Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Полиморфизм в иерархии классов

Описать четыре класса которые последовательно наследуют друг друга, с номерами классов 1, 2, 3, 4. В каждом классе реализовать виртуальный метод с открытым доступом и одинаковым именем. Метод вычисляет значение многочлена степени номера класса и возвращает полученный результат. Коэффициенты и переменная многочлена целочисленные.

В основной функции реализовать алгоритм, в котором использовать один указатель на объект класса. Алгоритм:

1. Объявление указателя на объект класса.
2. Объявление четырех целочисленных переменных a1, a2, a3 a4, которые соответствуют коэффициентам многочлена (a1*x + a2*x*x + a3*x*x*x + a4*x*x*x*x).
3. Объявление целочисленной переменной x, которая соответствует переменной многочлена.
4. Ввод значения переменных a1, a2, a3 a4.
5. Создание объекта класса 4 посредством параметризированного конструктора, передав в качестве аргументов a1, a2, a3 a4. Обеспечить передачу необходимых коэффициентов объектам согласно наследственности классов.
6. Начало цикла
    - 6.1. Реализовать ввод значения переменной x.
    - 6.2. Если значение x равно нулю, то завершить цикл.
    - 6.3. Иначе, реализовать ввод значения номера класса.
    - 6.4. Согласно номеру класса вызвать метод вычисления многочлена

посредством объекта, который соответствует номеру класса и результат вывести.

7. Конец цикла.

1.1 Описание входных данных

Первая строка:

«целое число, значение a1» «целое число, значение a2» «целое число, значение a3» «целое число, значение a4»

Начиная со второй строки, построчно:

«целое число, значение x» «целое число, номер класса»

1.2 Описание выходных данных

Первая строка:

a1 = «целое число» a2 = «целое число» a3 = «целое число» a4 = «целое число»

Наименование коэффициента отделяется от предыдущего целого числа четырьмя пробелами.

Со второй строки и далее построчно:

Class «номер класса» F(«значение переменной x») = «значение многочлена»

Фрагменту «F(» предшествует 4 пробела

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект Object класса cl4 предназначен для ;
- функция main для Главная функция программы;
- функция App для реализация основного алгоритма работы программы;
- Объет потокового ввода/вывода cin/cout стандартной библиотеки iostream;
- Цикл while;
- Условная конструкция if else;
- Библиотека iostream;
- Пространство имён std.

Класс cl1:

- свойства/поля:
  - поле Поле коэффициента многочлена a1:
    - наименование — a1;
    - тип — int;
    - модификатор доступа — protected;
  - поле Поле коэффициента многочлена a2:
    - наименование — a2;
    - тип — int;
    - модификатор доступа — protected;
  - поле Поле коэффициента многочлена a3:
    - наименование — a3;
    - тип — int;
    - модификатор доступа — protected;
  - поле Поле коэффициента многочлена a4:
    - наименование — a4;

        - тип — int;
        - модификатор доступа — protected;
- функционал:
    - метод cl1 — параметризированный конструктор, задающий значения полям a1, a2, a3 и a4 переданными ему значениями;
    - метод get_count — вычисляет значение многочлена 1-ой степени (a1*x).

Класс cl2:

- функционал:
    - метод cl2 — параметризированный конструктор наследуемый от класса cl1;
    - метод get_count — вычисляет значение многочлена 2-ой степени (a1 * x + a2 * x * x).

Класс cl3:

- функционал:
    - метод cl3 — параметризированный конструктор наследуемый от класса cl2;
    - метод get_count — вычисляет значение многочлена 3-ей степени (a1 * x + a2 * x * x + a3 * x * x * x).

Класс cl4:

- функционал:
    - метод cl4 — параметризированный конструктор наследуемый от класса cl4;
    - метод get_count — вычисляет значение многочлена 4-ой степени (a1 * x + a2 * x * x + a3 * x * x * x + a4 * x * x * x * x).

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	cl1				
		cl2	public		2
2	cl2				
		cl3	public		3
3	cl3				
		cl4	public		4
4	cl4				

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 cl1

Функционал: параметризированный конструктор, задающий значения полям a1, a2, a3 и a4 переданными ему значениями.

Параметры: a1, a2, a3, a4 - целочисленные коэффициенты многочлена .

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса cl1

№	Предикат	Действия	№ перехода
1		Присваивание свойству a1 значения переданного параметра a1 посредством указателя this	2
2		Присваивание свойству a2 значения переданного параметра a2 посредством указателя this	3
3		Присваивание свойству a3 значения переданного параметра a3 посредством указателя this	4
4		Присваивание свойству a4 значения переданного параметра a4 посредством указателя this	∅

3.2 Алгоритм метода get_count класса cl1

Функционал: вычисляет значение многочлена 1-ой степени (a1 * x).

Параметры: x - целочисленная переменная, значение которой используется для вычисления многочлена.

Возвращаемое значение: целое, значение многочлена 1-ой степени (a1 * x).

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода get_count класса cl1

№	Предикат	Действия	№ перехода
1		Переход на новую строку	2
2		Вывод: "cl1"	3
3		Возврат значения многочлена a1 * x	∅

3.3 Алгоритм конструктора класса cl2

Функционал: параметризированный конструктор наследуемый от класса cl1.

Параметры: a1, a2, a3, a4 - целочисленные коэффициенты многочлена .

Алгоритм конструктора представлен в таблице 4.

Таблица 4 – Алгоритм конструктора класса cl2

№	Предикат	Действия	№ перехода
1		Вызов конструктора cl1 класса cl1 со всеми параметрами (a1, a2, a3, a4)	∅

3.4 Алгоритм метода get_count класса cl2

Функционал: вычисляет значение многочлена 2-ой степени (a1 * x + a2 * x * x).

Параметры: x - целочисленная переменная, значение которой используется для вычисления многочлена.

Возвращаемое значение: целое, значение многочлена 2-ой степени (a1 * x + a2 * x * x).

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода get_count класса cl2

№	Предикат	Действия	№ перехода
1		Переход на новую строку	2
2		Вывод: "cl2"	3
3		Возврат значения многочлена a1 * x + a2 * x * x	∅

3.5 Алгоритм конструктора класса cl3

Функционал: параметризированный конструктор наследуемый от класса cl2.

Параметры: a1, a2, a3, a4 - целочисленные коэффициенты многочлена .

Алгоритм конструктора представлен в таблице 6.

Таблица 6 – Алгоритм конструктора класса cl3

№	Предикат	Действия	№ перехода
1		Вызов конструктора cl1 класса cl2 со всеми параметрами (a1, a2, a3, a4)	∅

3.6 Алгоритм метода get_count класса cl3

Функционал: вычисляет значение многочлена 3-ой степени (a1 * x + a2 * x * x + a3 * x * x * x).

Параметры: x - целочисленная переменная, значение которой используется для вычисления многочлена.

Возвращаемое значение: целое, значение многочлена 3-ой степени (a1 * x + a2 * x * x + a3 * x * x * x).

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода get_count класса cl3

№	Предикат	Действия	№ перехода
1		Переход на новую строку	2
2		Вывод: "cl3"	3
3		Возврат значения многочлена a1 * x + a2 * x * x + a3 * x * x * x	∅

3.7 Алгоритм конструктора класса cl4

Функционал: параметризированный конструктор наследуемый от класса cl4.

Параметры: a1, a2, a3, a4 - целочисленные коэффициенты многочлена .

Алгоритм конструктора представлен в таблице 8.

Таблица 8 – Алгоритм конструктора класса cl4

№	Предикат	Действия	№ перехода
1		Вызов конструктора cl1 класса cl3 со всеми параметрами (a1, a2, a3, a4)	∅

3.8 Алгоритм метода get_count класса cl4

Функционал: вычисляет значение многочлена 4-ой степени (a1 * x + a2 * x * x + a3 * x * x * x + a4 * x * x * x * x).

Параметры: x - целочисленная переменная, значение которой используется для вычисления многочлена.

Возвращаемое значение: целое, значение многочлена 4-ой степени (a1 * x + a2 * x * x + a3 * x * x * x + a4 * x * x * x * x).

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода get_count класса cl4

№	Предикат	Действия	№ перехода
1		Переход на новую строку	2
2		Вывод: "cl4"	3
3		Возврат значения многочлена a1 * x + a2 * x * x + a3 * x * x * x + a4 * x * x * x * x	∅

3.9 Алгоритм функции App

Функционал: реализация основного алгоритма работы программы.

Параметры: нет.

Возвращаемое значение: Отсутствует.

Алгоритм функции представлен в таблице 10.

Таблица 10 – Алгоритм функции App

№	Предикат	Действия	№ перехода
1		Объявление указателя Object на объект класса cl4	2
2		Объявление переменных целого типа a1, a2, a3, a4, x, num, result	3
3		Ввод значений переменных a1, a2, a3, a4	4
4		Создание объекта Object класса cl4 с помощью параметризированного конструктора, с передачей ему значений коэффициентов многочлена a1, a2, a3 и a4, посредством оператора new	5
5		Вывод: "a1 = ",<<значение переменной a1>>," a2 = ",<<значение переменной a2>>," a3 = ",<<значение переменной a3>>," a4 = ",<<значение переменной a4>>	6
6		Ввод значения x и номера класса num	7
7	Значение переменной x		∅

№	Предикат	Действия	№ перехода
	равно 0		
	Значение переменной x не равно 0		8
8	Значение переменной num равно 4	Вызов метода get_count с передачей параметра x	12
	Значение переменной num не равно 4		9
9	Значение переменной num равно 3	Вызов метода get_count с передачей параметра x	12
	Значение переменной num не равно 3		10
10	Значение переменной num равно 2	Вызов метода get_count с передачей параметра x	12
	Значение переменной num не равно 2		11
11	Значение переменной num равно 1	Вызов метода get_count с передачей параметра x	12
	Значение переменной num не равно 1		6
12		Присвоение переменной result возвращённого значения метода get_count	13
13		Вывод: " F(",<<значение переменной x << ,") = ", <<значение переменной result>>	∅

3.10 Алгоритм функции main

Функционал: Главная функция программы.

Параметры: нет.

Возвращаемое значение: целое, индикация корректности работы программы.

Алгоритм функции представлен в таблице 11.

Таблица 11 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Вызов фнукции App	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-2.

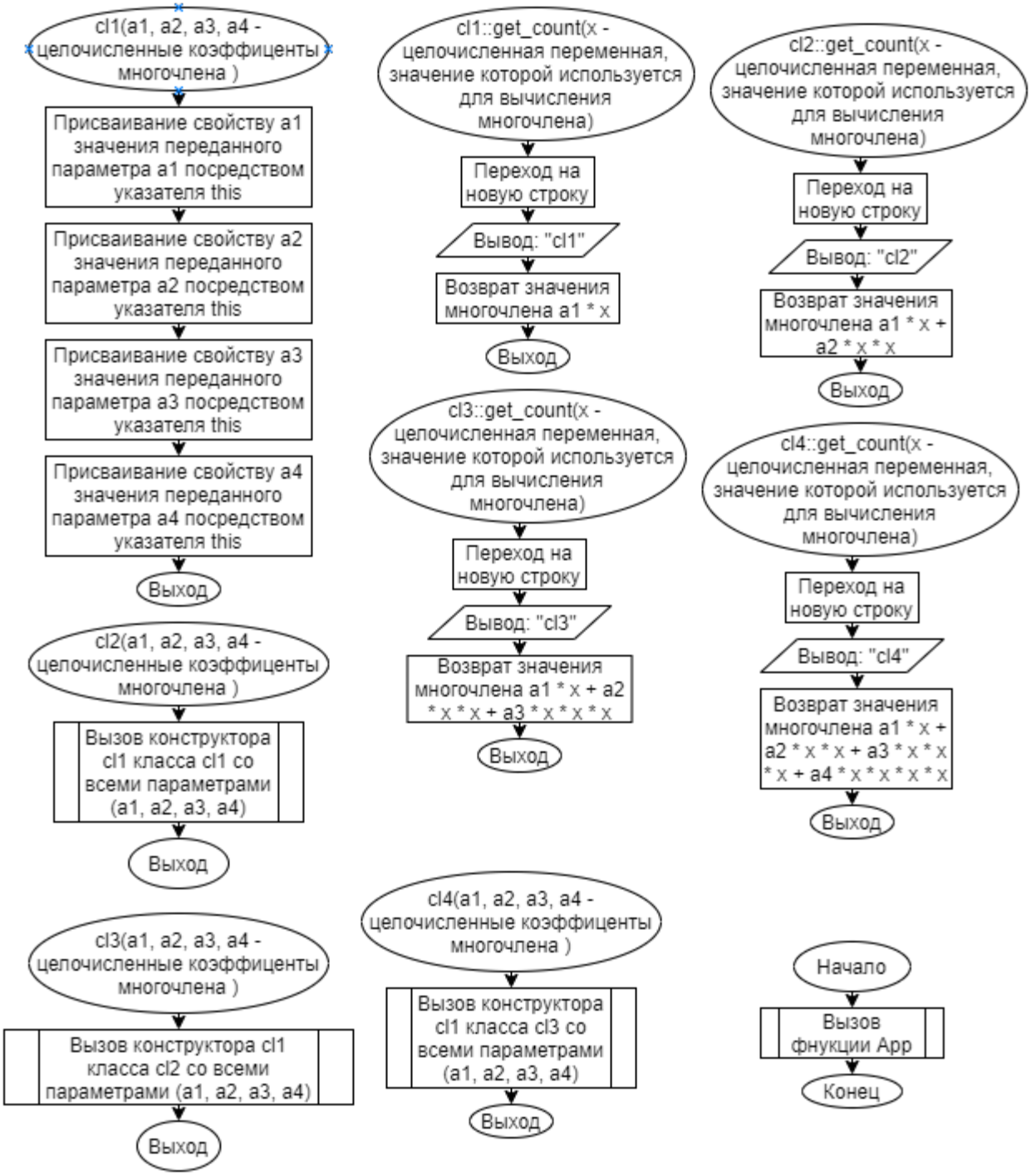

Рисунок 1 – Блок-схема алгоритма

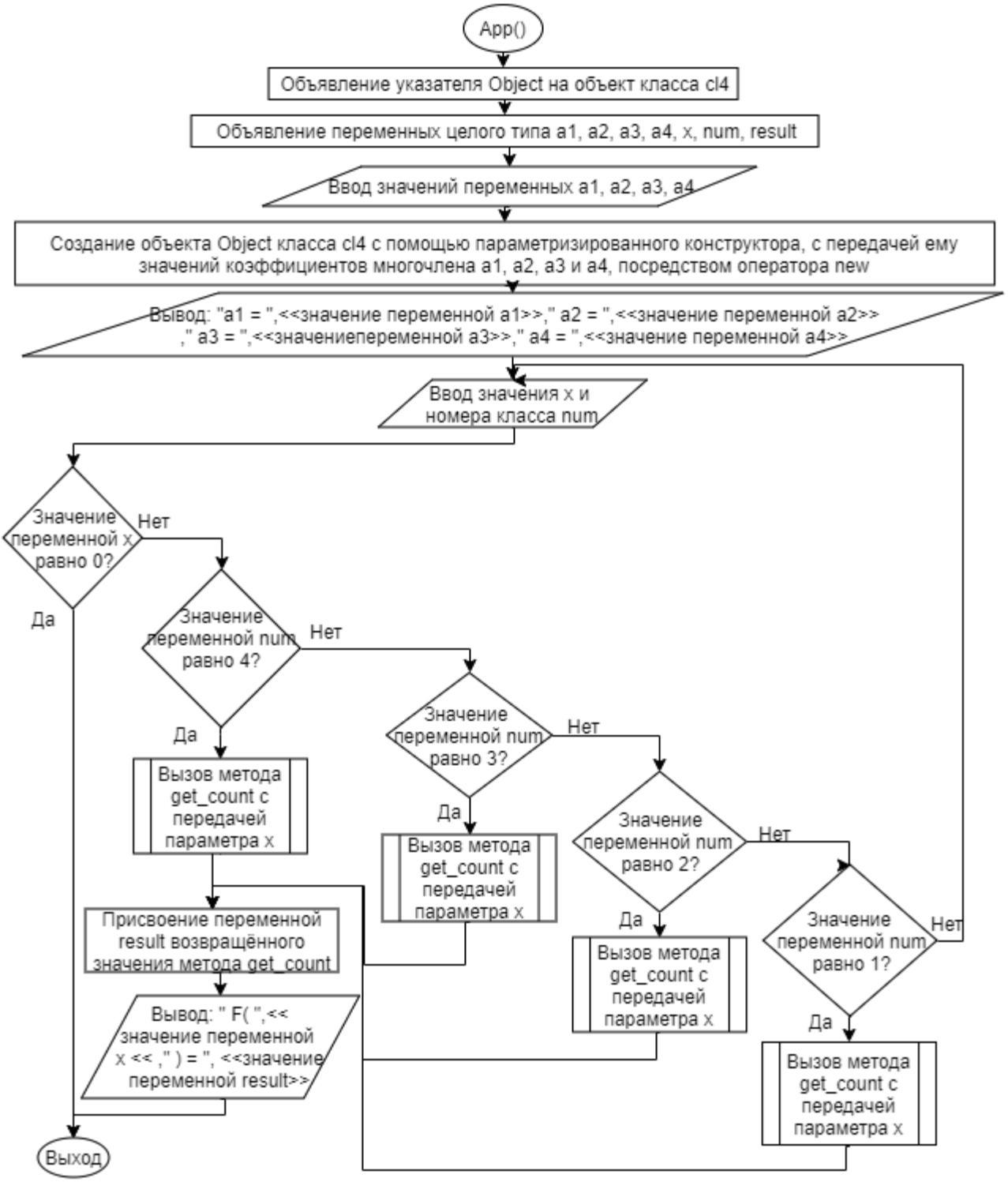

Рисунок 2 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл App.cpp

Листинг 1 – App.cpp

```
#include "cl4.h"
#include <iostream>
using namespace std;

void App()
{
   cl4* Object;
   int a1, a2, a3, a4, x, num, result;
   cin >> a1 >> a2 >> a3 >> a4;
   Object = new cl4(a1, a2, a3, a4);
   cout << "a1 =  " << a1 << "    " << "a2 =  " << a2 << "    " << "a3 =  "
<< a3 << "    " << "a4 =  " << a4;
   while(true)
   {
      cin >> x >> num;
      if (x == 0)
      {
         break;
      }
      if (num == 4)
      {
         result = Object -> get_count(x);
      }
      else if (num == 3)
      {
         result = Object -> cl3::get_count(x);
      }
      else if (num == 2)
      {
         result = Object -> cl2::get_count(x);
      }
      else if (num == 1)
      {
         result = Object -> cl1::get_count(x);
      }
      else
      {
         continue;
      }
```

```
            cout << "    F( " << x << " ) = " << result;
        }
    }
```

5.2 Файл App.h

Листинг 2 – App.h

```
    #ifndef __APP__H
    #define __APP__H

    void App();

    #endif
```

5.3 Файл cl1.cpp

Листинг 3 – cl1.cpp

```
    #include "cl1.h"
    #include <iostream>
    using namespace std;

    cl1::cl1(int a1, int a2, int a3, int a4)
    {
       this -> a1 = a1;
       this -> a2 = a2;
       this -> a3 = a3;
       this -> a4 = a4;
    }
    int cl1::get_count(int x)
    {
       cout << "\nClass 1";
       return a1 * x;
    }
```

5.4 Файл cl1.h

Листинг 4 – cl1.h

```
#ifndef __CL1__H
#define __CL1__H

#include <iostream>

class cl1
{
   public:
   cl1(int a1, int a2, int a3, int a4);
   virtual int get_count(int x);
   protected:
   int a1;
   int a2;
   int a3;
   int a4;
};

#endif
```

5.5 Файл cl2.cpp

Листинг 5 – cl2.cpp

```
#include "cl1.h"
#include "cl2.h"
#include <iostream>
using namespace std;

cl2 :: cl2(int a1, int a2, int a3, int a4): cl1(a1, a2, a3, a4){}
int cl2::get_count(int x)
{
   cout << "\nClass 2";
   return a1 * x + a2 * x * x;
}
```

5.6 Файл cl2.h

Листинг 6 – cl2.h

```
#ifndef __CL2__H
```

```
#define __CL2__H

#include <iostream>
#include "cl1.h"

class cl2 : public cl1
{
   public:
   cl2(int a1, int a2, int a3, int a4);
   virtual int get_count(int x);
};

#endif
```

5.7 Файл cl3.cpp

Листинг 7 – cl3.cpp

```
#include "cl2.h"
#include "cl3.h"
#include <iostream>
using namespace std;

cl3 :: cl3(int a1, int a2, int a3, int a4): cl2(a1, a2, a3, a4){}
int cl3::get_count(int x)
{
   cout << "\nClass 3";
   return a1 * x + a2 * x * x + a3 * x * x * x;
}
```

5.8 Файл cl3.h

Листинг 8 – cl3.h

```
#ifndef __CL3__H
#define __CL3__H

#include <iostream>
#include "cl2.h"

class cl3: public cl2
{
   public:
   cl3(int a1, int a2, int a3, int a4);
```

```
   virtual int get_count(int x);
};

#endif
```

5.9 Файл cl4.cpp

Листинг 9 – cl4.cpp

```
#include "cl3.h"
#include "cl4.h"
#include <iostream>
using namespace std;

cl4 :: cl4(int a1, int a2, int a3, int a4): cl3(a1, a2, a3, a4){}
int cl4::get_count(int x)
{
   cout << "\nClass 4";
   return a1 * x + a2 * x * x + a3 * x * x * x + a4 * x * x * x * x;
}
```

5.10 Файл cl4.h

Листинг 10 – cl4.h

```
#ifndef __CL4__H
#define __CL4__H

#include <iostream>
#include "cl3.h"
#include "cl4.h"

class cl4: public cl3
{
   public:
   cl4(int a1, int a2, int a3, int a4);
   virtual int get_count(int x);
};

#endif
```

5.11 Файл main.cpp

Листинг 11 – main.cpp

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

int main()
{
   // program here
   App();
   return(0);
}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 12.

Таблица 12 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
1 1 1 1 5 1 3 2 6 1 8 3 8 5 3 2 0 5 1 3 2 6 1 8 3	a1 = 1 a2 = 1 a3 = 1 a4 = 1 Class 1 F(5) = 5 Class 2 F(3) = 12 Class 1 F(6) = 6 Class 3 F(8) = 584 Class 2 F(3) = 12	a1 = 1 a2 = 1 a3 = 1 a4 = 1 Class 1 F(5) = 5 Class 2 F(3) = 12 Class 1 F(6) = 6 Class 3 F(8) = 584 Class 2 F(3) = 12
1 1 1 1 2 1 2 2 2 3 2 4 2 5 2 6 0	a1 = 1 a2 = 1 a3 = 1 a4 = 1 Class 1 F(2) = 2 Class 2 F(2) = 6 Class 3 F(2) = 14 Class 4 F(2) = 30	a1 = 1 a2 = 1 a3 = 1 a4 = 1 Class 1 F(2) = 2 Class 2 F(2) = 6 Class 3 F(2) = 14 Class 4 F(2) = 30

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).