Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ.....5
1.1 Описание входных данных.....7
1.2 Описание выходных данных.....8
2 МЕТОД РЕШЕНИЯ.....10
3 ОПИСАНИЕ АЛГОРИТМОВ.....11
3.1 Алгоритм функции main.....11
3.2 Алгоритм функции func.....13
3.3 Алгоритм метода get_array класса Object.....14
3.4 Алгоритм метода set_array класса Object.....14
4 БЛОК-СХЕМЫ АЛГОРИТМОВ.....15
5 КОД ПРОГРАММЫ.....22
5.1 Файл main.cpp.....22
5.2 Файл Object.cpp.....24
5.3 Файл Object.h.....26
6 ТЕСТИРОВАНИЕ.....28
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....29

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 используетя:

- объект стандартного потока ввода std::cin (используется для ввода с клавиатуры);
- объект стандартного потока вывода std::cout (используется для вывода на экран);
- условный оператор if..else;
- объекты класса Object.

Класс Object:

- Свойства (поля) - дополнительных полей добавлено не было;
- Методы:
    - Метод get_array:
        - Функционал - возврат указателя на массив, содержащийся в закрытой области объекта класса Object;
        - Возвращаемое значение - указатель на целочисленный массив;
        - Модификатор доступа - открытый;
        - Параметры - отсутствуют;
    - Метод set_array:
        - Функционал - устновка значения массива, содержащегося в закрытой области объекта класса Object;
        - Возвращаемое значение - отсутствует (void);
        - Модификатор доступа - открытый;
        - Параметры:
            - указатель на целочисленный массив new_array.

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 программы.

Параметры: остутствуют.

Возвращаемое значение: целочисленный код завершения работы программы.

Алгоритм функции представлен в таблице 1.

Таблица 1 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Инициализация целочисленной переменной size значением 0	2
2		Ввод значения переменной size	3
3		Вывод значения переменной size	4
4	Значение переменной size меньше 2 или остаток от деления значения переменной size на 2 равен 1	Вывод строки "?"	∅
			5
5		Объявление указателя на объект класса Object obj1	6
6		Присваивание указателю obj1 значения указателя, возвращённого функцией func, вызванной с значением переменной size в качестве параметра	7
7		Вызов метода method1 объекта, на который	8

№	Предикат	Действия	№ перехода
		указывает указатель obj1	
8		Инициализация указателя на объект класса Object obj2 адресом нового объекта класса Object, созданного с помощью оператора new с использованием параметризованного конструктора и объектом, на который указывает указатель obj1, в качестве параметра	9
9		Вызов метода method2 объекта, на который указывает указатель obj2	10
10		Вывод строки "\n" (переход на новую строку)	11
11		Вызов метода print объекта, на который указывает указатель obj1	12
12		Вывод строки "\n", значения, возвращённого методом sum объекта, на который указывает указатель obj1, и строки "\n"	13
13		Вызов метода print объекта, на который указывает указатель obj2	14
14		Вывод строки "\n", значения, возвращённого методом sum объекта, на который указывает указатель obj2, и строки "\n"	15
15		Инициализация указателя на объект класса Object obj2 значением указателя obj2	16
16		Присваивание указателю obj2 значения указателя obj1	17
17		Вызов метода method1 объекта, на который указывает указатель obj1	18
18		Присваивание указателю obj2 значения указателя obj3	19
19		Вызов метода print объекта, на который указывает	20

№	Предикат	Действия	№ перехода
		указатель obj2	
20		Вывод строки "\n" и значения, возвращённого методом sum объекта, на который указывает указатель obj2	21
21		Удаление объекта, на который указывает указатель obj1 с помощью оператора delete	22
22		Удаление объекта, на который указывает указатель obj2 с помощью оператора delete	∅

3.2 Алгоритм функции func

Функционал: создание объекта класса Object по заданному размеру массива и возврат указателя на созданный объект.

Параметры: целочисленный параметр size.

Возвращаемое значение: указатель на объект класса Object.

Алгоритм функции представлен в таблице 2.

Таблица 2 – Алгоритм функции func

№	Предикат	Действия	№ перехода
1		Инициализация указателя на объект класса obj адресом нового объекта, созданного с помощью оператора new с использованием параметризованного конструктора со значением параметра size в качестве аргумента	2
2		Вызов метода make_array объекта, на который указывает указатель obj	3
3		Вызов метода read_array объекта, на который указывает указатель obj	4
4		Вызов метода method2 объекта, на который указывает указатель obj	5
5		Возврат указателя obj	∅

3.3 Алгоритм метода get_array класса Object

Функционал: возврат указателя на массив, содержащийся в закрытой области объекта класса Object.

Параметры: отсутствуют.

Возвращаемое значение: указатель на целочисленный массив.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода get_array класса Object

№	Предикат	Действия	№ перехода
1		Возврат значение поля array	∅

3.4 Алгоритм метода set_array класса Object

Функционал: устновка значения массива, содержащегося в закрытой области объекта класса Object.

Параметры: указатель на целочисленный массив new_array.

Возвращаемое значение: отсутствует (void).

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода set_array класса Object

№	Предикат	Действия	№ перехода
1		Присваивание полю array значения параметра new_array	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-7.

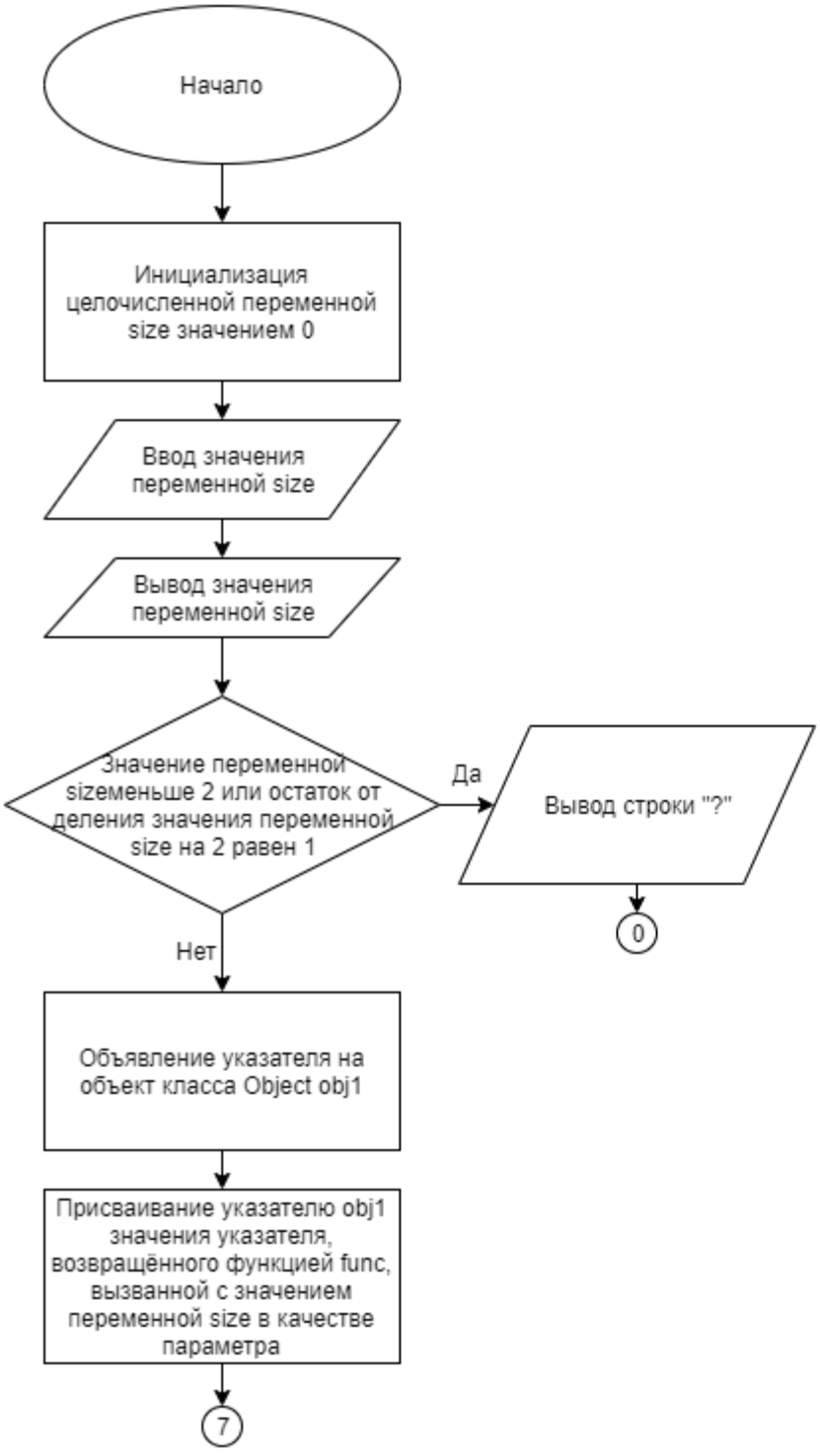

Рисунок 1 – Блок-схема алгоритма

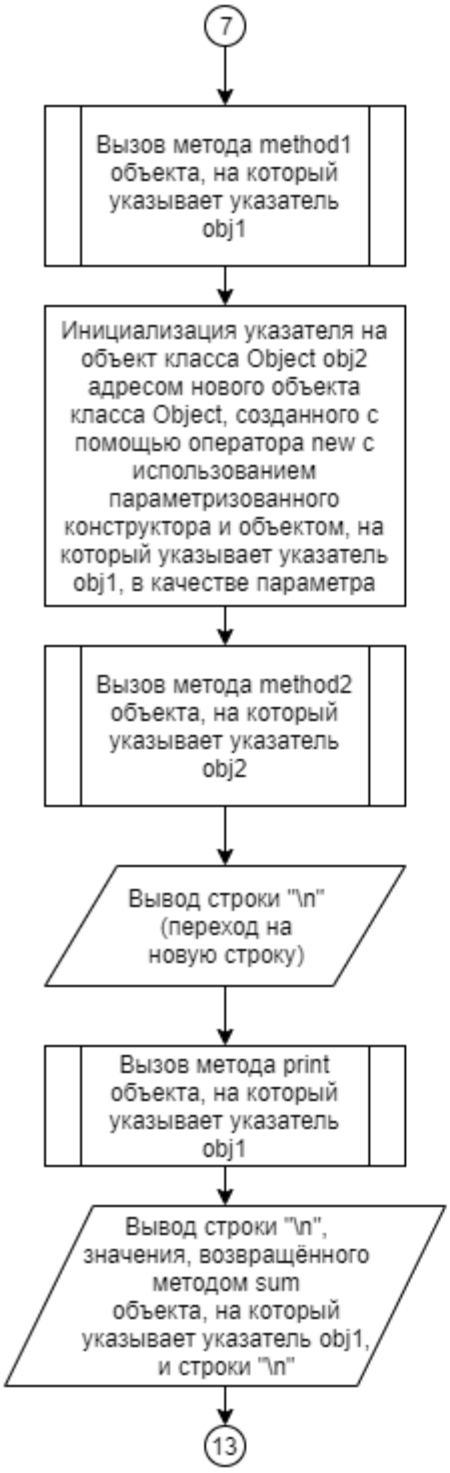

Рисунок 2 – Блок-схема алгоритма

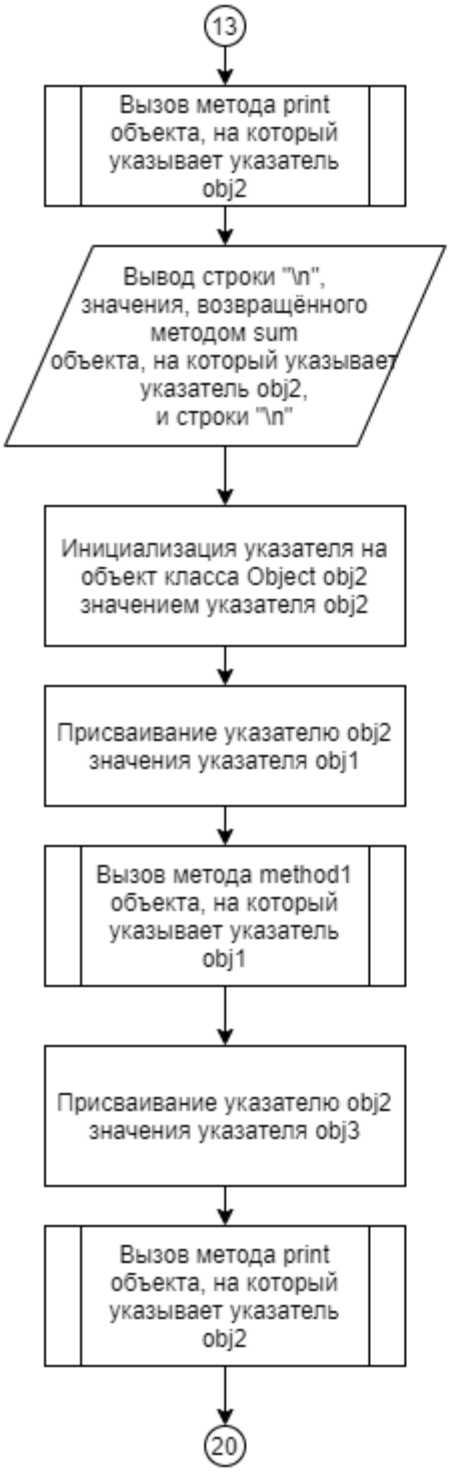

Рисунок 3 – Блок-схема алгоритма

Рисунок 4 – Блок-схема алгоритма

Рисунок 5 – Блок-схема алгоритма

Рисунок 6 – Блок-схема алгоритма

Рисунок 7 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

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

Object* func(int size)
{
   // Инициализация указателя obj
   // новым объектом класса Object,
   // созданным с использованием
   // параметризованного
   // конструктора
   Object* obj = new Object(size);
   // Вызов метода make_array объекта
   // на который указывает указатель obj
   obj->make_array();
   // Вызов метода make_array объекта
   // на который указывает указатель obj
   obj->read_array();
   // Вызов метода method2 объекта
   // на который указывает указатель obj
   obj->method2();
   // Возвращение
   // указателя obj
   return obj;
}

int main()
{
   // Инициализация целочисленной переменной
   // size значением 0
   int size = 0;
   // Ввод size
   std::cin >> size;
   // Вывод size
   std::cout << size;

   // Если значение size меньше
   // двух или нечётно
   if (size < 2 || size % 2 == 1)
   {
```

```
            // Вывод "?"
            std::cout << '?';
        }
        // иначе
        else
        {
            // Объявление указателя на объект класса Object obj1
            Object* obj1;
            // Присваивание указателю obj1 значения указателя,
            // возвращённого функцией func, вызванной с значением
            // переменной size в качестве параметра
            obj1 = func(size);

            // Вызов метода method1 объекта,
            // на который указывает указатель obj1
            obj1->method1();

            // Инициализация указателя на объект класса Object obj2
            // адресом нового объекта класса Object, созданного
            // с помощью оператора new с использованием
            // параметризованного конструктора
            Object* obj2 = new Object(*obj1);

            // Вызов метода method2 объекта,
            // на который указывает указатель obj2
            obj2->method2();

            // Вывод строки "\n" (переход на новую строку)
            std::cout << '\n';

            // Вызов метода print объекта, на который указывает указатель obj1
            obj1->print();

            // Вывод строки "\n", значения, возвращённого методом sum объекта,
            // на который указывает указатель obj1, и строки "\n"
            std::cout << '\n' << obj1->sum() << '\n';

            // Вызов метода print объекта, на который указывает указатель obj2
            obj2->print();

            // Вывод строки "\n", значения, возвращённого методом sum объекта,
            // на который указывает указатель obj2, и строки "\n"
            std::cout << '\n' << obj2->sum() << '\n';

            // Инициализация указателя на объект класса
            // Object obj2 значением указателя obj2
            Object* obj3 = obj2;

            // Присваивание указателю obj2 значения указателя obj1
            obj2 = obj1;

            // Вызов метода method1 объекта,
            // на который указывает указатель obj1
            obj1->method1();
```

```
        // Присваивание указателю obj2 значения указателя obj3
        obj2 = obj3;

        // Вызов метода print объекта, на который указывает указатель obj2
        obj2->print();

        // Вывод строки "\n" и значения, возвращённого методом
        // sum объекта, на который указывает указатель obj2
        std::cout << '\n' << obj2->sum();

        // Удаление объекта, на который указывает
        // указатель obj1 с помощью оператора delete
        delete obj1;

        // Удаление объекта, на который указывает
        // указатель obj2 с помощью оператора delete
        delete obj2;
    }
    return 0;
}
```

5.2 Файл Object.cpp

Листинг 2 – Object.cpp

```
#include "Object.h"

Object::Object()
{
   std::cout << "\nDefault constructor";
   size = 0;
   array = nullptr;
}

Object::Object(int size)
{
   std::cout << "\nConstructor set";
   this->size = size;
   array = new int[size];
}

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

```
}

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

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

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

   return sum();
}

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

   return sum();
}

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

   return sum;
}

void Object::make_array()
{
   array = new int[size];
}

void Object::print()
{
   for (int i = 0; i < size; i++)
```

```
        {
            std::cout << array[i];
            if (i + 1 < size)
            {
                std::cout << "  ";
            }
        }
    }

    int* Object::get_array()
    {
        return array;
    }

    void Object::set_array(int* new_array)
    {
        array = new_array;
    }
```

5.3 Файл Object.h

Листинг 3 – Object.h

```
    #ifndef OBJECT_H
    #define OBJECT_H

    #include <iostream>

    class Object
    {
    private:
        int* array;
        int size;
    public:
        Object();
        Object(int size);
        Object(const Object& obj);
        ~Object();
        void read_array();
        int method1();
        int method2();
        int sum();

        void make_array();
        void print();

        int* get_array();
        void set_array(int* new_array);
    };
```

```
    #endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 5.

Таблица 5 – 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).