_object) {
   size_array_data = temporary_object.size_array_data;
   data_array_size = new int[size_array_data];
   for(int i = 0; i < size_array_data; i++) {
      data_array_size[i] = temporary_object.data_array_size[i];
   }
   cout << "Copy constructor" << endl;
}

ArrayObject::~ArrayObject() {
   cout << "Destructor" << endl;
   delete[] data_array_size;
}

void ArrayObject::read_elements(){
   //cout << "Enter " << size_array_data << " elements:" << endl;
   for(int i = 0; i < size_array_data; ++i) {
      cin >> data_array_size[i];
   }
}

int ArrayObject::function_one() {
   for(int i = 1; i < size_array_data; i += 2) {
      //data_array_size[i - 1] = data_array_size[i - 1] + data_array_size[i];
```

```
      data_array_size[i - 1] += data_array_size[i];
   }
   return calculate_sum_elements();
}

int ArrayObject::function_two() {
   for(int i = 1; i < size_array_data; i += 2) {
      //data_array_size[i - 1] = data_array_size[i - 1] * data_array_size[i];
      data_array_size[i - 1] *= data_array_size[i];
   }
   return calculate_sum_elements();
}

int ArrayObject::calculate_sum_elements() {
   int sum = 0;
   for(int i = 0; i < size_array_data; ++i) {
      sum += data_array_size[i];
   }
   return sum;
}

void array_func(ArrayObject temporary_object) {
   cout << temporary_object.function_two() << endl;
   //cout << temporary_object.function_two() << endl;
}
```

5.2 Файл ArrayObject.h

Листинг 2 – ArrayObject.h

```
#ifndef __ARRAYOBJECT__H
#define __ARRAYOBJECT__H

#include <iostream>

class ArrayObject {
public:
   ArrayObject();
   ArrayObject(int _size_array_data);
   ArrayObject(const ArrayObject& temporary_object);
   ~ArrayObject();

   void read_elements();
   int function_one();
   int function_two();
   int calculate_sum_elements();
private:
   int size_array_data;
   int* data_array_size;
};
```

```
void array_func(ArrayObject temporary_object);

#endif
```

5.3 Файл main.cpp

Листинг 3 – main.cpp

```
#include "ArrayObject.h"
#include <stdlib.h>
#include <stdio.h>

using namespace std;

int main() {
   int _size;
   cin >> _size;

   if(_size <= 2 || _size % 2 != 0) {
      cout << _size << "?" << endl;
      return 0;
   }
   cout << _size << endl;
   ArrayObject obj(_size);

   obj.read_elements();

   array_func(obj);
   cout << obj.function_one() << endl;
   return(0);
}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 11.

Таблица 11 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
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
4 1 2 3 4	4 Constructor set Copy constructor 20 Destructor 16 Destructor	4 Constructor set Copy constructor 20 Destructor 16 Destructor
6 1 2 3 4 5 6	6 Constructor set Copy constructor 56 Destructor 33 Destructor	6 Constructor set Copy constructor 56 Destructor 33 Destructor
2	2?	2?
3	3?	3?

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).