Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Разработать систему, которая демонстрирует возможность использования дружественной функции.

Спроектировать объект, с свойствами в закрытом доступе:

- целого типа, для хранения размерности массива;
- указатель на объект целого типа;
- строкового типа, для хранения наименования объекта.

С параметризированным конструктором. У конструктора есть параметр целого типа. Параметр передает (содержит) значение размерности целочисленного массива. В конструкторе создается целочисленный массив заданной размерности. Вводится и выводиться значение наименования объекта. Вводится и выводится значения элементов.

Объект имеет метод, который возвращает сумму элементов целочисленного массива.

В деструкторе, первоначально выводится значение наименования объекта, а далее значения элементов целочисленного массива и освобождается память, выделенная для массива.

Спроектировать функцию, которая значения элементов массива одного объекта присвоит к элементам массива другого объекта.

Алгоритм конструирования и отработки системы:

1. Объявляется целочисленная переменная, для хранения значения количества объектов.
2. Объявляется целочисленная переменная, для хранения значения размерности массива.
3. Объявляется строковая переменная, для хранения наименования объекта.
4. Могут быть другие объявления.

5. Вводится значение количества объектов.
6. Вводится значение размерности массива.
7. В цикле создаются объекты, согласно введенному количеству.
8. Определяется значение суммы элементов для каждого объекта. Фиксируется объект, с первой минимальной суммой. Этот объект принимается за эталон.
9. В цикле, посредством последовательного вызова дружественной функции значения элементов массива эталонного объекта присваиваются элементам всех остальных объектов.
10. После завершения цикла, созданные объекты удаляются (уничтожаются).

1.1 Описание входных данных

Первая строка:

`«целое число, количество объектов»`

Вторая строка:

`«целое число, размерность массива»`

Начиная с третей строки, имя очередного объекта и значения элементов массивов, согласно количеству объектов:

`«строка» «целое число» «целое число» . . . «целое число»`

Количество целых чисел в этих строках больше или равно количеству размерности массива.

Пример ввода.

```
5
5
obj_2 2 2 2 2 2 2 2
```

```
obj_3 3 3 3 3 3 3 3 3 3
obj_1 1 1 1 1 1
obj_4 4 4 4 4 4 4 4 4
obj_5 5 5 5 5 5
```

1.2 Описание выходных данных

С первой строки, построчно, для каждого объекта:

«строка» «целое число» «целое число» . . . «целое число»

Имя объекта и значения элементов массива, согласно последовательности создания объектов.

Далее, построчно, для каждого объекта:

«имя объекта»: «целое число» «целое число» . . . «целое число»

Имя объекта и значения элементов массива, согласно последовательности создания объектов.

Пример вывода.

```
obj_2  2  2  2  2  2
obj_3  3  3  3  3  3
obj_1  1  1  1  1  1
obj_4  4  4  4  4  4
obj_5  5  5  5  5  5
obj_2: 1 1 1 1 1
obj_3: 1 1 1 1 1
obj_1: 1 1 1 1 1
obj_4: 1 1 1 1 1
obj_5: 1 1 1 1 1
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- функция main для определения входной точки программы;
- функция CopyArray для копирования элементов одного массива в другой;
- библиотека ввода-вывода;
- указатели;
- заголовочный файл;
- динамический массив.

Класс MyArray:

- свойства/поля:
    - поле хранит размер массива:
        - наименование — m_size;
        - тип — int;
        - модификатор доступа — private;
    - поле динамичесский массив:
        - наименование — m_arr;
        - тип — int*;
        - модификатор доступа — private;
    - поле хранит имя объекта:
        - наименование — m_name;
        - тип — std::string;
        - модификатор доступа — private;
- функционал:
    - метод MyArray — параметризированный конструктор;
    - метод Sum — возвращает сумму элементов массива;
    - метод ~MyArray — деструктор.

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 MyArray

Функционал: параметризированный конструктор.

Параметры: int size.

Алгоритм конструктора представлен в таблице 1.

Таблица 1 – Алгоритм конструктора класса MyArray

№	Предикат	Действия	№ перехода
1		выделение памяти для целочисленного динамического массива, размерностью равной параметру size с помощью оператора new	2
2		вызов метода ignore у объекта std::cin	3
3		ввод значения строковой переменной m_name	4
4		вывод значения строковой переменной m_name	5
5		инициализация целочисленной переменной i = 0	6
6	i < size	ввод значения m_arr[i]	7
			10
7		вывод двух пробельных символов	8
8		вывод значения m_arr[i]	9
9		увеличение значения переменной i на 1	6
10		вывод символа переноса строки	∅

3.2 Алгоритм метода Sum класса MyArray

Функционал: возвращает сумму элементов массива.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода Sum класса MyArray

№	Предикат	Действия	№ перехода
1		инициализация целочисленной переменной sum = 0	2
2		инициализация целочисленной переменной i = 0	3
3	i < m_size	прибавление значения m_arr[i] к значению целочисленной переменной sum	4
			5
4		увеличение значения переменной i на 1	3
5		возврат значения целочисленной переменной sum	∅

3.3 Алгоритм деструктора класса MyArray

Функционал: деструктор.

Параметры: нет.

Алгоритм деструктора представлен в таблице 3.

Таблица 3 – Алгоритм деструктора класса MyArray

№	Предикат	Действия	№ перехода
1		вывод m_name ": "	2
2		инициализация целочисленной переменной i = 0	3
3	i < m_size	вывод значения m_arr[i]	4
			6
4	i != m_size - 1	вывод пробельного символа	5

№	Предикат	Действия	№ перехода
			5
5		увеличение значения переменной i на 1	3
6		вывод символа переноса строки	7
7		освобождение памяти указателя m_arr с помощью оператора delete	∅

3.4 Алгоритм функции CopyArray

Функционал: копирует элементы одного массива в другой.

Параметры: нет.

Возвращаемое значение: MyArray* oldArray, MyArray* newArray.

Алгоритм функции представлен в таблице 4.

Таблица 4 – Алгоритм функции CopyArray

№	Предикат	Действия	№ перехода
1		инициализация целочисленной переменной i = 0	2
2	i < oldArray->m_size	newArray->m_arr[i] = oldArray->m_arr[i]	3
			∅
3		увеличение значения переменной i на 1	2

3.5 Алгоритм функции main

Функционал: основная функция программы.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм функции представлен в таблице 5.

Таблица 5 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		объявление двух целочисленных переменных objCount, size	2
2		ввод значений переменных objCount, size	3
3		выделение памяти под динамический массив objects указателей на объекты размером size с помощью оператора new	4
4		инициализация целочисленной переменной i = 0	5
5	i < objCount	создание объекта класса MyArray с помощью оператора new и присваивание его objects[i]	6
			7
6		увеличение значения переменной i на 1	5
7		инициализация указателя на объект класса MyArray адресом объекта objects[0]	8
8		инициализация целочисленной переменной i = 0	9
9	i < objCount		10
			12
10	objects[i]->Sum() < minSumObj->Sum()	minSumObj = objects[i]	11
			11
11		увеличение значения переменной i на 1	9
12		инициализация целочисленной переменной i = 0	13
13	i < objCount	вызов CopyArray(minSumObj, objects[i])	14
			15
14		увеличение значения переменной i на 1	13
15		инициализация целочисленной переменной i = 0	16
16	i < objCount	удаление объекта objects[i] с помощью оператора delete с освобождением памяти указателя	16
			∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

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

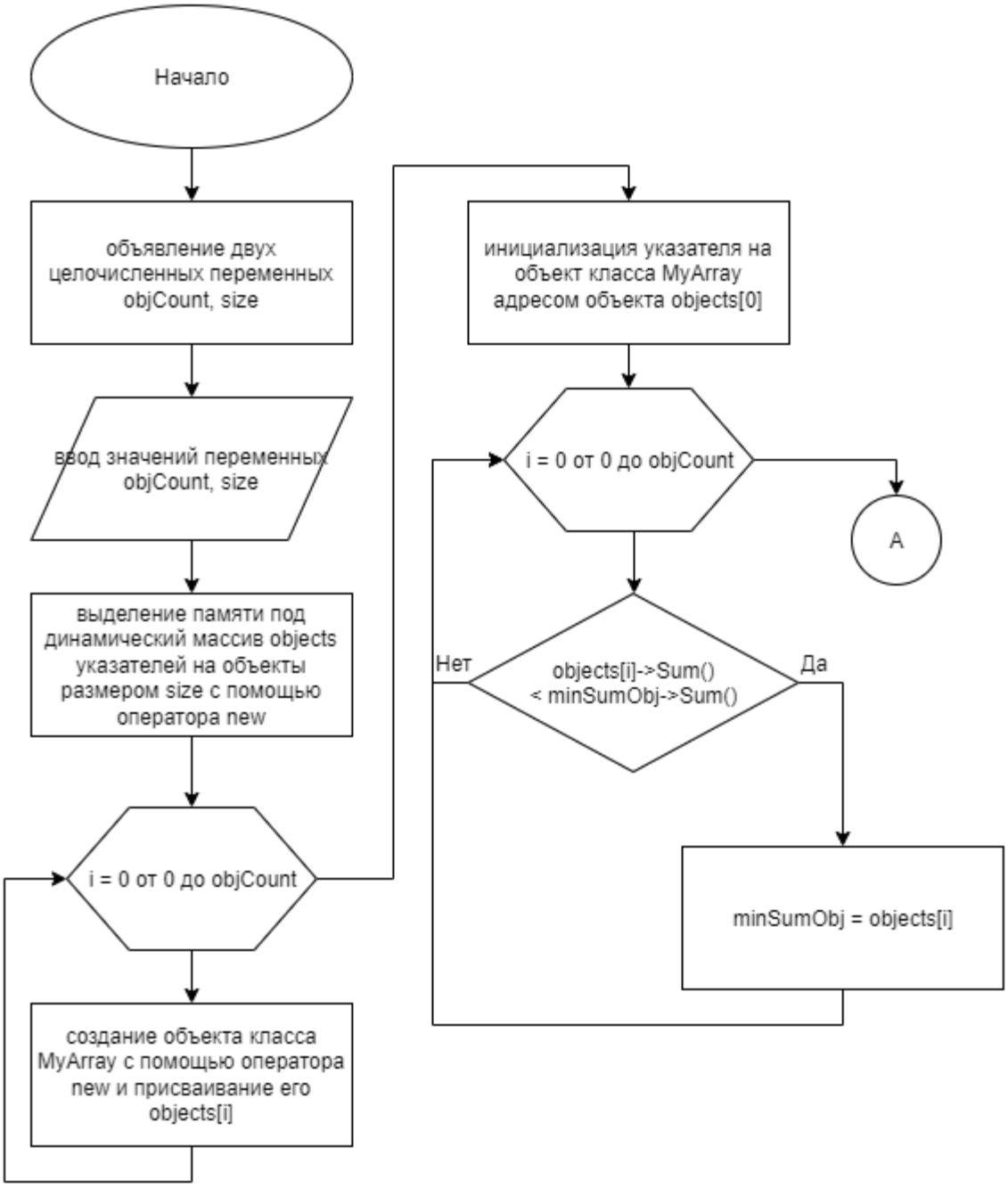

Рисунок 1 – Блок-схема алгоритма

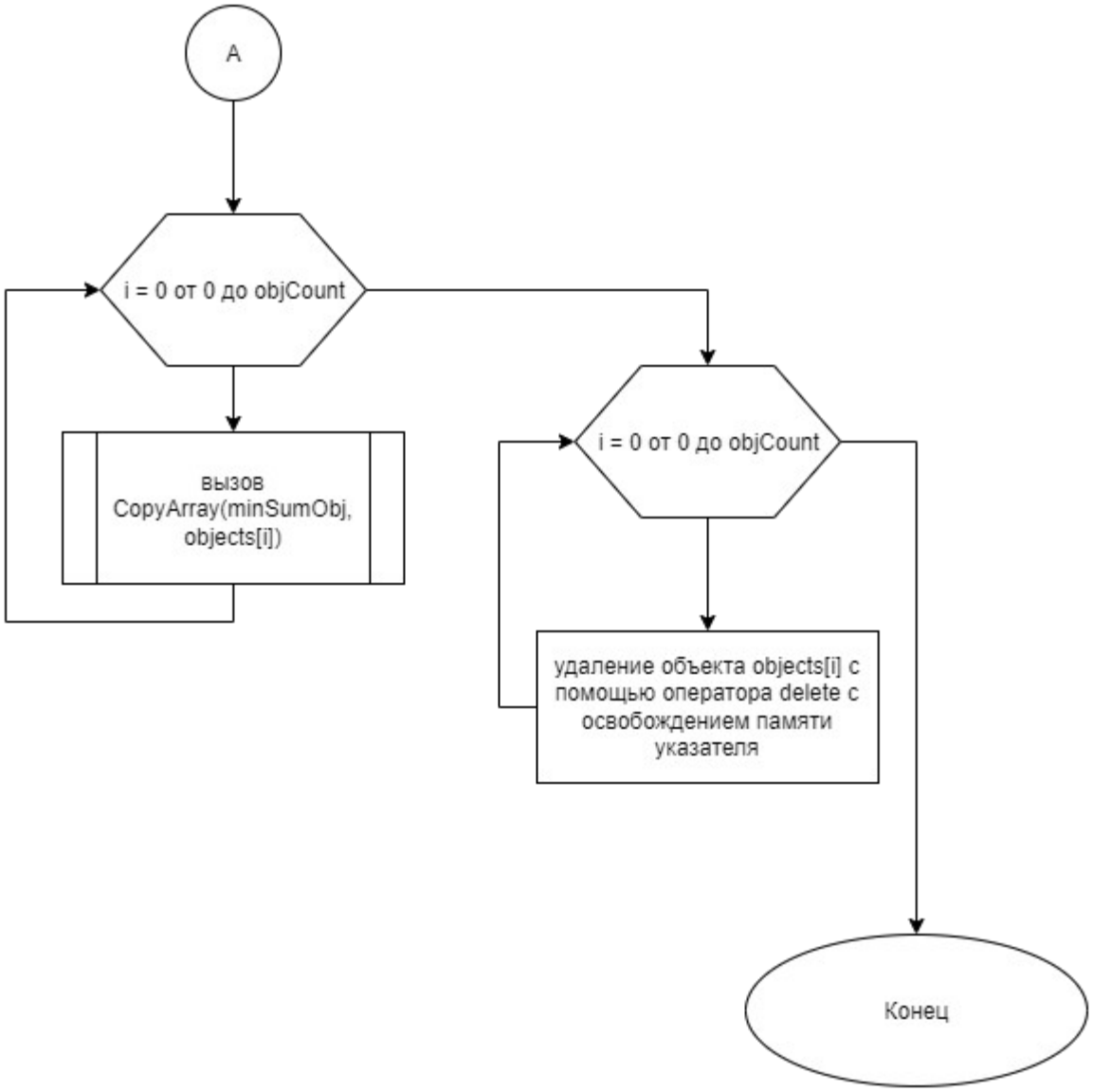

Рисунок 2 – Блок-схема алгоритма

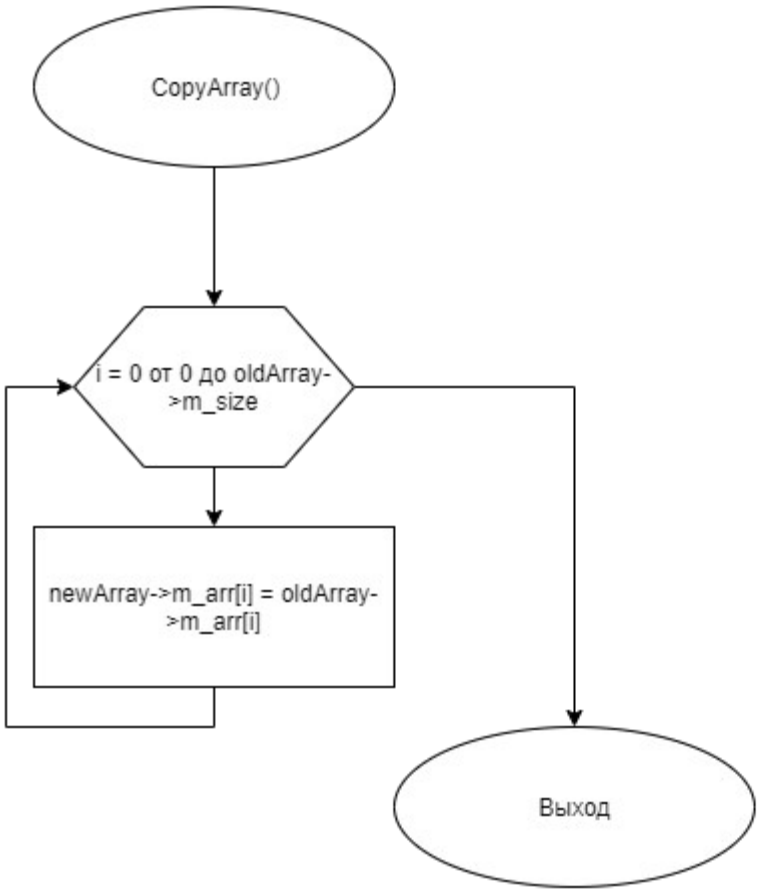

Рисунок 3 – Блок-схема алгоритма

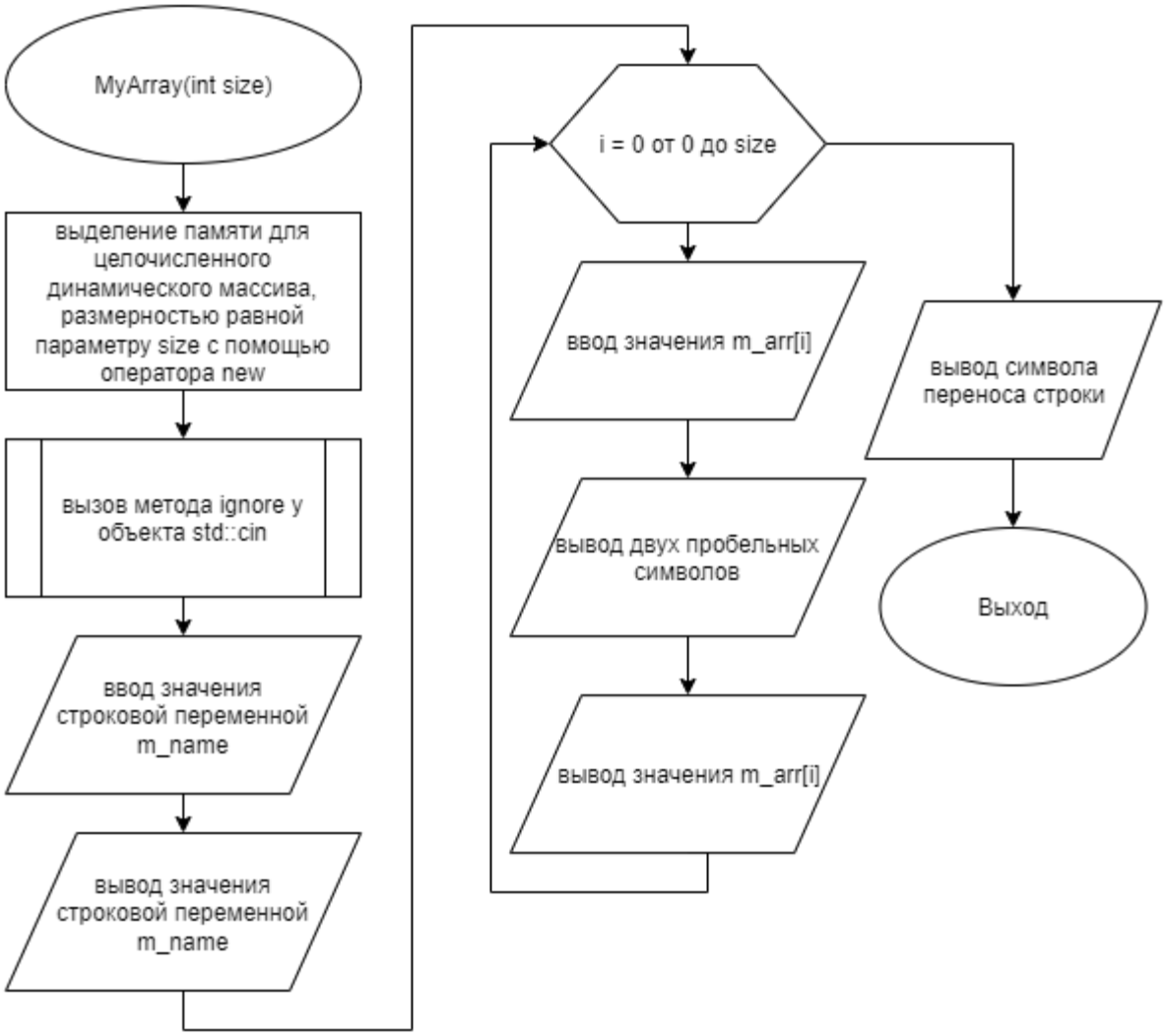

Рисунок 4 – Блок-схема алгоритма

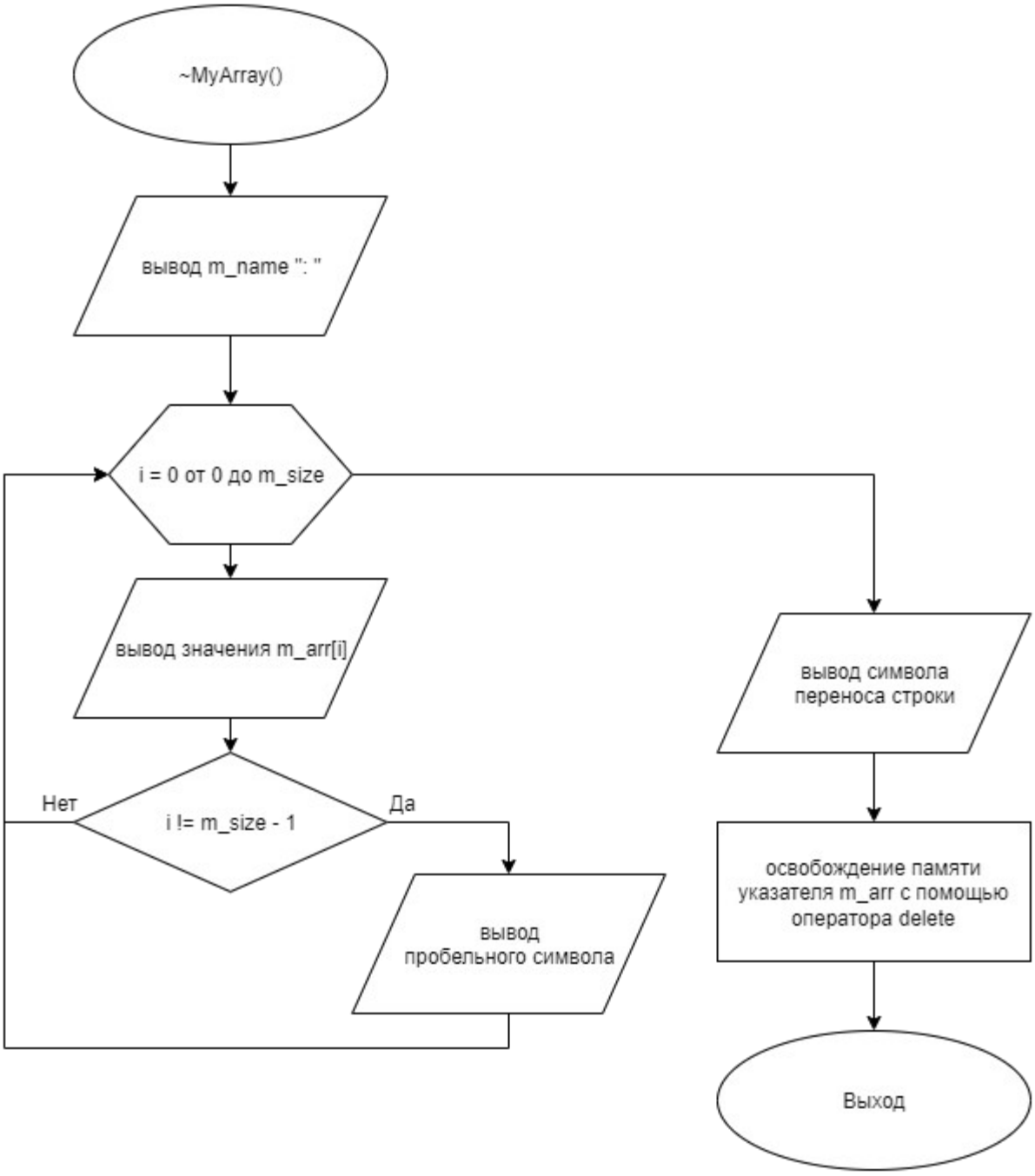

Рисунок 5 – Блок-схема алгоритма

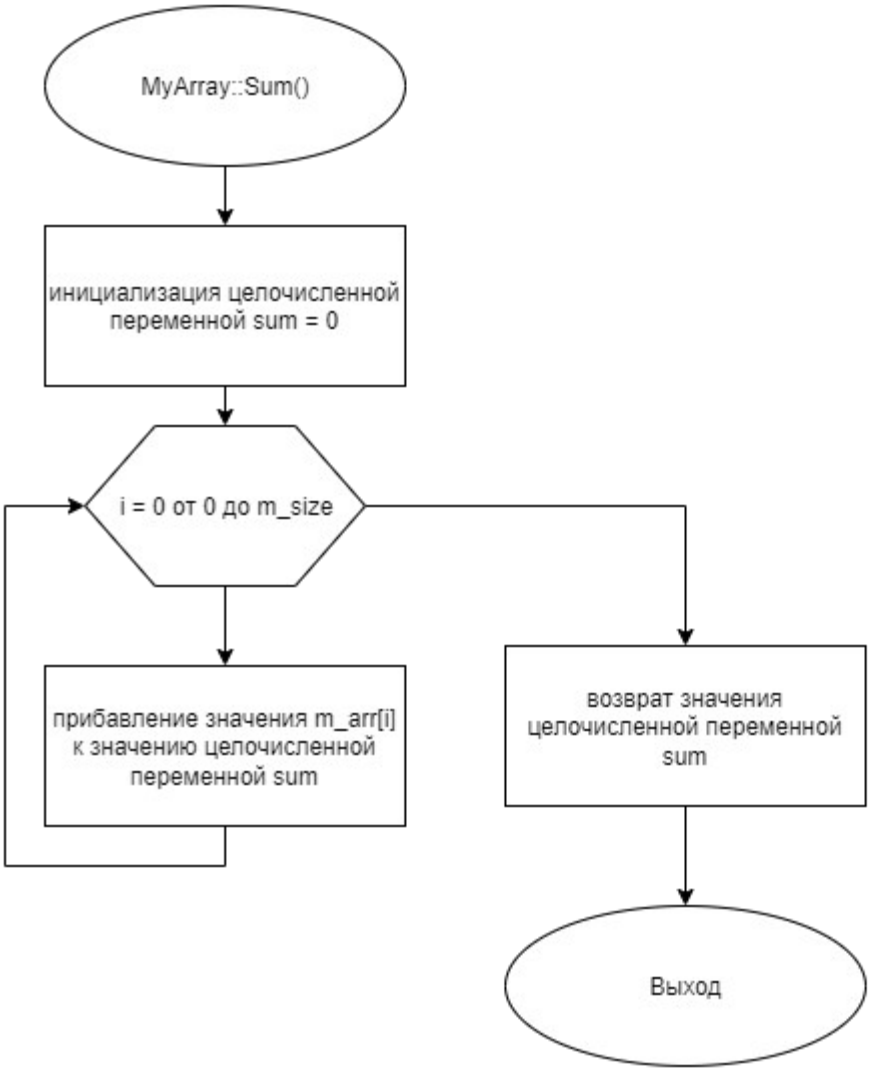

Рисунок 6 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

```
#include "MyArray.h"
#include <iostream>
#include <vector>

void CopyArray(MyArray* oldArray, MyArray* newArray)
{
   for (int i = 0; i < oldArray->m_size; i++)
   {
      newArray->m_arr[i] = oldArray->m_arr[i];
   }
}

int main()
{
   int objCount, size;

   std::cin >> objCount >> size;

   MyArray** objects = new MyArray*[objCount];

   for (int i = 0; i < objCount; i++)
   {
      objects[i] = new MyArray(size);
   }

   MyArray* minSumObj = objects[0];

   for (int i = 0; i < objCount; i++)
   {
      if (objects[i]->Sum() < minSumObj->Sum())
      {
         minSumObj = objects[i];
      }
   }

   for (int i = 0; i < objCount; i++)
   {
      CopyArray(minSumObj, objects[i]);
   }
```

```
   for (int i = 0; i < objCount; i++)
   {
      delete objects[i];
   }

   return 0;
}
```

5.2 Файл MyArray.cpp

Листинг 2 – MyArray.cpp

```
#include "MyArray.h"

MyArray::MyArray(int size) : m_size(size)
{
   m_arr = new int[size];

   std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');

   std::cin >> m_name;
   std::cout << m_name;

   for (int i = 0; i < size; i++)
   {
      std::cin >> m_arr[i];
      std::cout << "  ";
      std::cout << m_arr[i];
   }
   std::cout << std::endl;
}

int MyArray::Sum()
{
   int sum = 0;
   for (int i = 0; i < m_size; i++)
   {
      sum += m_arr[i];
   }

   return sum;
}

MyArray::~MyArray()
{
   std::cout << m_name << ": ";
   for (int i = 0; i < m_size; i++)
   {
      std::cout << m_arr[i];
```

```
            if (i != m_size - 1) std::cout << " ";
        }
        std::cout << std::endl;

        delete m_arr;
    }
```

5.3 Файл MyArray.h

Листинг 3 – MyArray.h

```
    #ifndef __MYARRAY__H
    #define __MYARRAY__H

    #include <iostream>
    #include <string>
    #include <limits>

    class MyArray
    {
        int m_size;
        int* m_arr;
        std::string m_name;
    public:
        MyArray(int size);
        int Sum();

        ~MyArray();

        friend void CopyArray(MyArray* oldArray, MyArray* newArray);
    };

    #endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 6.

Таблица 6 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
5 5 obj_2 2 2 2 2 2 2 2 obj_3 3 3 3 3 3 3 3 3 3 obj_1 1 1 1 1 1 obj_4 4 4 4 4 4 4 4 4 obj_5 5 5 5 5 5	obj_2 2 2 2 2 2 obj_3 3 3 3 3 3 obj_1 1 1 1 1 1 obj_4 4 4 4 4 4 obj_5 5 5 5 5 5 obj_2: 1 1 1 1 1 obj_3: 1 1 1 1 1 obj_1: 1 1 1 1 1 obj_4: 1 1 1 1 1 obj_5: 1 1 1 1 1	obj_2 2 2 2 2 2 obj_3 3 3 3 3 3 obj_1 1 1 1 1 1 obj_4 4 4 4 4 4 obj_5 5 5 5 5 5 obj_2: 1 1 1 1 1 obj_3: 1 1 1 1 1 obj_1: 1 1 1 1 1 obj_4: 1 1 1 1 1 obj_5: 1 1 1 1 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).