Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Для организации иерархического построения объектов необходимо разработать базовый класс, который содержит функционал и свойства для построения иерархии объектов. В последующем, в приложениях использовать этот класс как базовый для всех создаваемых классов. Это позволит включать любой объект в состав дерева иерархии объектов.

Каждый объект на дереве иерархии имеет свое место и наименование. Не допускается для одного головного объекта одинаковые наименования в составе подчиненных объектов.

Создать базовый класс со следующими элементами:

- свойства:
  - наименование объекта (строкового типа);
  - указатель на головной объект для текущего объекта (для корневого объекта значение указателя равно nullptr);
  - динамический массив указателей на объекты, подчиненные к текущему объекту в дереве иерархии.
- функционал:
  - параметризированный конструктор с параметрами: указатель на объект базового класса, содержащий адрес головного объекта в дереве иерархии; строкового типа, содержащий наименование создаваемого объекта (имеет значение по умолчанию);
  - метод редактирования имени объекта. Один параметр строкового типа, содержит новое наименование объекта. Если нет дубляжа имени подчиненных объектов у головного, то редактирует имя и возвращает «истину», иначе возвращает «ложь»;
  - метод получения имени объекта;

- метод получения указателя на головной объект текущего объекта;
- метод вывода наименований объектов в дереве иерархии слева направо и сверху вниз;
- метод получения указателя на непосредственно подчиненный объект по его имени. Если объект не найден, то возвращает nullptr. Один параметр строкового типа, содержит наименование искомого подчиненного объекта.

Для построения дерева иерархии объектов в качестве корневого объекта используется объект приложение. Класс объекта приложения наследуется от базового класса. Объект приложение реализует следующий функционал:

- метод построения исходного дерева иерархии объектов (конструирования моделируемой системы);
- метод запуска приложения (начало функционирования системы, выполнение алгоритма решения задачи).

Написать программу, которая последовательно строит дерево иерархии объектов, слева направо и сверху вниз. Переход на новый уровень происходит только от правого (последнего) объекта предыдущего уровня. Для построения дерева использовать объекты двух производных классов, наследуемых от базового. Исключить создание объекта если его наименование совпадает с именем уже имеющегося подчиненного объекта у предполагаемого головного. Исключить добавление нового объекта, не последнему подчиненному предыдущего уровня.

Построчно, по уровням вывести наименования объектов построенного иерархического дерева.

Основная функция должна иметь следующий вид:

```
int main ( )
{
      cl_application   ob_cl_application ( nullptr ); // создание корневого
объекта
       ob_cl_application.build_tree_objects ( );          // конструирование
```

```
системы, построение дерева объектов
    return ob_cl_application.exec_app ( );         // запуск системы
}
```

Наименование класса cl_application и идентификатора корневого объекта ob_cl_application могут быть изменены разработчиком.

Все версии курсовой работы имеют такую основную функцию.

1.1 Описание входных данных

Первая строка:

«имя корневого объекта»

Вторая строка и последующие строки:

«имя головного объекта» «имя подчиненного объекта»

Создается подчиненный объект и добавляется в иерархическое дерево. Если «имя головного объекта» равняется «имени подчиненного объекта», то новый объект не создается и построение дерева объектов завершается.

Пример ввода:

```
Object_root
Object_root Object_1
Object_root Object_2
Object_root Object_3
Object_3 Object_4
Object_3 Object_5
Object_6 Object_6
```

Дерево объектов, которое будет построено по данному примеру:

```
Object_root
    Object_1
    Object_2
    Object_3
        Object_4
        Object_5
```

1.2 Описание выходных данных

Первая строка:

«имя корневого объекта»

Вторая строка и последующие строки имена головного и подчиненных объектов очередного уровня разделенных двумя пробелами.

«имя головного объекта» «имя подчиненного объекта»[[«имя подчиненного объекта»] …….]

Пример вывода:

```
Object_root
Object_root  Object_1  Object_2  Object_3
Object_3  Object_4  Object_5
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект класса cl_base предназначен для базовый класс;
- объект класса cl_application предназначен для класс корневого объекта;
- объект класса cl_1 предназначен для подчиненый класс класса cl_application.

Класс cl_base:

Собственные свойства и методы отсутствуют.

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_base

Функционал: создание объекта класса cl_base.

Параметры: нет.

Алгоритм конструктора представлен в таблице 1.

Таблица 1 – Алгоритм конструктора класса cl_base

№	Предикат	Действия	№ перехода
1		определение поля параметром p_head_obj	2
2		определение поля s_name параметром s_name	3
3	p_head_obj != nullptr	добавление в список p_head_obj родительского объекта для подчинееных объектов значение создаваемого объекта	∅
			∅

3.2 Алгоритм деструктора класса cl_base

Функционал: высвобождение памяти, выделенной на поле.

Параметры: нет.

Алгоритм деструктора представлен в таблице 2.

Таблица 2 – Алгоритм деструктора класса cl_base

№	Предикат	Действия	№ перехода
1		объявление переменной объекта p_head_object	2

№	Предикат	Действия	№ перехода
		универсиального типа отвечающий за хранение отдельного значения объекта списка	
2	p_head_object находится в списке p_head_object	Высвобождение памяти, выделенной под объект p_head_objects	2
			∅

3.3 Алгоритм метода set_name класса cl_base

Функционал: присвоение нового значения полю s_name.

Параметры: нет.

Возвращаемое значение: bool.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода set_name класса cl_base

№	Предикат	Действия	№ перехода
1		вызов метода get_head	2
2		вызов метода sub_object	3
3	возвращаемые значения не равные нулю	false	∅
		присвоение полю s_name значение параметра s_name	4
4		воззврат true	∅

3.4 Алгоритм метода get_name класса cl_base

Функционал: получение значения s_name.

Параметры: нет.

Возвращаемое значение: string.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода get_name класса cl_base

№	Предикат	Действия	№ перехода
1		возвращеие s_name	∅

3.5 Алгоритм метода get_sub_object класса cl_base

Функционал: получение значения p_sub_objects.

Параметры: нет.

Возвращаемое значение: cl_base*.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода get_sub_object класса cl_base

№	Предикат	Действия	№ перехода
1		возвращение p_sub_objects	∅

3.6 Алгоритм метода print_tree класса cl_base

Функционал: вывод дерева объектов.

Параметры: нет.

Возвращаемое значение: отсутствуети.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода print_tree класса cl_base

№	Предикат	Действия	№ перехода
1	p_sub_objects пустой		∅
		вывод переноса каретки и s_name	2
2		объявление переменной объекта p_sub_objects универсиального типа, отвечающий за хранение отдельного хначения объект списка	3
3	p_sub_object находится в	вывод двух пробелов и поля s_name объекта	3

№	Предикат	Действия	№ перехода
	списке p_sub_objects	p_sub_objects	
			4
4		вызов мтеода print_tree для последнего элемента массива p_sub_objects	∅

3.7 Алгоритм метода get_sub_object класса cl_base

Функционал: поиск объекта с полем s_name равным p_sub_object.

Параметры: нет.

Возвращаемое значение: cl_base*.

Алгоритм метода представлен в таблице 7.

Таблица 7 – Алгоритм метода get_sub_object класса cl_base

№	Предикат	Действия	№ перехода
1		объявление переменной объекта p_sub_object уиверсиального типа, отвечающий за храненеие отдельного значения объекта списка	2
2	p_sub_object находится в списке p_sub_objects		3
			6
3		вызов метода get_sub_object	4
4		вызов метода get_name	5
5	результат метода get_sub_object равно результату метода get_name		∅
			2
6		возвращение nellptr	∅

3.8 Алгоритм конструктора класса cl_1

Функционал: создание объекта на основе класса cl_1.

Параметры: нет.

Алгоритм конструктора представлен в таблице 8.

Таблица 8 – Алгоритм конструктора класса cl_1

№	Предикат	Действия	№ перехода
1		вызов параметризированного конструктора базового класса с аргументами p_sub_objects и s_name, являющимися параметром конструктора производного класса cl_1	∅

3.9 Алгоритм метода build_tree класса cl_application

Функционал: построение исхожнонго дерева иерархии объектов.

Параметры: нет.

Возвращаемое значение: отсутствует.

Алгоритм метода представлен в таблице 9.

Таблица 9 – Алгоритм метода build_tree класса cl_application

№	Предикат	Действия	№ перехода
1		объявление s_sub, s_head	2
2		инициализация объекта p_head класса cl_base типа указатель на базовый класс со значение типа указатель на текущий объект	3
3		инициализация объекта p_head класса cl_base	4
4		ввод значения переменной s_head	5
5		вызов метода set_name с параметром s_head	6
6		ввод значений s_head и s_sub	7

№	Предикат	Действия	№ перехода
7	значение переменной s_head = s_sub		∅
			8
8	значение объекта p_sub не равно nellptr и значение переменной s_head равно наименованию объекта p_sub	присвоение объекту p_head значение объекта p_sub	9
			9
9	значеие переменной s_head = наименованию объекта p_head и объекта, находящийся в списке подчинееных объектов p_head, с наименованием s_sub равен nullptr	присвоение объекту p_sub значение создаваемого объекта cl_1, путем вызова паармтеризированного корнструктора через опереторр new	6
			6

3.10 Алгоритм метода exec_app класса cl_applicaion

Функционал: запуск приложения и вывод всего деерва объектов на экран.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм метода представлен в таблице 10.

Таблица 10 – Алгоритм метода exec_app класса cl_applicaion

№	Предикат	Действия	№ перехода
1		вывод рехзультата выполнения метода get_name текужего оюъекта на экран	2
2		вызов метода print_tree	3

№	Предикат	Действия	№ перехода
3		возвраение нуля	∅

3.11 Алгоритм конструктора класса cl_application

Функционал: создание объекта на основе класса cl_application.

Параметры: нет.

Алгоритм конструктора представлен в таблице 11.

Таблица 11 – Алгоритм конструктора класса cl_application

№	Предикат	Действия	№ перехода
1		вызов параметризированного конструктора базового класса с фрагментом p_sub_objects являющимся параметром конструктора произвожног окдасса cl_1	∅

3.12 Алгоритм функции main

Функционал: основной алгоритм работы программы.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм функции представлен в таблице 12.

Таблица 12 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		создание объекта cl_application с аргументом равным nullptr	2
2		вызов метода build_tree_object класса cl_application	3
3		вызов метоа exec_app класса cl_application	4
4		возврат результата работы мтеода exec_app	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-9.

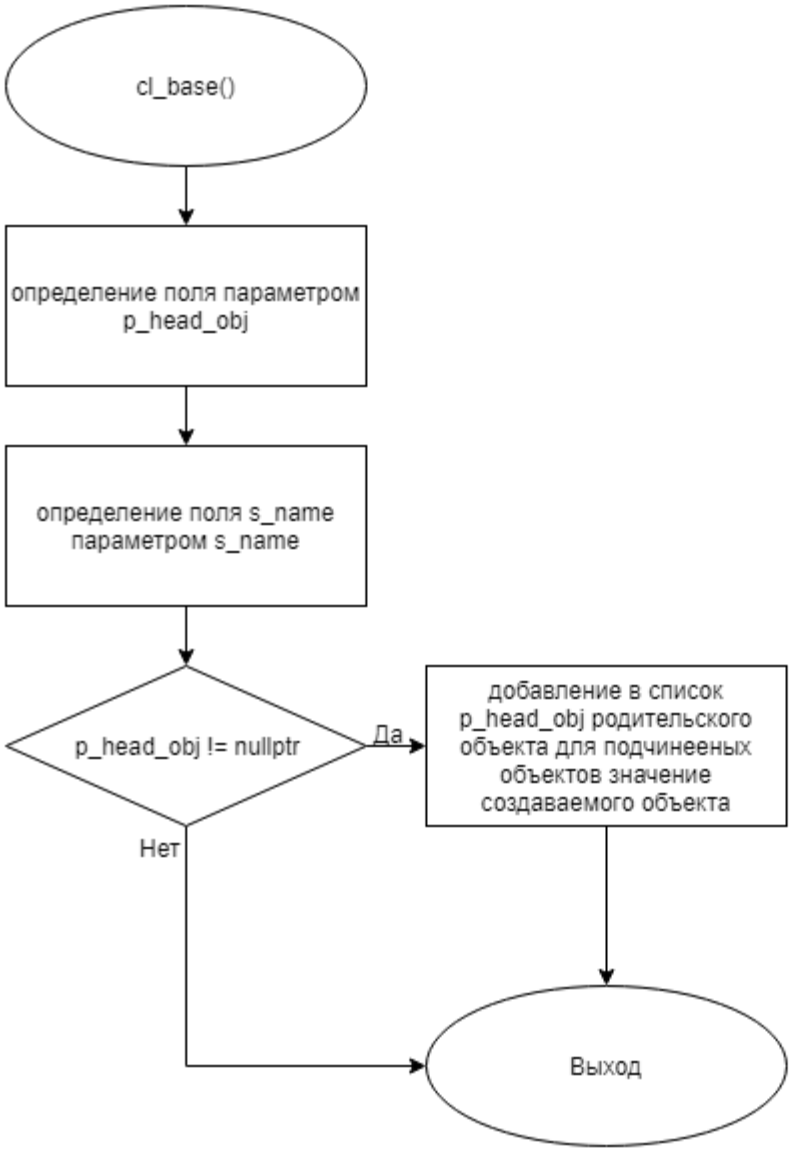

Рисунок 1 – Блок-схема алгоритма

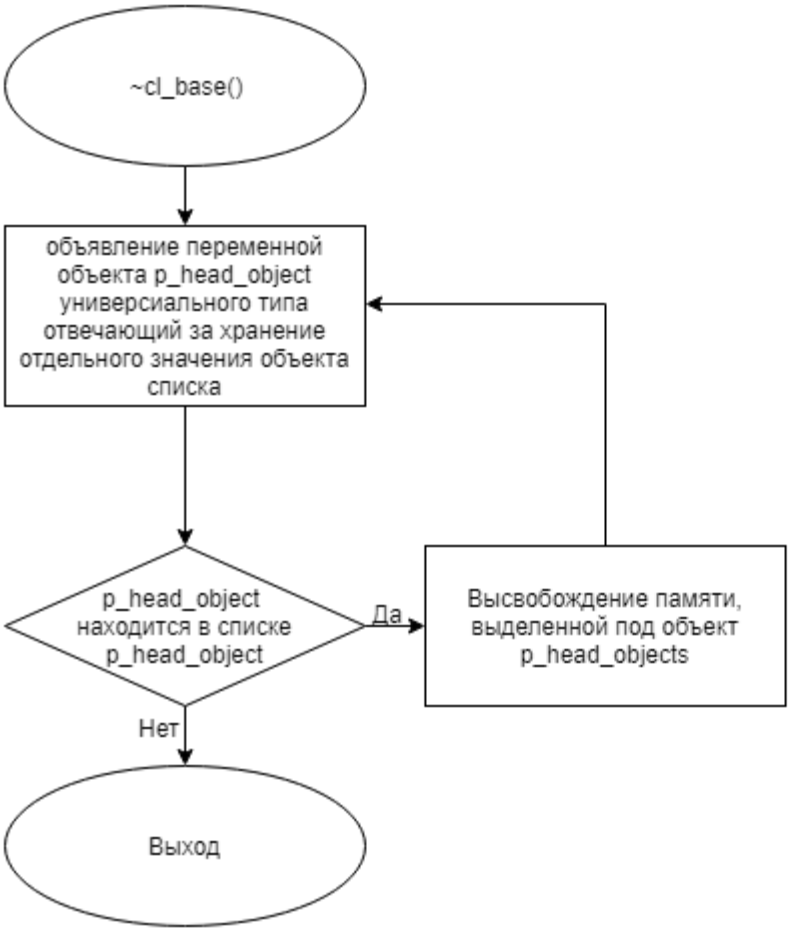

Рисунок 2 – Блок-схема алгоритма

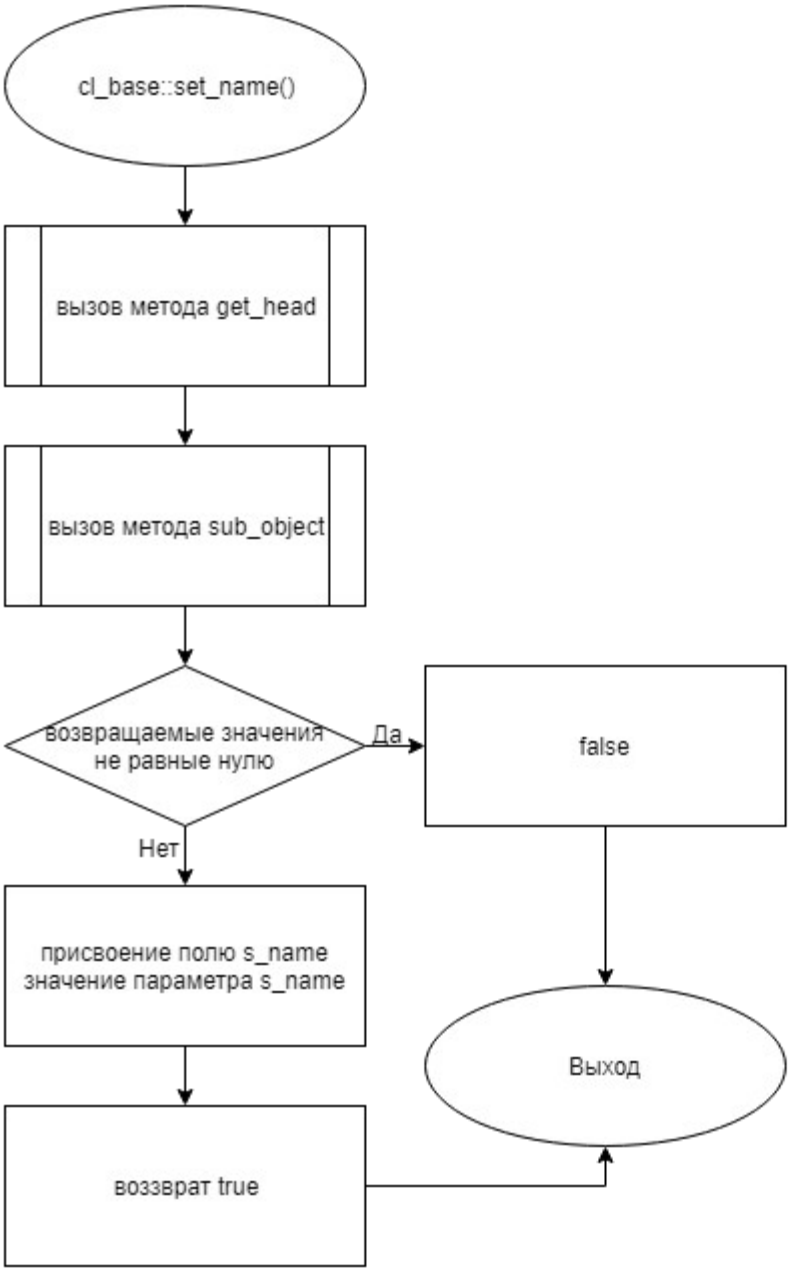

Рисунок 3 – Блок-схема алгоритма

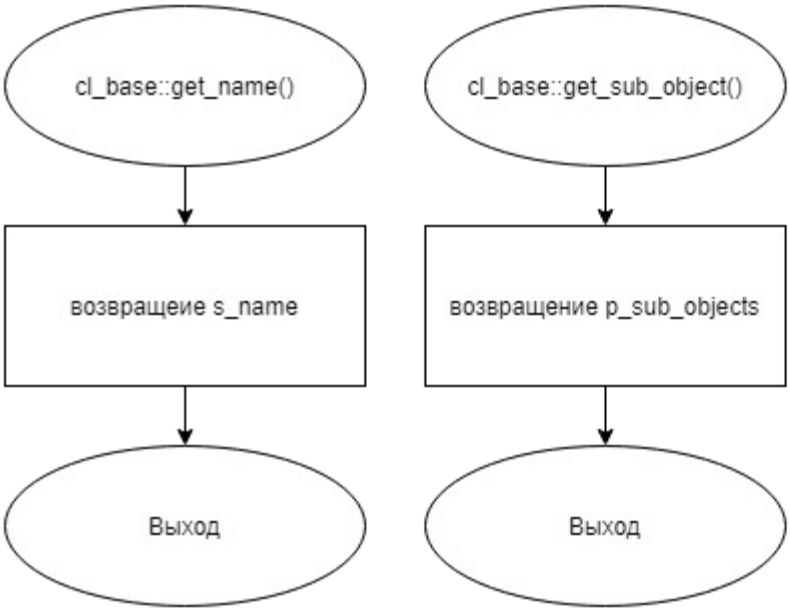

Рисунок 4 – Блок-схема алгоритма

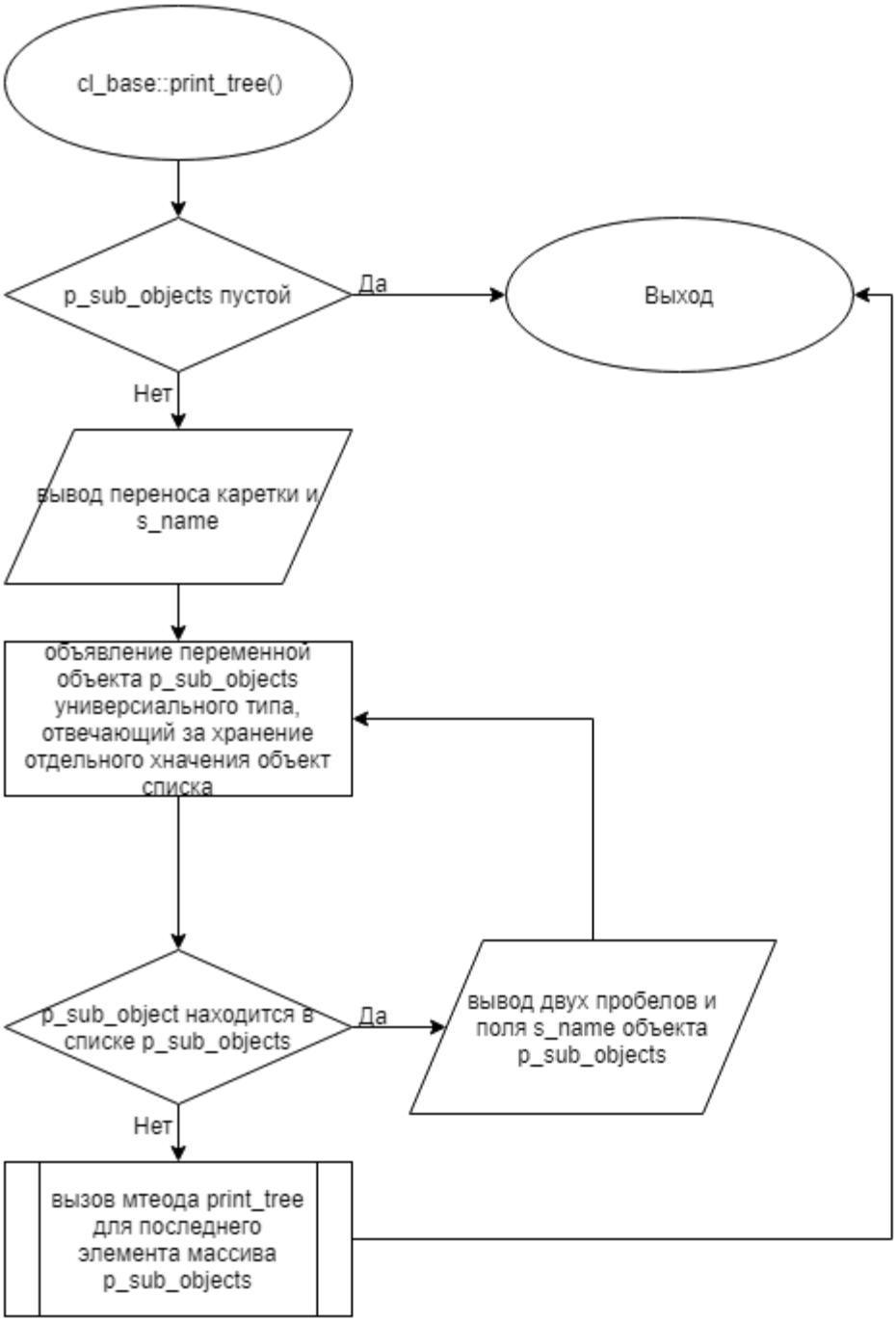

Рисунок 5 – Блок-схема алгоритма

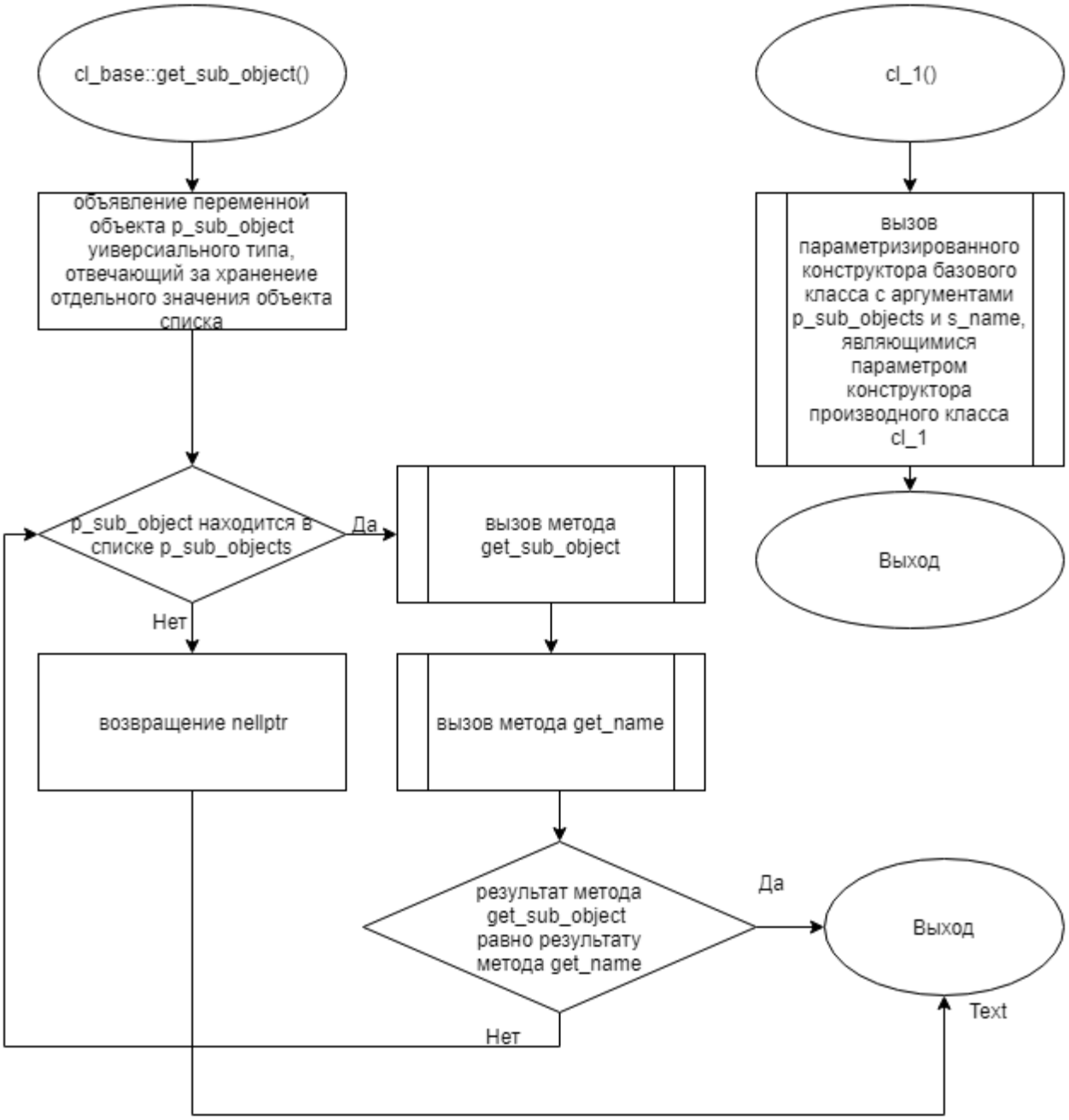

Рисунок 6 – Блок-схема алгоритма

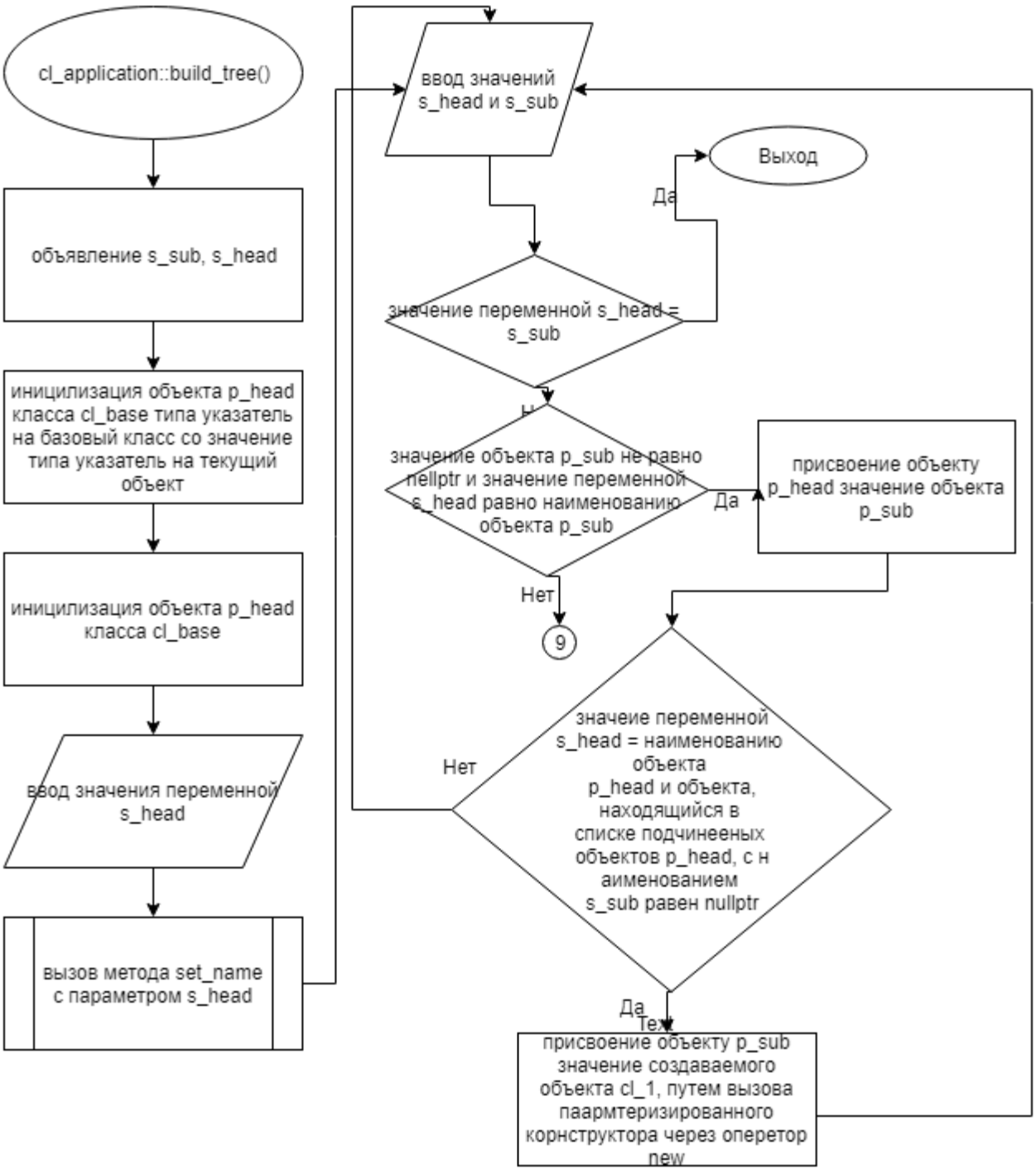

Рисунок 7 – Блок-схема алгоритма

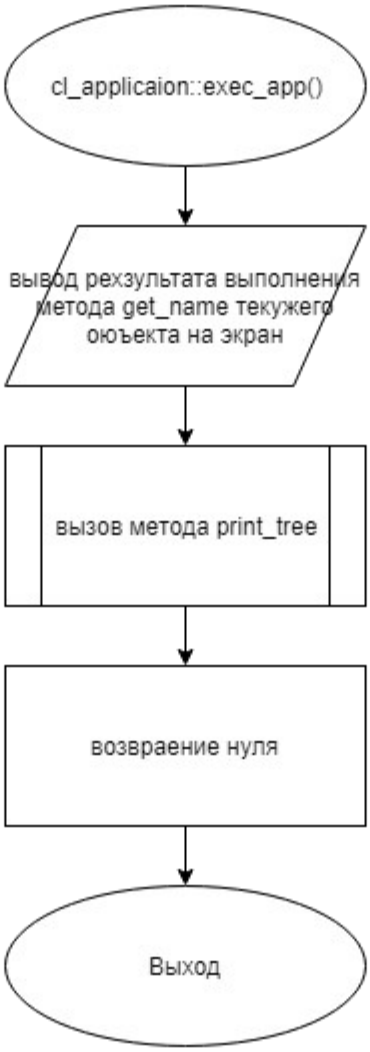

Рисунок 8 – Блок-схема алгоритма

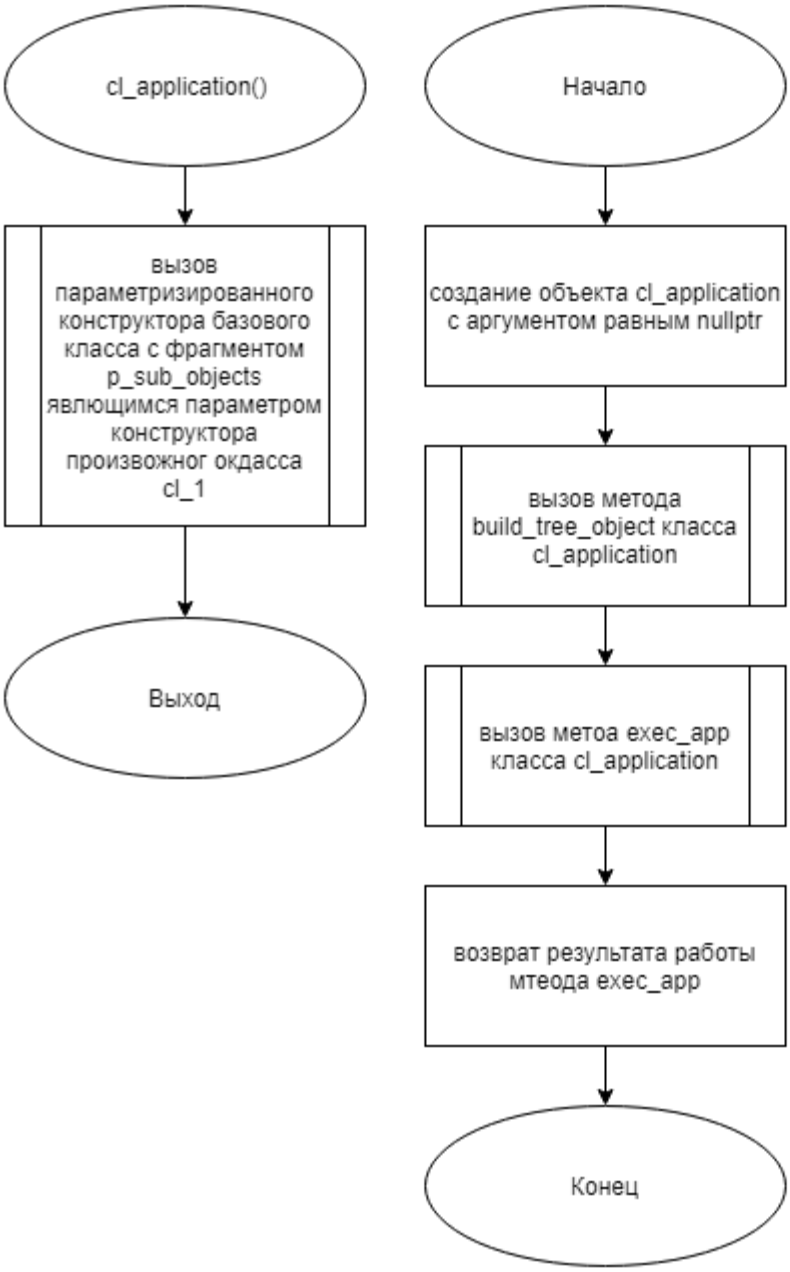

Рисунок 9 – Блок-схема алгоритма

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(cl_base* p_head_obj, string s_name):cl_base(p_head_obj, s_name){}
```

5.2 Файл cl_1.h

Листинг 2 – cl_1.h

```
#ifndef __CL_1__H
#define __CL_1__H
#include "cl_base.h"

class cl_1 : public cl_base
{
public:
   cl_1(cl_base* p_head_obj, string s_name);
};


#endif
```

5.3 Файл cl_application.cpp

Листинг 3 – cl_application.cpp

```
#include "cl_application.h"

cl_application::cl_application(cl_base* p_head_obj):cl_base(p_head_obj){}
```

```
void cl_application::build_tree_objects()
{
   /*
   описание: метож построения деерва иерархии объектов
   */
   string s_head, s_sub;
   cl_base* p_head;
   cl_base* p_sub = nullptr;

   cin >> s_head;
   this -> set_name(s_head);
   p_head = this;

   while (true)
   {
      cin >> s_head >> s_sub;

      if (s_head == s_sub)
         break;

      if (p_sub != nullptr && s_head == p_sub->get_name())
         p_head = p_sub;

      if  (p_head->get_sub_object(s_sub)  ==  nullptr  &&  s_head  ==  p_head-
>get_name())
         p_sub = new cl_1(p_head, s_sub);
   }
}

int cl_application::exec_app()
{
   cout << this -> get_name();
   this->print_tree();
   return 0;
}
```

5.4 Файл cl_application.h

Листинг 4 – cl_application.h

```
#ifndef __CL_APPLICATION__H
#define __CL_APPLICATION__H
#include "cl_base.h"
#include "cl_1.h"

class cl_application : public cl_base
{
public:
   cl_application(cl_base* p_head_obj);
   void build_tree_objects();
```

```
   int exec_app();
};



#endif
```

5.5 Файл cl_base.cpp

Листинг 5 – cl_base.cpp

```
#include "cl_base.h"
using namespace std;

cl_base::cl_base(cl_base* p_head_obj, string s_name )
{
   this->s_name = s_name;
   this->p_head_obj = p_head_obj;

   if (this->p_head_obj)
   {
      p_head_obj->p_sub_objects.push_back(this);
   }

};

cl_base::~cl_base()
{
   /*
   опиание: десруктор
   параметры
   */
   for (int i = 0; i < p_sub_objects.size(); i++)
   {
      delete p_sub_objects[i];
   }
}

bool cl_base::set_name(string s_new_name)
{
   /*
   описание: метод редактирования имени объекта
   параметры Ж
   S_new_name - новое имя узла дееова
   */
   if (this -> p_head_obj)
   {
      for (int i = 0; i < p_head_obj->p_sub_objects.size(); i ++)
      {
         if(p_head_obj->p_sub_objects[i] -> get_name() == s_new_name)
```

```
            {
               return false;
            }
         }
      }
      this->s_name = s_new_name;
      return true;
   }

   void cl_base::print_tree()
   {
      if(p_sub_objects.size() != 0)
      {
         cout << endl << this -> get_name();
         for (int i = 0; i < p_sub_objects.size(); i++)
         {
            cout << "  " << p_sub_objects[i]->get_name();
            p_sub_objects[i]->print_tree();
         }
      }
   }

   cl_base *cl_base::get_sub_object(string s_name)
   {
      /*
      опиание: получение указателя на непосредственно подчиненный объект по
   имени
      параметры
      s_name - имя искомого объекта
      */
      for (int i = 0; i < p_sub_objects.size(); i++)
         if (p_sub_objects[i] -> get_name() == s_name)
            return p_sub_objects[i];
      return nullptr;
   }

   string cl_base::get_name()
   {
      return s_name;
   }
```

5.6 Файл cl_base.h

Листинг 6 – cl_base.h

```
   #ifndef __cl_BASE__H
   #define __cl_BASE__H
   #include <iostream>
   #include <vector>
   using namespace std;
```

```
class cl_base
{
   string s_name;
   cl_base* p_head_obj;
   vector<cl_base*> p_sub_objects;
public:
   cl_base(cl_base* p_head_obj, string s_name = "Base_object");
   bool set_name(string s_new_name);
   string get_name();
   cl_base* get_head_obj();
   void print_tree();
   cl_base* get_sub_object(string s_name);
   ~cl_base();
   /*
   свойства:
наименование объекта (строкового типа);

указатель на головной объект для текущего объекта (для корневого объекта значение
указателя равно nullptr);

динамический массив указателей на объекты, подчиненные к текущему объекту в дереве иерархии.

функционал:
параметризированный конструктор с параметрами: указатель на объект базового класса,
содержащий адрес головного объекта в дереве иерархии;
строкового типа,
содержащий наименование создаваемого объекта (имеет значение по умолчанию);

метод редактирования имени объекта. Один параметр строкового типа, содержит новое
наименование объекта. Если нет дубляжа имени подчиненных объектов у головного,
то редактирует имя и возвращает «истину», иначе возвращает «ложь»;
метод получения имени объекта;

метод получения указателя на головной объект текущего объекта;

метод вывода наименований объектов в дереве иерархии слева направо и сверху вниз;

метод получения указателя на непосредственно подчиненный объект по его имени.

Если объект не найден, то возвращает nullptr. Один параметр строкового типа, содержит наименование искомого подчиненного объекта.
*/
};

#endif
```

5.7 Файл main.cpp

Листинг 7 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include "cl_application.h"
using namespace std;


int main()
{
   cl_application ob_cl_cl_application (nullptr);//создание корневого объекта
   ob_cl_cl_application.build_tree_objects();  //  конструирование  системы,
построение дерва объектов
   return ob_cl_cl_application.exec_app(); // запуск системы
}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 13.

Таблица 13 – Результат тестирования программы

Входные данные	**Ожидаемые выходные данные**	**Фактические выходные данные**
Object_root Object_root Object_1 Object_root Object_2 Object_root Object_3 Object_3 Object_4 Object_3 Object_5 Object_6 Object_6	Object_root Object_root Object_1 Object_2 Object_3 Object_3 Object_4 Object_5	Object_root Object_root Object_1 Object_2 Object_3 Object_3 Object_4 Object_5

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).