Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Множественное наследование

Даны 8 классов, которые нумеруются от 1 до 8. Классы 2, 3, 4 и 5 наследованы от первого класса. Шестой класс от второго и третьего. Седьмой от четвертого и пятого. Восьмой от шестого и седьмого.

У каждого класса есть параметризированный конструктор с одним параметром строкового типа и закрытое свойство строкового типа для хранения наименования объекта класса. Значение данного свойства определяется в параметризированном конструкторе согласно шаблону:

```
«значение строкового параметра»_«номер класса»
```

У каждого класса есть метод в открытом разделе с одинаковым наименованием, который возвращает наименование объекта класса.

В реализации конструкторов со второго по восьмой класс, вызвать конструктор или конструкторы родительских классов. При вызове передать в качестве параметра выражение:

```
«параметр производного класса + «_» + «номер производного класса»
```

Например, для конструктора второго класса

```
cl_2 :: cl_2 ( string s_name ) : cl_1 ( s_name + “_2” )
```

В основной функции реализовать алгоритм:

1. Объявить один указатель на объект класса x.
2. Объявить переменную строкового типа.
3. Ввести значение строковой переменной. Вводимое значение является идентификатором.
4. Создать объект класса 8 посредством параметризированного конструктора, передав в качестве аргумента строковую переменную.
5. Адрес созданного объекта присвоить указателю на объект класса x.
6. Используя только указатель на объект класса x вывести имена всех объектов в

составе объекта класса 8 и имя самого объекта класса 8. Вывод выполнить построчно, упорядочивая согласно возрастанию номеров класса. Наименования объектов первого класса вывести последовательно для производных объектов 2,3.4 и 5 класса.

Наследственность реализовать так, чтобы всего объектов было 10 и обеспечить вывод по аналогии приведенному примеру вывода.

1.1 Описание входных данных

Первая строка:

«идентификатор»

Пример ввода

```
Object
```

1.2 Описание выходных данных

Построчно (одиннадцать строк):

«наименование объекта»

Пример вывода:

```
Object_8_6_2_1
Object_8_6_3_1
Object_8_1
Object_8_1
Object_8_6_2
Object_8_6_3
Object_8_7_4
Object_8_7_5
Object_8_6
Object_8_7
Object_8
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

7. Объекты стандартного потока ввода/вывода cin/cout
8. Операторы new/delete

Таблица иирархии классов

Таблица 1 – Иерархия наследования классов

№	Наименнования	Классы наследники	Модификатор доступа	Описание	Номер класса наследника	Комментарий
1	cl_1			Базовый класс		
		cl_2	public		2	
		cl_3	public		3	
		cl_4	public		4	
		cl_5	public		5	
2	cl_2			Класс производный от cl_1		
		cl_6	public		6	
3	cl_3			Класс производный от cl_1		
		cl_6	public		6	
4	cl_4			Класс производный от cl_1		С использованием

						virtual
		cl_7	public		7	
5	cl_5			Класс производный от cl_1		С использованием virtual
		cl_7	public		7	
6	cl_6			Класс производный от cl_2, cl_3		
		cl_8	public		8	
7	cl_7			Класс производный от cl_4, cl_5		
		cl_8	public		8	
8	cl_8			Класс производный от cl_6, cl_7		

- Классы cl_1, cl_2, cl_3, cl_4, cl_5, cl_6, cl_7, cl_8
  - Поля/свойства
    - Строковое поле
      - Наименование: name
      - Тип данных: строка
      - Модификатор доступа: закрытое
  - Методы:

- Конструктор cl_1, cl_2, cl_3, cl_4, cl_5, cl_6, cl_7, cl_8
    - Функционал: параметрический конструктор
- Метод Get_Name
    - Фунционал: возвращение строкового поля name

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 cl_1

Функционал: присваевание названия полю name.

Параметры: string s_name.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса cl_1

№	Предикат	Действия	№ перехода
1		name = s_name + "_" + номер объекта	∅

3.2 Алгоритм метода Get_Name класса cl_1

Функционал: Возвращение значения поля name.

Параметры: Отсутствуют.

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода Get_Name класса cl_1

№	Предикат	Действия	№ перехода
1		Возвращение значения поля name	∅

3.3 Алгоритм функции main

Функционал: Основная функция.

Параметры: Отсутствуют.

Возвращаемое значение: текст.

Алгоритм функции представлен в таблице 4.

Таблица 4 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление поля name типа string	2
2		Ввод поля name типа string	3
3		Объявление указателя obj на класс типа cl_8 с параметром name	4
4		Последовательный вывод с использованием метода Get_name()	5
5		Освобождение выделенной памяти под obj	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

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

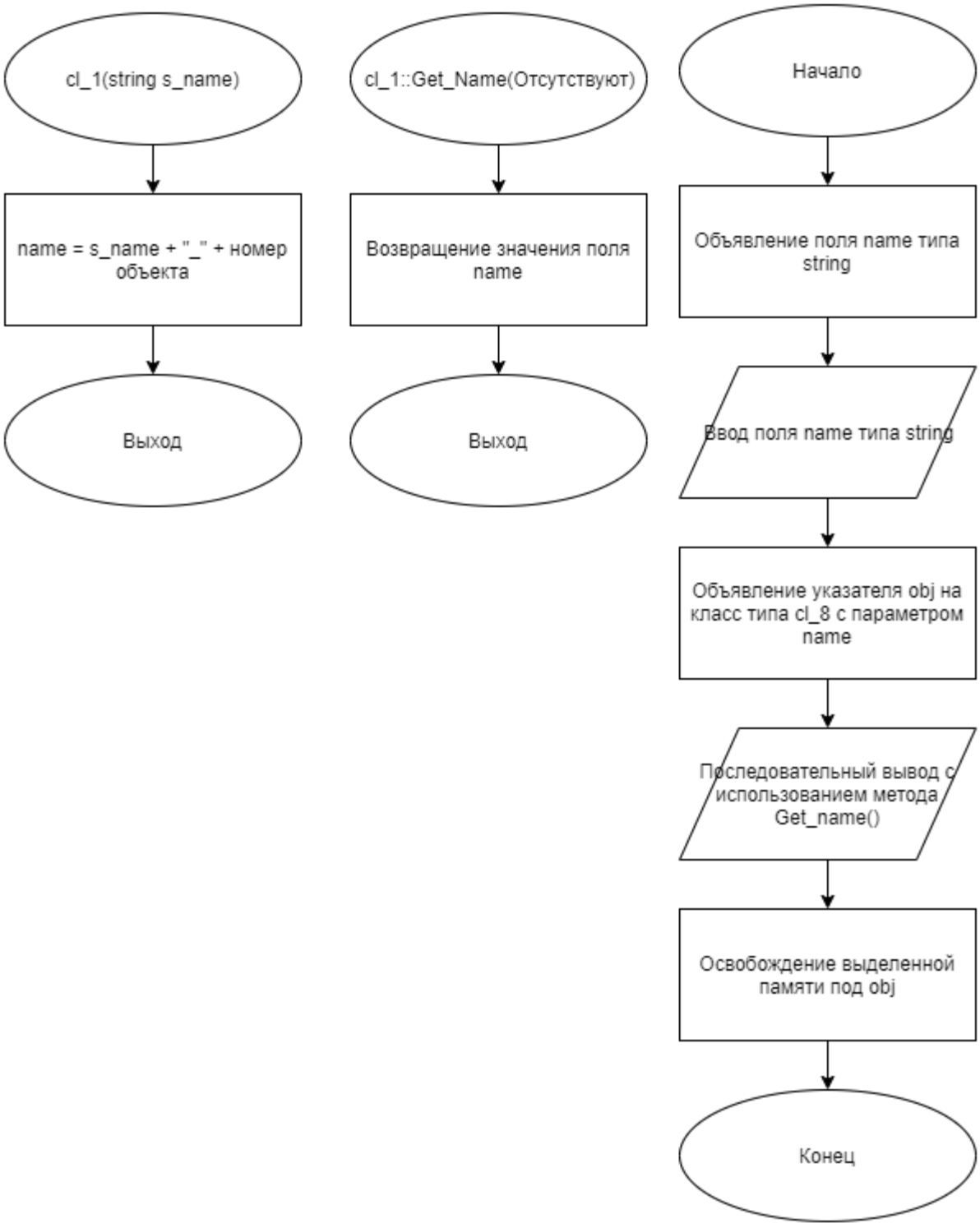

Рисунок 1 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл cl_1.cpp

Листинг 1 – cl_1.cpp

```
#include "cl_1.h"

cl_1::cl_1(string s_name)
{
	name = s_name + "_1";
}

string cl_1::Get_Name()
{
	return name;
}
```

5.2 Файл cl_1.h

Листинг 2 – cl_1.h

```
#ifndef __CL_1__H
#define __CL_1__H
#include <iostream>
#include <string>
using namespace std;

class cl_1
{
private:
	string name;
public:
	cl_1(string name);
	string Get_Name();
};
#endif
```

5.3 Файл cl_2.cpp

Листинг 3 – cl_2.cpp

```
#include "cl_2.h"

cl_2::cl_2(string s_name):cl_1(s_name +"_2")
{
	name = s_name + "_2";
}

string cl_2::Get_Name()
{
	return name;
}
```

5.4 Файл cl_2.h

Листинг 4 – cl_2.h

```
#ifndef __CL_2__H
#define __CL_2__H
#include "cl_1.h"

class cl_2 : public cl_1
{
private:
	string name;
public:
	cl_2(string name);
	string Get_Name();
};
#endif
```

5.5 Файл cl_3.cpp

Листинг 5 – cl_3.cpp

```
#include "cl_3.h"

cl_3::cl_3(string s_name):cl_1(s_name +"_3")
{
	name = s_name + "_3";
}

string cl_3::Get_Name()
{
```

```
	return name;
}
```

5.6 Файл cl_3.h

Листинг 6 – cl_3.h

```
#ifndef __CL_3__H
#define __CL_3__H
#include "cl_1.h"

class cl_3 : public cl_1
{
private:
	string name;
public:
	cl_3(string name);
	string Get_Name();
};
#endif
```

5.7 Файл cl_4.cpp

Листинг 7 – cl_4.cpp

```
#include "cl_4.h"

cl_4::cl_4(string s_name):cl_1(s_name +"_4")
{
	name = s_name + "_4";
}

string cl_4::Get_Name()
{
	return name;
}
```

5.8 Файл cl_4.h

Листинг 8 – cl_4.h

```
#ifndef __CL_4__H
#define __CL_4__H
#include "cl_1.h"

```

```
class cl_4 : public virtual cl_1
{
private:
	string name;
public:
	cl_4(string name);
	string Get_Name();
};
#endif
```

5.9 Файл cl_5.cpp

Листинг 9 – cl_5.cpp

```
#include "cl_5.h"

cl_5::cl_5(string s_name):cl_1(s_name +"_5")
{
	name = s_name + "_5";
}

string cl_5::Get_Name()
{
	return name;
}
```

5.10 Файл cl_5.h

Листинг 10 – cl_5.h

```
#ifndef __CL_5__H
#define __CL_5__H
#include "cl_1.h"

class cl_5 : public virtual cl_1
{
private:
	string name;
public:
	cl_5(string name);
	string Get_Name();
};
#endif
```

5.11 Файл cl_6.cpp

Листинг 11 – cl_6.cpp

```
#include "cl_6.h"

cl_6::cl_6(string s_name):cl_2(s_name +"_6"),cl_3(s_name +"_6")
{
	name = s_name + "_6";
}
string cl_6::Get_Name()
{
	return name;
}
```

5.12 Файл cl_6.h

Листинг 12 – cl_6.h

```
#ifndef __CL_6__H
#define __CL_6__H
#include "cl_2.h"
#include "cl_3.h"

class cl_6 : public cl_2, public cl_3
{
private:
	string name;
public:
	cl_6(string name);
	string Get_Name();
};
#endif
```

5.13 Файл cl_7.cpp

Листинг 13 – cl_7.cpp

```
#include "cl_7.h"

cl_7::cl_7(string  s_name):cl_4(s_name  +"_7"),cl_5(s_name  +"_7"),  cl_1(s_name  +
"_1")
{
	name = s_name + "_7";
}

string cl_7::Get_Name()
```

```
{
	return name;
}
```

5.14 Файл cl_7.h

Листинг 14 – cl_7.h

```
#ifndef __CL_7__H
#define __CL_7__H
#include "cl_4.h"
#include "cl_5.h"

class cl_7 : public cl_4, public cl_5
{
private:
	string name;
public:
	cl_7(string name);
	string Get_Name();
};
#endif
```

5.15 Файл cl_8.cpp

Листинг 15 – cl_8.cpp

```
#include "cl_8.h"

cl_8::cl_8(string  s_name):cl_6(s_name  +"_8"),cl_7(s_name  +"_8"),  cl_1(s_name  +
"_8")
{
	name = s_name + "_8";
}

string cl_8::Get_Name()
{
	return name;
}
```

5.16 Файл cl_8.h

Листинг 16 – cl_8.h

```
#ifndef __CL_8__H
```

```
#define __CL_8__H
#include "cl_6.h"
#include "cl_7.h"

class cl_8 : public cl_6, public cl_7
{
private:
      string name;
public:
      cl_8(string name);
      string Get_Name();
};
#endif
```

5.17 Файл main.cpp

Листинг 17 – main.cpp

```
#include "cl_8.h"
int main()
{
      string name;
      cin >> name;
      cl_8 *obj = new cl_8(name);
      cout << ((cl_1*)(cl_2*)(cl_6*)obj)->Get_Name() << endl;
      cout << ((cl_1*)(cl_3*)(cl_6*)obj)->Get_Name() << endl;
      cout << ((cl_1*)(cl_4*)(cl_7*)obj)->Get_Name() << endl;
      cout << ((cl_1*)(cl_5*)(cl_7*)obj)->Get_Name() << endl;
      cout << obj->cl_8::cl_6::cl_2::Get_Name() << endl;
      cout << obj->cl_8::cl_6::cl_3::Get_Name() << endl;
      cout << obj->cl_8::cl_7::cl_4::Get_Name() << endl;
      cout << obj->cl_8::cl_7::cl_5::Get_Name() << endl;
      cout << obj->cl_8::cl_6::Get_Name() << endl;
      cout << obj->cl_8::cl_7::Get_Name() << endl;
      cout << obj->Get_Name();
      delete obj;
}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 5.

Таблица 5 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
Object	Object_8_6_2_1 Object_8_6_3_1 Object_8_1 Object_8_1 Object_8_6_2 Object_8_6_3 Object_8_7_4 Object_8_7_5 Object_8_6 Object_8_7 Object_8	Object_8_6_2_1 Object_8_6_3_1 Object_8_1 Object_8_1 Object_8_6_2 Object_8_6_3 Object_8_7_4 Object_8_7_5 Object_8_6 Object_8_7 Object_8

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Васильев А.Н. Объектно-ориентированное программирование на C++. Издательство: Наука и Техника. Санкт-Петербург, 2016г. 543 стр.

2. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2017. — 624 с.

3. Методическое пособие для проведения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [Электронный ресурс] – URL: https://mirea.aco-avrora.ru/student/files/methodichescoe_posobie_dlya_laboratorny h_rabot_3.pdf (дата обращения 05.05.2021).

4. 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).

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).