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

Класс obj:

- функционал:
    - метод print — вывод массива объекта на экран;
    - метод get_point — возврат значения указателя на массив из закрытой области;
    - метод set_point — присваивание значение указателя массива из закрытой области.

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 print класса obj

Функционал: вывод массива объекта на экран.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 1.

Таблица 1 – Алгоритм метода print класса obj

№	Предикат	Действия	№ перехода
1		вывод на экран перехода на новую строку и значение элемента массива с индексом 0	2
2		инициализация переменной i типа int значением 1	3
3	i<n	вывод на экран " " (2 пробела) и значение элемента массива с индексом i	4
			∅
4		i++	3

3.2 Алгоритм метода get_point класса obj

Функционал: возврат значения указателя на массив из закрытой области.

Параметры: нет.

Возвращаемое значение: int*.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода get_point класса obj

№	Предикат	Действия	№ перехода
1		возврат matrix	∅

3.3 Алгоритм метода set_point класса obj

Функционал: присваивание значение указателя массива из закрытой области.

Параметры: int* matrix.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода set_point класса obj

№	Предикат	Действия	№ перехода
1		присваивание полю объекта matrix значение параметра matrix	∅

3.4 Алгоритм функции main

Функционал: основная функция программы.

Параметры: нет.

Возвращаемое значение: int - код ошибки.

Алгоритм функции представлен в таблице 4.

Таблица 4 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		объявление переменной n типа int и ввод значения с клавиатуры	2
2	!(n>2 && n%2==0)	вывод на экран значение переменной n и ?	20
		вывод на экран значение переменной n	3

№	Предикат	Действия	№ перехода
3		объявление первого указателя point1 на объект класса obj	4
4		присвоение первому указателю point1 результата работы функции func с аргументом n	5
5		с использованием первого указателя point1 вызов метода sum()	6
6		инициализация второго указателя point2 на объект класса obj адресом объекта, созданного с использованием конструктора копии с аргументом первого объекта	7
7		с использованием второго указателя point2 вызов метода mult()	8
8		вывод содержимого массива первого объекта с использованием метода print() на первом указателе point1	9
9		вывод на экран суммы элементов массива первого объекта с использованием метода output() на первом указателе point1	10
10		вывод содержимого массива второго объекта с использованием метода print() на втором указателе point2	11
11		вывод суммы элементов массива первого объекта с использованием метода output() на втором указателе point2	12
12		инициализация переменной temp типа int* результатом метода get_point(), вызванном на указателе на второй объект point2	13
13		присвоение второму объекту первого	14
14		с использованеим первого указателя point1 вызов	15

№	Предикат	Действия	№ перехода
		метода sum()	
15		вызов метода set_point() с параметром temp для указателя на второй объект point2	16
16		вывод содержимого второго объекта с использованием метода print() на втором указателе point2	17
17		вывод суммы элементов массива второго объекта с использованием метода output() на втором указателе point2	18
18		удаление указателя на первый объект point1	19
19		удаление указателя на второй объект point2	20
20		возврат 0	∅

3.5 Алгоритм функции func

Функционал: возвращает указатель на объект класса obj.

Параметры: int n.

Возвращаемое значение: obj*.

Алгоритм функции представлен в таблице 5.

Таблица 5 – Алгоритм функции func

№	Предикат	Действия	№ перехода
1		инициализация указателя на объект класса obj адресом объекта, созданного с использованием параметризированного конструктора	2
2		с использованием указателя на объект класса obj вызов метода create()	3
3		с использованием указателя на объект класса obj вызов метода input()	4
4		с использованием указателя на объект класса obj вызов метода mult()	5
5		возврат указателя на объект класса obj	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-3.

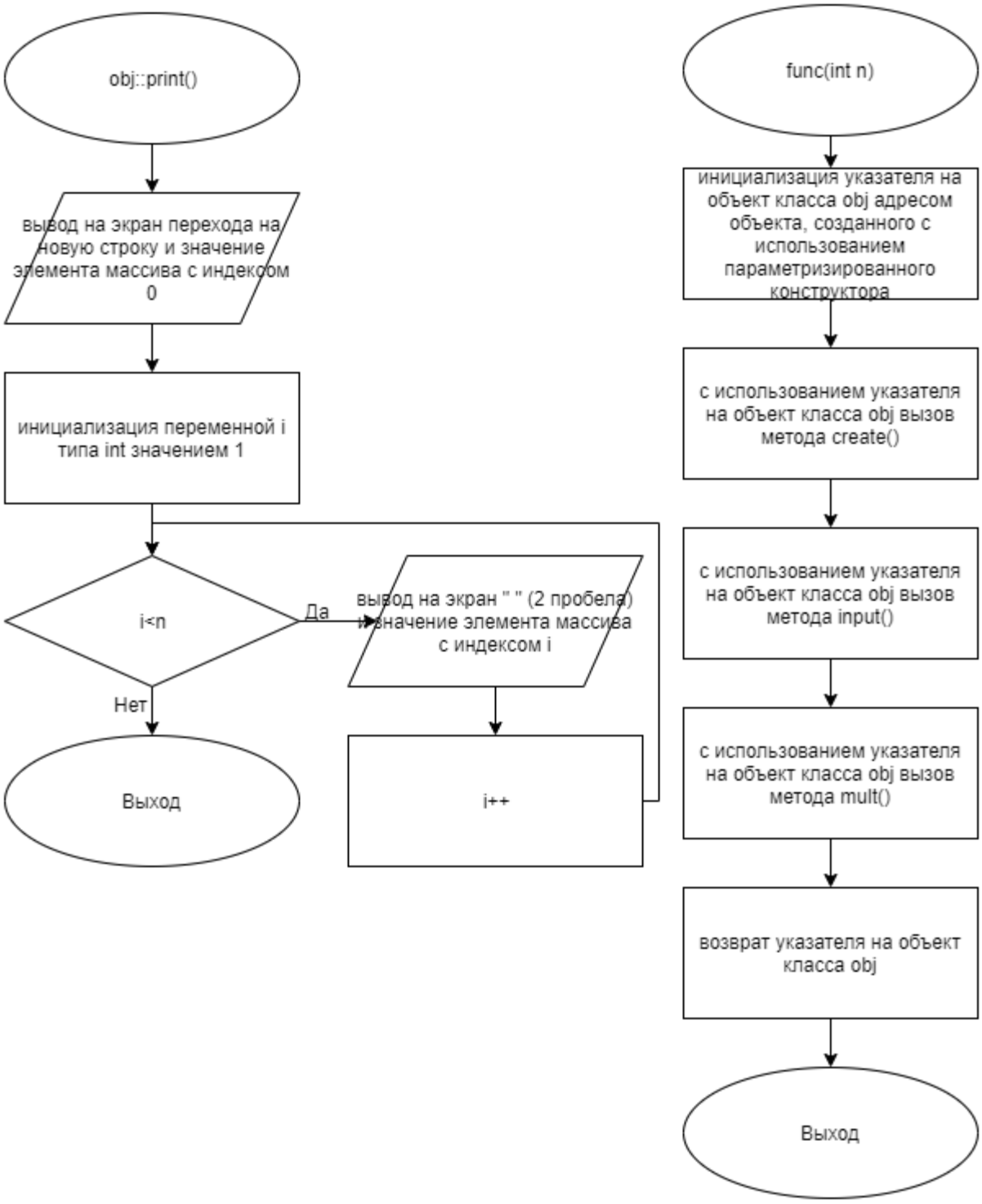

Рисунок 1 – Блок-схема алгоритма

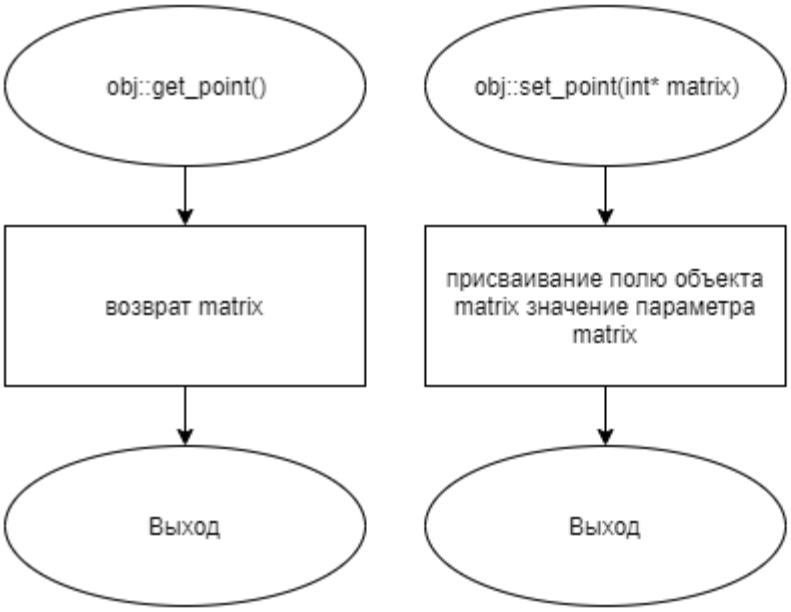

Рисунок 2 – Блок-схема алгоритма

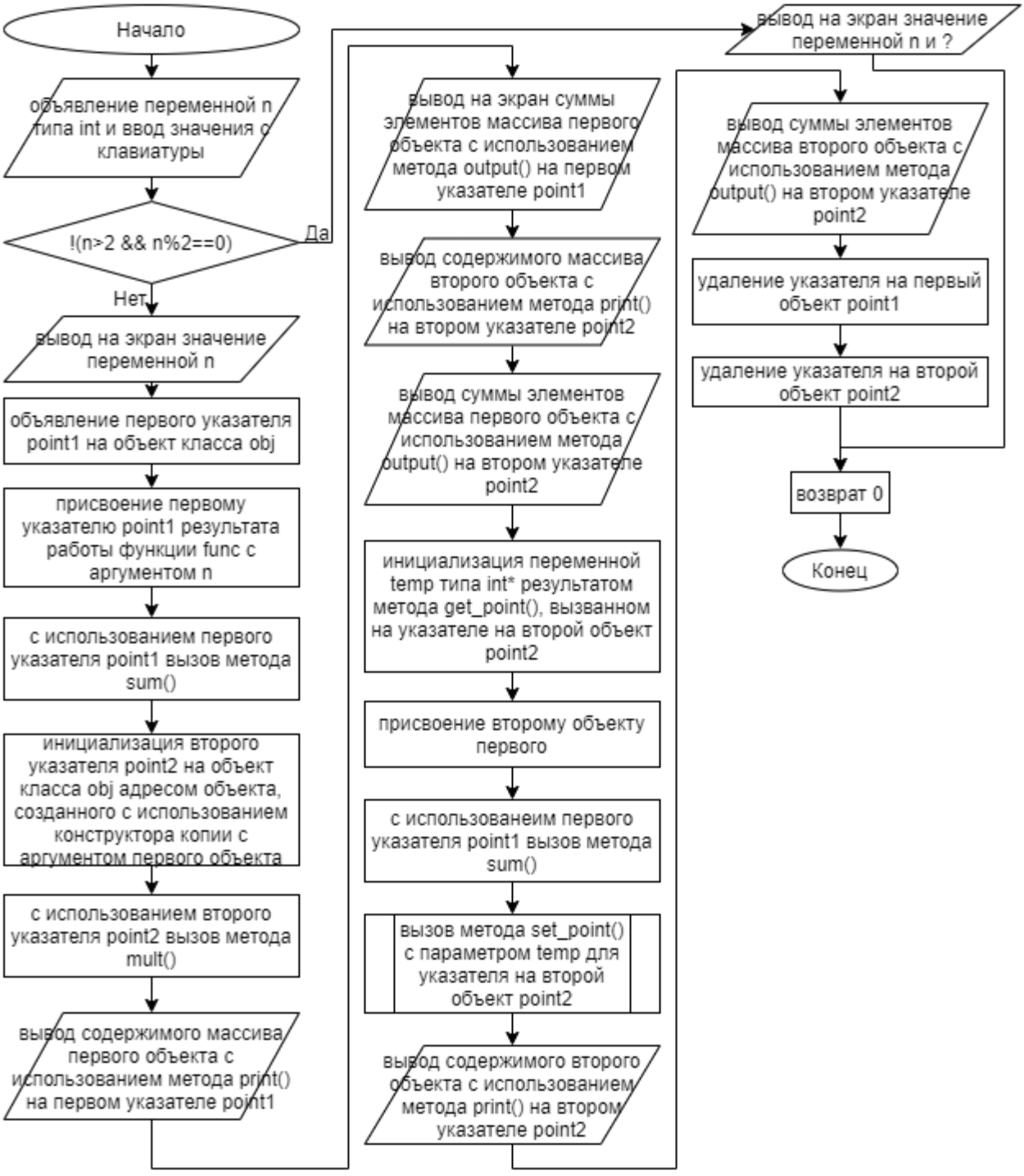

Рисунок 3 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include "obj.h"

obj* func(int n)
{
   obj* point = new obj(n);
   point -> create();
   point -> input();
   point -> mult();
   return point;
}

int main()
{
   int n; std::cin >> n;
   if (!(n > 2 && n % 2 == 0))
      std::cout << n << '?';
   else
   {
      std::cout << n;
      obj* point1;
      point1 = func(n);
      point1 -> sum();
      obj* point2 = new obj(*point1);
      point2 -> mult();
      point1 -> print();
      std::cout << point1 -> output();
      point2 -> print();
      std::cout << point2 -> output();
      int* temp = point2 -> get_point();
      *point2 = *point1;
      point2 -> set_point(temp);
      point1 -> sum();
      point2 -> print();
      std::cout << point2 -> output();
      delete point1;
      delete point2;
```

```
   }
   return(0);
}
```

5.2 Файл obj.cpp

Листинг 2 – obj.cpp

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

obj::obj() {
   std::cout << "\nDefault constructor";
}
obj::obj(int n) {
   std::cout << "\nConstructor set";
   this -> n = n;
   matrix = new int[n];
}
obj::obj(const obj & ob) {
   std::cout << "\nCopy constructor";
   n = ob.n;
   matrix = new int [n];
   for (int i = 0; i < n; i++)
      matrix[i] = ob.matrix[i];
}
obj::~obj() {
   std::cout << "\nDestructor";
   delete[] matrix;
}

void obj::create() {
   matrix = new int [n];
}
void obj::input() {
   for (int i = 0; i < n; i++)
      std::cin >> matrix[i];
}
void obj::sum() {
   for (int i = 0; i < n-1; i+=2)
      matrix[i] += matrix[i+1];
}
void obj::mult() {
   for (int i = 0; i < n-1; i+=2)
      matrix[i] *= matrix[i+1];
}
int obj::output() {
   int sum = 0;
   for (int i = 0; i < n; i++)
      sum += matrix[i];
```

```
   std::cout << std::endl;
   return sum;
}
void obj::print() { //changed
   std::cout << std::endl << matrix[0];
   for (int i = 1; i < n; i++)
      std::cout << "  " << matrix[i];
}

int* obj::get_point() { //new
   return matrix;
}
void obj::set_point(int* matrix) { //new
   this->matrix = matrix;
}
```

5.3 Файл obj.h

Листинг 3 – obj.h

```
#ifndef __OBJ__H
#define __OBJ__H

class obj
{
private:
   int n;
   int *matrix = nullptr;
public:
   obj();
   obj(int n);
   obj(const obj & ob);
   ~obj();
   void create();
   void input();
   void sum();
   void mult();
   int output();
   void print(); //changed
   int* get_point(); //new
   void set_point(int* matrix); //new
};

#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 6.

Таблица 6 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
4 5 6 7 8	4 Constructor set Copy constructor 36 6 64 8 114 216 6 512 8 742 216 6 512 8 742 Destructor Destructor	4 Constructor set Copy constructor 36 6 64 8 114 216 6 512 8 742 216 6 512 8 742 Destructor Destructor
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).