

Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ.....	5
1.1 Описание входных данных.....	7
1.2 Описание выходных данных.....	7
2 МЕТОД РЕШЕНИЯ.....	8
3 ОПИСАНИЕ АЛГОРИТМОВ.....	10
3.1 Алгоритм метода Method_ru класса cl_child.....	10
3.2 Алгоритм метода Vivod класса cl_child.....	10
3.3 Алгоритм конструктора класса cl_child.....	11
3.4 Алгоритм метода Method_ru класса cl_parent.....	11
3.5 Алгоритм метода Vivod класса cl_parent.....	11
3.6 Алгоритм конструктора класса cl_parent.....	12
3.7 Алгоритм метода Method класса cl_parent.....	12
3.8 Алгоритм функции main.....	12
4 БЛОК-СХЕМЫ АЛГОРИТМОВ.....	14
5 КОД ПРОГРАММЫ.....	18
5.1 Файл cl_child.cpp.....	18
5.2 Файл cl_child.h.....	18
5.3 Файл cl_parent.cpp.....	19
5.4 Файл cl_parent.h.....	19
5.5 Файл main.cpp.....	20
6 ТЕСТИРОВАНИЕ.....	22
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	23

1 ПОСТАНОВКА ЗАДАЧИ

Описать класс `cl_parent` объекта, в котором следующий состав элементов:

В закрытом разделе:

- одно свойство целого типа;
- метод, с одним целочисленным параметром, который меняет значение свойства в закрытом разделе на удвоенное значение параметра.

В открытом разделе:

- одно свойство целого типа;
- параметризованный конструктор, с двумя целочисленными параметрами, который устанавливает значения свойств в закрытом и открытом разделе. Значение закрытого свойства меняется посредством вызова метода из закрытого раздела;
- метод с двумя целочисленными параметрами, который устанавливает значения свойств в закрытом и открытом разделе. Значение закрытого свойства меняется посредством вызова метода из закрытого раздела;
- метод, который выводит на экран значение обоих свойств. Сперва значение закрытого свойства, потом значение открытого свойства.

Назовем объект данного класса родительским. Соответственно его класс родительским классом.

На базе родительского объекта сконструируем производный объект. Производный объект должен сохранить открытый доступ к открытым элементам родительского класса. Он должен иметь следующие собственные элементы:

В закрытом разделе:

- одно свойство целого типа, наименование которого совпадает с наименованием закрытого свойства родительского объекта;

В открытом разделе:

- одно свойство целого типа, наименование которого совпадает с наименованием открытого свойства родительского объекта;
- параметризованный конструктор, с двумя целочисленными параметрами, который устанавливает значения свойств в закрытом и открытом разделе;
- метод с двумя целочисленными параметрами, который устанавливает значения свойств в закрытом и открытом разделе. Наименование метода совпадает с наименованием аналогичного метода родительского объекта;
- метод, который выводит на экран значение обоих свойств. Сперва значение закрытого свойства, потом значение открытого свойства. Наименование метода совпадает с наименованием аналогичного метода родительского объекта.

Разработать производный класс используя класс `cl_parent` в качестве родительского.

В основной функции реализовать алгоритм:

1. Ввод значения двух целочисленных переменных.
2. Создать объект производного класса используя целочисленных переменных в конструкторе в качестве аргументов в последовательности, как им были присвоены значения. Первый аргумент содержит значение для свойства закрытого раздела, второй для свойства открытого раздела.
3. Вывод значений свойств родительского объекта.
4. Вывод значений свойств производного объекта.
5. Если исходное значение закрытого свойства больше нуля, то:
 - 5.1. Переопределить значения свойств производного объекта, увеличив на единицу введенные исходные значения.
 - 5.2. Переопределить значения свойств родительского объекта, уменьшив на единицу введенные исходные значения.
 - 5.3. Вывод значений свойств производного объекта.

5.4. Вывод значений свойств родительского объекта.

6. Иначе:

6.1. Переопределить значения свойств родительского объекта, увеличив на единицу введенные исходные значения.

6.2. Переопределить значения свойств производного объекта, уменьшив на единицу введенные исходные значения.

6.3. Вывод значений свойств родительского объекта.

6.4. Вывод значений свойств производного объекта.

1.1 Описание входных данных

В первой строке:

«Целое число» «Целое число»

Пример ввода:

8 5

1.2 Описание выходных данных

Начиная с первой строки:

«Целое число»	«Целое число»
«Целое число»	«Целое число»
«Целое число»	«Целое число»
«Целое число»	«Целое число»

Пример вывода:

16	5
8	5
9	6
14	4

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект `cl_child` класса `cl_child` предназначен для ;
- функция `main` для ;
- Условный оператор;
- Объект стандартного потока ввода/ вывода `cin/cout`.

Класс `cl_child`:

- свойства/поля:
 - поле Значение закрытого свойства:
 - наименование — `pr`;
 - тип — Целочисленное;
 - модификатор доступа — `private`;
 - поле Значение открытого свойства:
 - наименование — `pu`;
 - тип — Целочисленное;
 - модификатор доступа — `public`;
- функционал:
 - метод `Method_pu` — Необходима смена значений;
 - метод `Vivod` — Вывод закрытого и открытого значений;
 - метод `cl_child` — Присвоение значений.

Класс `cl_parent`:

- свойства/поля:
 - поле Значение закрытого свойства:
 - наименование — `pr`;
 - тип — Целочисленное;
 - модификатор доступа — `private`;

- о поле Значение закрытого свойства:
 - наименование — ru;
 - тип — Целочисленное;
 - модификатор доступа — public;
- функционал:
 - о метод Method_ru — Необходима смена значений;
 - о метод Vivod — Вывод закрытого и открытого значений;
 - о метод cl_parent — Присвоение значений;
 - о метод Method — Закрытое изменение.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	cl_child				
2	cl_parent				

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 Method_ru класса cl_child

Функционал: Необходима смена значений.

Параметры: int x, int y.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода Method_ru класса cl_child

№	Предикат	Действия	№ перехода
1		Присвоение rg значения x	2
2		Присвоение ru значение y	Ø

3.2 Алгоритм метода Vivod класса cl_child

Функционал: Вывод закрытого и открыто значений.

Параметры: нет.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода Vivod класса cl_child

№	Предикат	Действия	№ перехода
1		Вывод значени rg " " и ru	Ø

3.3 Алгоритм конструктора класса `cl_child`

Функционал: Присвоение значений.

Параметры: `int x`, `int y`.

Алгоритм конструктора представлен в таблице 4.

Таблица 4 – Алгоритм конструктора класса `cl_child`

№	Предикат	Действия	№ перехода
1		Параметризированный конструктор	Ø

3.4 Алгоритм метода `Method_ru` класса `cl_parent`

Функционал: Необходима смена значений.

Параметры: `int x`, `int y`.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода `Method_ru` класса `cl_parent`

№	Предикат	Действия	№ перехода
1		Присвоение <code>pr</code> значения <code>x</code>	2
2		Присвоение <code>ru</code> значение <code>y</code>	Ø

3.5 Алгоритм метода `Vivod` класса `cl_parent`

Функционал: Вывод закрытого и открыто значений.

Параметры: нет.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода *Vivod* класса *cl_parent*

№	Предикат	Действия	№ перехода
1		Вывод закрытого и открытого значений	Ø

3.6 Алгоритм конструктора класса *cl_parent*

Функционал: Присвоение значений.

Параметры: нет.

Алгоритм конструктора представлен в таблице 7.

Таблица 7 – Алгоритм конструктора класса *cl_parent*

№	Предикат	Действия	№ перехода
1		Параметризированный конструктор	Ø

3.7 Алгоритм метода *Method* класса *cl_parent*

Функционал: Закрытое изменение.

Параметры: нет.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 8.

Таблица 8 – Алгоритм метода *Method* класса *cl_parent*

№	Предикат	Действия	№ перехода
1		Умножение закрытого свойства на 2	Ø

3.8 Алгоритм функции *main*

Функционал: Основная функция.

Параметры: Отсутствует.

Возвращаемое значение: Целое.

Алгоритм функции представлен в таблице 9.

Таблица 9 – Алгоритм функции *main*

№	Предикат	Действия	№ перехода
1		Объявление целочисленных переменных a, b	2
2		Ввод a, b	3
3		Создание объекта cl_chlid класса cl_chlid	4
4		Вывод значений родительского объекта	5
5		Вывод значений производного объекта	6
6	a > 0	Перераспределить значения свойств производного объекта	7
			10
7		Перераспределить значения свойств родительского объекта	8
8		Вывод значений производного объекта	9
9		Вывод значений родительского объекта	∅
10		Перераспределить значения свойств родительского объекта	11
11		Перераспределить значения свойств производного объекта	12
12		Вывод значений родительского объекта	13
13		Вывод значений производного объекта	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-4.

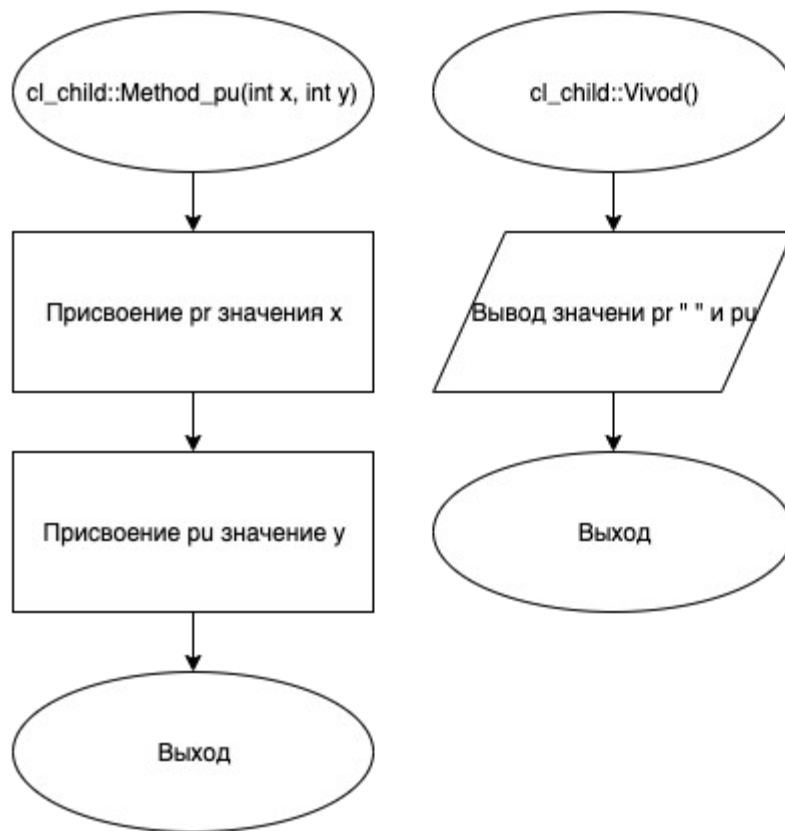


Рисунок 1 – Блок-схема алгоритма

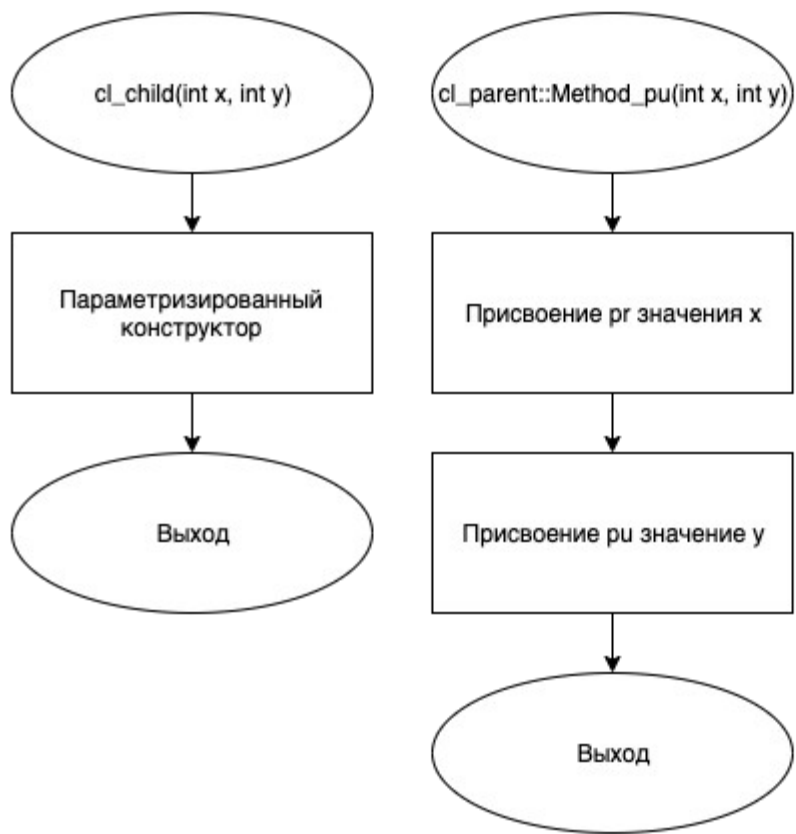


Рисунок 2 – Блок-схема алгоритма

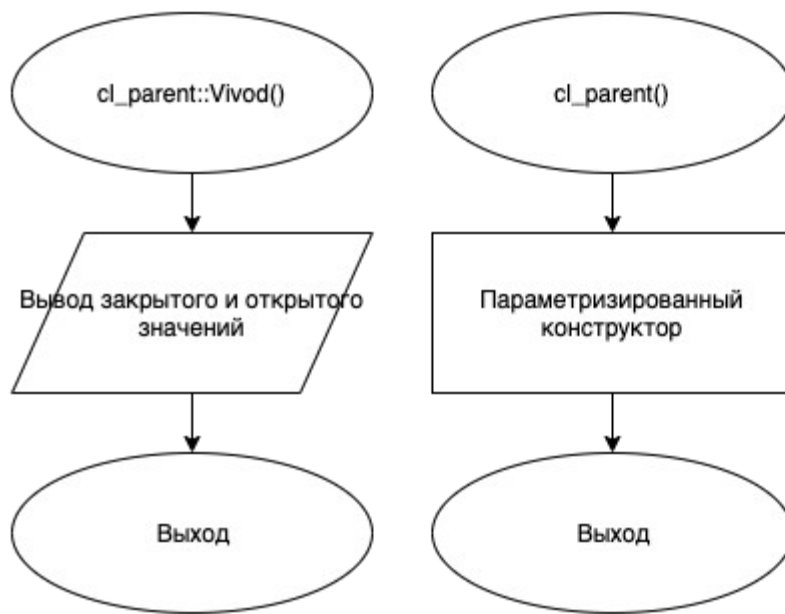


Рисунок 3 – Блок-схема алгоритма

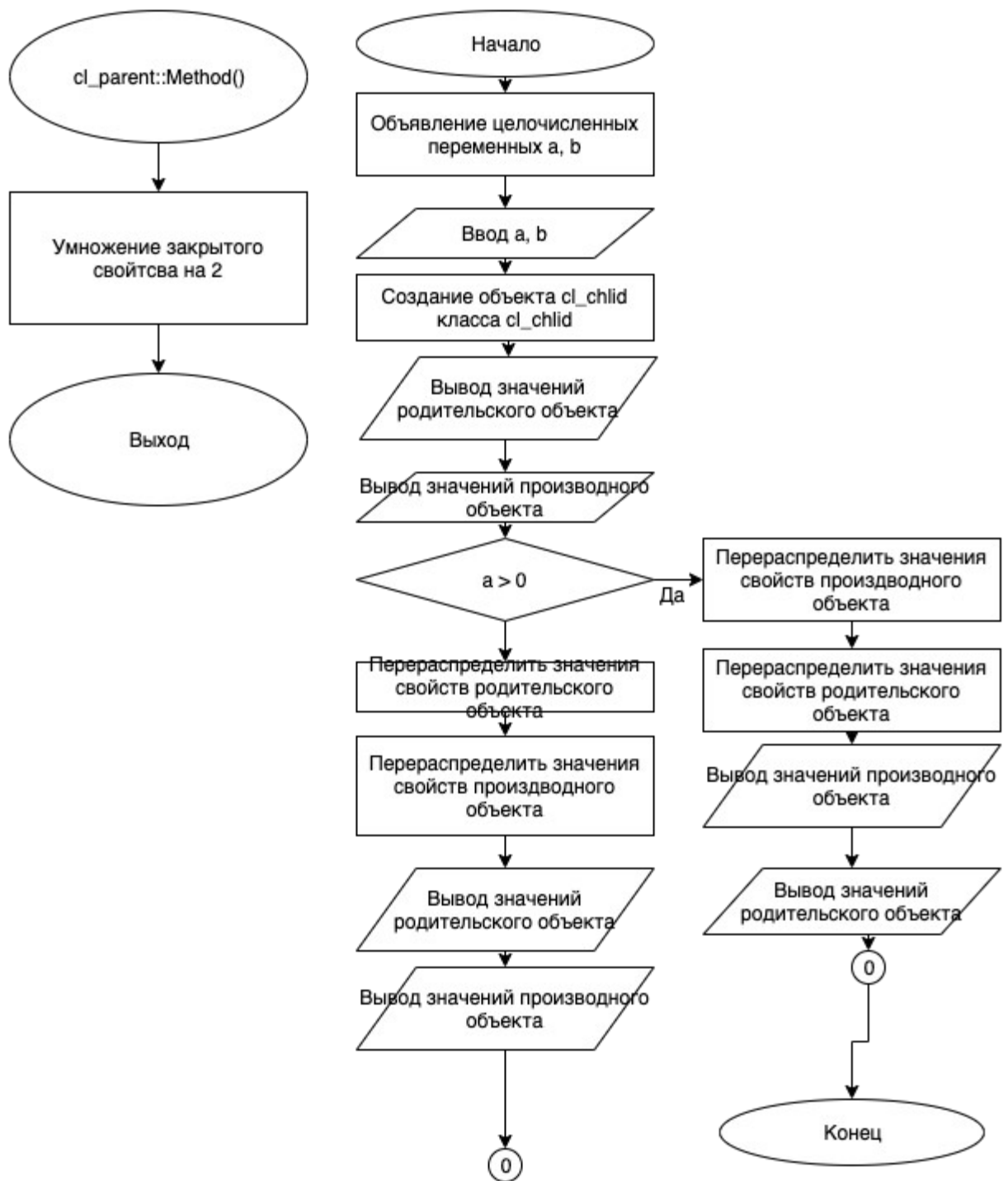


Рисунок 4 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл cl_child.cpp

Листинг 1 – cl_child.cpp

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

using namespace std;
cl_child::cl_child(int x, int y):cl_parent(x, y)
{
    pr = x;
    pu = y;
}
void cl_child::Method_pu(int x, int y)
{
    pr = x;
    pu = y;
}
void cl_child::Vivod()
{
    cout << pr << "    " << pu;
}
```

5.2 Файл cl_child.h

Листинг 2 – cl_child.h

```
#ifndef __CL_CHILD__H
#define __CL_CHILD__H
#include "cl_parent.h"

class cl_child: public cl_parent
{
private:
    int pr;
public:
    int pu;
    cl_child(int x, int y);
}
```



```
        void Method_pu(int x, int y);  
        void Vivod();  
};  
  
#endif
```

5.3 Файл cl_parent.cpp

Листинг 3 – cl_parent.cpp

```
#include "cl_parent.h"  
#include <iostream>  
  
using namespace std;  
  
void cl_parent::Method(int n)  
{  
    pr = n*2;  
}  
  
cl_parent::cl_parent(int x, int y)  
{  
    pu = y;  
    Method(x);  
}  
  
void cl_parent::Method_pu(int x, int y)  
{  
    pu = y;  
    Method(x);  
}  
  
void cl_parent::Vivod()  
{  
    cout << pr << "    " << pu;  
}
```

5.4 Файл cl_parent.h

Листинг 4 – cl_parent.h

```
#ifndef __CL_PARENT__H  
#define __CL_PARENT__H  
class cl_parent  
{
```

```

private:
    int pr;
    void Method(int n);
public:
    int pu;
    cl_parent(int x, int y);
    void Method_pu(int x, int y);
    void Vivod();
};

#endif

```

5.5 Файл main.cpp

Листинг 5 – main.cpp

```

#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include "cl_parent.h"
#include "cl_child.h"
using namespace std;
int main()
{
    int a,b;
    cin >> a >> b;
    cl_child cl_child(a,b);
    cl_child.cl_parent::Vivod();
    cout << endl;
    cl_child.cl_child::Vivod();
    cout << endl;
    if(a > 0)
    {
        cl_child.cl_child::Method_pu(a + 1, b + 1);
        cl_child.cl_parent::Method_pu(a - 1, b - 1);
        cl_child.Vivod();
        cout << endl;
        cl_child.cl_parent::Vivod();
    }
    else
    {
        cl_child.cl_parent::Method_pu(a + 1, b + 1);
        cl_child.cl_child::Method_pu(a - 1, b - 1);
        cl_child.cl_parent::Vivod();
        cout << endl;
        cl_child.cl_child::Vivod();
    }

    return(0);
}

```

}

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 10.

Таблица 10 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
8 5	16 5 8 5 9 6 14 4	16 5 8 5 9 6 14 4

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-avvora.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-avvora.ru/student/files/Prilozheniye_k_methodichke.pdf (дата обращения 05.05.2021).
4. Шилдт Г. С++: 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).