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, вначале работы выдает сообщение;
- параметризированный конструктор, передается целочисленный параметр. Параметр должен иметь значение больше 2 и быть четным. Вначале работы выдает сообщение;
- конструктор копии, обеспечивает создание копии объекта в новой области памяти. Вначале работы выдает сообщение;
- метод деструктор, который в начале работы выдает сообщение;
- метод который создает целочисленный массив в закрытой области, согласно ранее заданной размерности.
- метод ввода значений элементов созданного массива;
- метод 1, который суммирует значения очередной пары элементов и сумму присваивает первому элементу пары. 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. 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;
- метод последовательного вывода содержимого элементов массива,

которые разделены двумя пробелами;

- метод, который возвращает значение указателя на массив из закрытой области;
- метод, который присваивает значение указателя массива из закрытой области.

Назовём класс описания данного объекта cl_obj (для примера, у вас он может называться иначе).

Разработать функцию func, которая имеет один целочисленный параметр, содержащий размерность массива. В функции должен быть реализован алгоритм:

1. Инициализация указателя на объект класса cl_obj адресом объекта, созданного с использованием параметризированного конструктора.
2. С использованием указателя на объект класса cl_obj вызов метода создания массива.
3. С использованием указателя на объект класса cl_obj вызов метода ввода значений элементов массива.
4. С использованием указателя на объект класса cl_obj вызов метода 2.
5. Возврат указателя на объект класса cl_obj.

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. Объявить первый указатель на объект класса cl_obj.
5. Присвоение первому указателю результата работы функции func с аргументом, содержащим значение размерности массива.
6. С использованием первого указателя вызов метода 1.
7. Инициализация второго указателя на объект класса cl_obj адресом

объекта, созданного с использованием конструктора копии с аргументом первого объекта.

8. С использованием второго указателя вызов метода 2.
9. Вывод содержимого массива первого объекта.
10. Вывод суммы элементов массива первого объекта.
11. Вывод содержимого массива второго объекта.
12. Вывод суммы элементов массива второго объекта.
13. Второму объекту присвоить первый объект.
14. С использованием первого указателя вызов метода 1.
15. Вывод содержимого массива второго объекта.
16. Вывод суммы элементов массива второго объекта.
17. Удалит первый объект.
18. Удалить второй объект.

Добавить в этот алгоритм пункты, которые обеспечат корректное завершение работы программы.

1.1 Описание входных данных

Первая строка:

«Целое число»

Вторая строка:

«Целое число» «Целое число» . . .

Пример:

```
4
3 5 1 2
```

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 копии в начале работы с новой строки выдает сообщение:

```
Copy constructor
```

Деструктор в начале работы с новой строки выдает сообщение:

```
Destructor
```

Метод последовательного вывода содержимого элементов массива, с новой строки выдает:

```
«Целое число»  «Целое число»  «Целое число»  . . .
```

Пример вывода:

```
4
Constructor set
Copy constructor
20  5  4  2
31
100  5  8  2
```

```
115
100  5  8  2
115
Destructor
Destructor
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- стандартная библиотека потока ввода и вывода.

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 метода size класса Class

Функционал: конструктор.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 1.

Таблица 1 – Алгоритм метода size класса Class

№	Предикат	Действия	№ перехода
1		присвоение размера массива	2
2		вывод сообщеия	∅

3.2 Алгоритм деструктора класса Class

Функционал: деструктор.

Параметры: нет.

Алгоритм деструктора представлен в таблице 2.

Таблица 2 – Алгоритм деструктора класса Class

№	Предикат	Действия	№ перехода
1		удаление массива	2
2		вывод сообщения	∅

3.3 Алгоритм метода createarray класса Class

Функционал: создание маасива.

Параметры: нет.

Возвращаемое значение: none.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода createarray класса Class

№	Предикат	Действия	№ перехода
1		изменение размера массива	∅

3.4 Алгоритм метода puch_back класса Class

Функционал: ввод значений элементов.

Параметры: нет.

Возвращаемое значение: none.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода puch_back класса Class

№	Предикат	Действия	№ перехода
1	закончился массив		∅
		ввод значений и присвоение эл. массива	1

3.5 Алгоритм метода method1 класса Class

Функционал: первый метод.

Параметры: нет.

Возвращаемое значение: none.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода method1 класса Class

№	Предикат	Действия	№ перехода
1		суммирование каждой пары	∅

3.6 Алгоритм метода method2 класса Class

Функционал: второй етод.

Параметры: нет.

Возвращаемое значение: none.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода method2 класса Class

№	Предикат	Действия	№ перехода
1		произведение каждой пары	∅

3.7 Алгоритм метода print класса class

Функционал: Вывод элементов.

Параметры: нет.

Возвращаемое значение: none.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода print класса class

№	Предикат	Действия	№ перехода
1		вывод всех элементов массива	∅

3.8 Алгоритм метода getarray класса Class

Функционал: получение массива.

Параметры: нет.

Возвращаемое значение: int*.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода getarray класса Class

№	Предикат	Действия	№ перехода
1		возврат массива	∅

3.9 Алгоритм функции main

Функционал: основная функция.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм функции представлен в таблице 9.

Таблица 9 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		вввод размерности массива	2
2	размерность массива	вывод сооб. и завершение алгоритма	∅
		вывод значения размерности массива	3
3		объявить первый указатель	4
4		присвоение первому указ. рез. func с аргументом	5
5		вызов метода 1	6
6		инициализация второго указателя	7
7		вызов метода 2	8
8		вывод первого объекта	9
9		вывод суммы первого объекта	10
10		вывод второго объекта	11
11		вывод суммы второго объекта	12
12		второму объекту присвоить первый	13
13		вызов метода 1	14

№	Предикат	Действия	№ перехода
14		вывод второго объекта	15
15		вывод суммы эл. второго объекта	16
16		удалть первый и второй объекты	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-5.

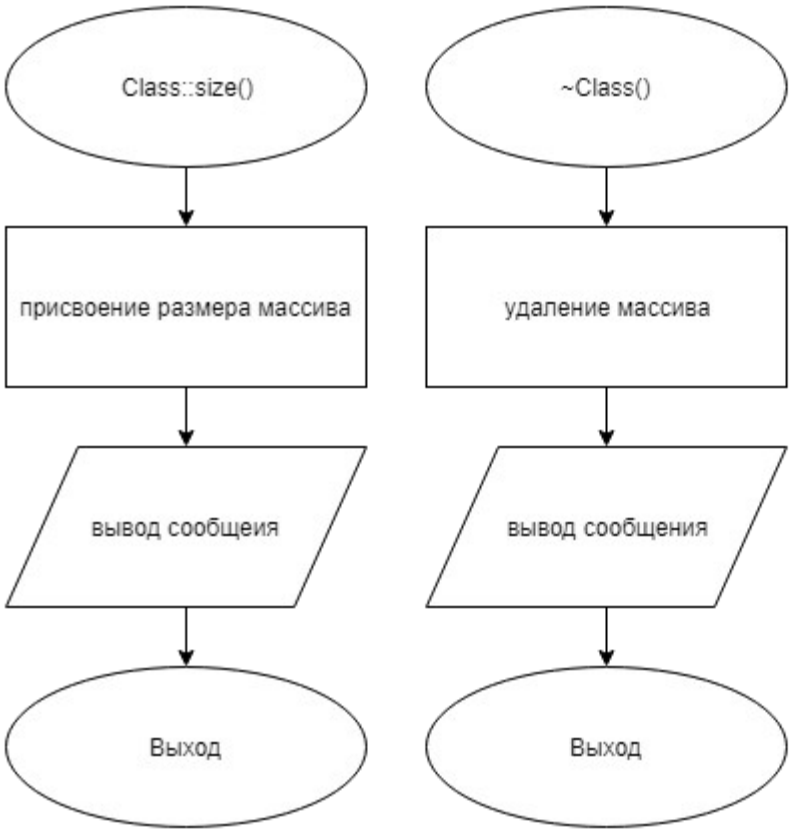

Рисунок 1 – Блок-схема алгоритма

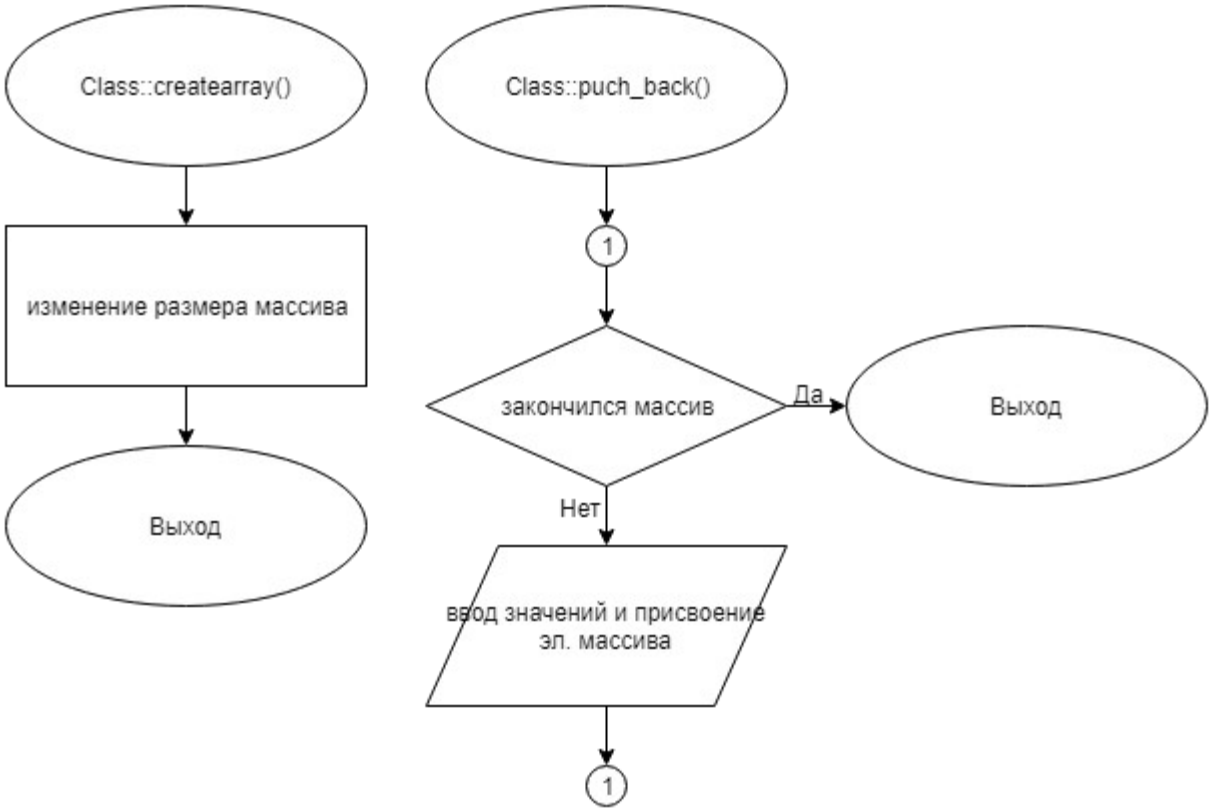

Рисунок 2 – Блок-схема алгоритма

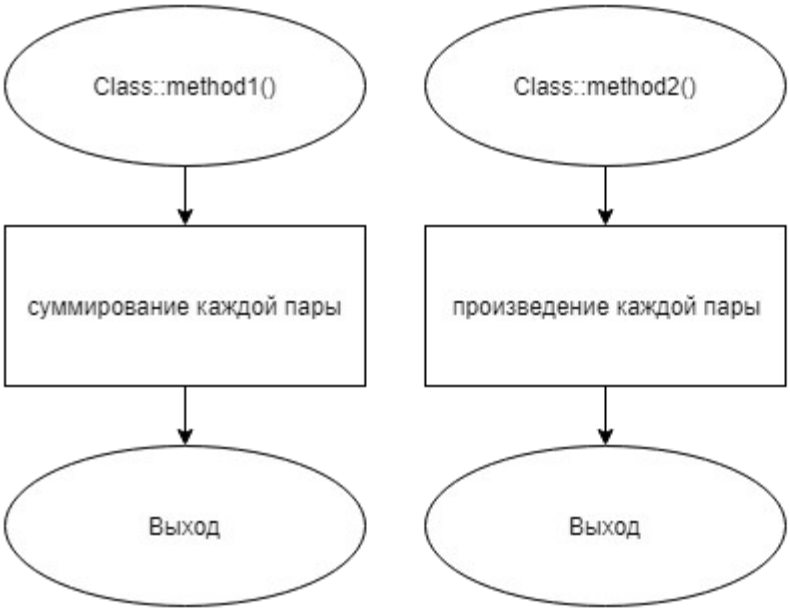

Рисунок 3 – Блок-схема алгоритма

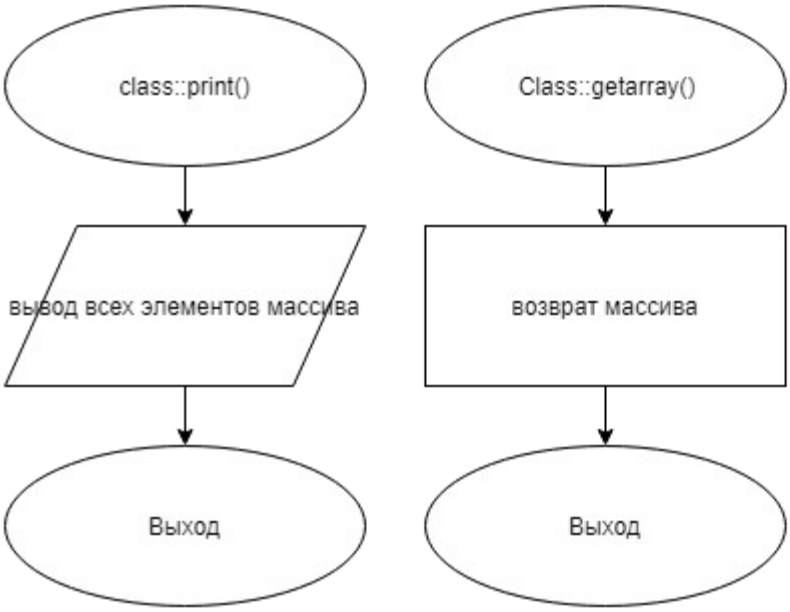

Рисунок 4 – Блок-схема алгоритма

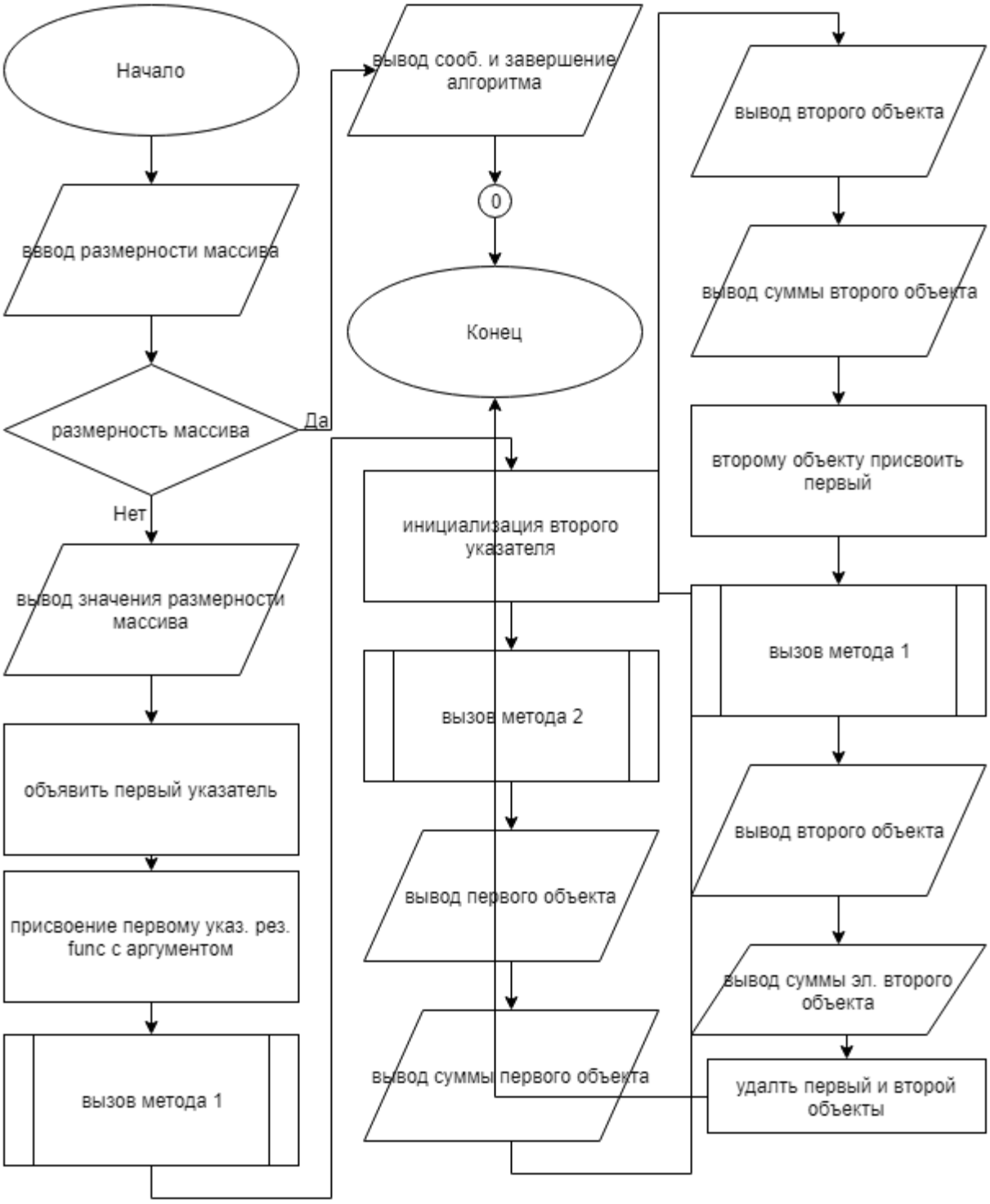

Рисунок 5 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл Class.cpp

Листинг 1 – Class.cpp

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

Class::Class(){cout << "Default constructor\n";}
Class::Class(int n)
{
   size = n;
   cout << "Constructor set";
}

Class::Class(const Class& obj)
{
   size = obj.size;
   array = new int[size];
   for (int i = 0; i < size; i++)
   {
      array[i] = obj.array[i];
   }
   cout << "\nCopy constructor\n";
}

Class::~Class()
{
   delete [] array;
   cout << "\nDestructor";
}

void Class::createarray()
{
   array = new int[size];
}

void Class::push_back()
{
   for (int i = 0; i < size; i++)
   {
      int a= 0;
      while(!(cin >> a))
```

```
            {
               cin.clear();
               cin.ignore(size, '\n');
            }
            array[i] = a;
         }
      }

      void Class::method1()
      {
         for (int i = 0; i < size - 1; i +=2)
         {
            array[i] += array[i + 1];
         }
      }
      void Class::method2()
      {
         for (int i = 0; i < size - 1; i +=2)
         {
            array[i] *= array[i + 1];
         }
      }

      int Class::sum()
      {
         int result = 0;
         for (int i = 0; i < size; i++)
         {
            result += array[i];
         }
         return result;
      }

      void Class::print()
      {
         for(int i = 0; i < size - 1; i++)
         {
            cout << array[i] << "  ";
         }
         cout << array[size - 1];
         cout << endl;
      }

      void Class::setarray(int* array)
      {
         this->array = new int[size];
         for (int i = 0; i < size; i++)
         {
            this->array[i] = array[i];
         }
      }

      int* Class::getarray()
      {
         return array;
```

```
}
```

5.2 Файл Class.h

Листинг 2 – Class.h

```
#ifndef __CLASS__H
#define __CLASS__H
#include <iostream>
#include <iomanip>
using namespace std;

class Class
{
private:
   int* array = nullptr;
   int size;
public:
   Class();
   Class(int n);
   Class(const Class& obj);
   ~Class();

   void createarray();
   void push_back();
   void method1();
   void method2();
   int sum();
   void print();
   int* getarray();
   void setarray(int* array);
};
#endif
```

5.3 Файл main.cpp

Листинг 3 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include "Class.h"
#include <iostream>
using namespace std;

Class* func(int size)
```

```
{
   Class* object = new Class(size);
   object->createarray();
   object->push_back();
   object->method2();
   return object;
}
int main()
{
   int s1;
   cin >> s1;
   if ((s1 > 2) && (s1 % 2 == 0))
   {
      cout << s1 << endl;

      Class* obj1 = func(s1);
      obj1->method1();
      Class* obj2 = new Class(*obj1);
      obj2->method2();
      obj1->print();
      cout << obj1->sum() << endl;
      obj2->print();
      cout << obj2->sum() <<endl;

      obj2->print();
      cout << obj2->sum();

      delete obj1;
      delete obj2;
   }
   else
   {
      cout << s1 << "?";
   }
   return(0);
}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 10.

Таблица 10 – Результат тестирования программы

Входные данные	**Ожидаемые выходные данные**	**Фактические выходные данные**
4 3 5 1 2	4 Constructor set Copy constructor 20 5 4 2 31 100 5 8 2 115 100 5 8 2 115 Destructor Destructor	4 Constructor set Copy constructor 20 5 4 2 31 100 5 8 2 115 100 5 8 2 115 Destructor 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).