Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Разработать систему, которая демонстрирует возможность размещения и использования объекта одного класса в составе объекта другого класса. При этом, жизненный цикл внутреннего объекта полностью реализуется в рамках внешнего объекта.

Спроектировать внешний объект, который содержит внутренний объект в закрытом доступе. Содержит свойство строкового типа для хранения наименования объекта в закрытом доступе.

Внутренний объект имеет одно целочисленное свойство, в закрытом доступе, для хранения суммы полученных целочисленных значений.

Внутренний объект имеет метод в открытом доступе, с одним целочисленным параметром, который суммирует полученные по параметру значения.

В деструкторе внутреннего объекта выводиться значение полученной в процессе функционирования объекта суммы.

Внешний объект имеет параметризированный конструктор, с параметром строкового типа, который передает наименование объекта.

Внешний объект имеет метод с целочисленным параметром в открытом доступе, который передает значение этого параметра другому внешнему объекту посредством вызова метода и последующего суммирования.

Внешний объект имеет деструктор, который выводит "Object_" и наименование объекта.

Организовать связи между объектами.

Для организации связи между объектами добавить не более одного метода в открытом доступе к внешнему объекту, не более одного свойства и одного метода в открытом доступе внутреннему объекту,

Алгоритм конструирования и отработки системы:

1. Объявляется строковая переменная.

2. Вводиться наименование первого внешнего объекта.

3. Объявляется первый объект внешнего класса, с использованием параметризированного конструктора.

4. Вводиться наименование второго внешнего объекта.

5. Объявляется второй объект внешнего класса, с использованием параметризированного конструктора.

6. Организуется связь между объектами.

7. Объявляется целочисленная переменная, для хранения номера внешнего объекта.

8. Объявляется целочисленная переменная, для хранения целочисленного значения.

9. Начало цикла.

9.1. Вводится номер внешнего объекта и целочисленное значение.

9.2. Если номер внешнего объекта равно 0, работа цикла завершается.

9.3. Вызывается метод внешнего объекта согласно введенному номеру и передается в качестве аргумента переменная с целочисленным значением, которое посредством организованной связи предается другому внешнему объекту.

10. Конец цикла.

11. Завершается работа системы.

1.1 Описание входных данных

Первая строка:

«наименование первого объекта»

Вторая строка:

```
«наименование второго объекта»
```

Начиная с третьей строки, построчно:

```
«целое число, номер объекта» «целое число»
```

Последняя строка:

```
0
```

Пример ввода:

```
ffff
ssss
1 1
2 2
2 2
1 4
0
```

1.2 Описание выходных данных

Первая строка:

```
Object_«наименование второго объекта»    SUMM = «целое число, значение суммы»
```

Вторая строка:

```
Object_«наименование первого объекта»    SUMM = «целое число, значение суммы»
```

Курсор установливается в начале третьей строки.

Пример вывода:

```
Object_ssss    SUMM = 5
Object_ffff    SUMM = 4
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект outer1 класса Outer предназначен для демонстрации работы программы;
- объект outer2 класса Outer предназначен для демонстрации работы программы;
- функция main для определения входной точки программы;
- библиотека ввода-вывода;
- заголовочные файлы;
- строки;
- указатели;
- классы.

Класс Inner:

- свойства/поля:
    - поле хранит сумму:
        - наименование — m_sum;
        - тип — int;
        - модификатор доступа — private;
- функционал:
    - метод Sum — суммирует полученные по параметру значения;
    - метод ~Inner — выводит значение полученной в процессе функционирования объекта суммы.

Класс Outer:

- свойства/поля:
    - поле хранит наименование объекта:
        - наименование — m_name;

        - тип — std::string;
        - модификатор доступа — private;
    - поле связь с другим внешним объектом:
        - наименование — m_other;
        - тип — Outer*;
        - модификатор доступа — private;
    - поле внутренний класс:
        - наименование — inner;
        - тип — Inner*;
        - модификатор доступа — private;
- функционал:
    - метод Outer — инициализирует m_name и inner;
    - метод SetOther — инициализирует указатель на объект класса m_other;
    - метод PassValue — передает значение параметра другому внешнему объекту посредством вызова метода и последующего суммирования;
    - метод ~Outer — выводит имя объекта.

Таблица 1 – Иерархия наследования классов

№	Имя класса	Классы-наследники	Модификатор доступа при наследовании	Описание	Номер
1	Inner				
2	Outer				

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 Sum класса Inner

Функционал: Суммирует полученные по параметру значения.

Параметры: int num.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода Sum класса Inner

№	Предикат	Действия	№ перехода
1		Инициализация m_sum += num	∅

3.2 Алгоритм деструктора класса Inner

Функционал: Выводит значение полученной в процессе функционирования объекта суммы.

Параметры: нет.

Алгоритм деструктора представлен в таблице 3.

Таблица 3 – Алгоритм деструктора класса Inner

№	Предикат	Действия	№ перехода
1		Вывод "SUMM = " m_sum	∅

3.3 Алгоритм конструктора класса Outer

Функционал: Инициализирует m_name и inner.

Параметры: std::string name.

Алгоритм конструктора представлен в таблице 4.

Таблица 4 – Алгоритм конструктора класса Outer

№	Предикат	Действия	№ перехода
1		Инициализация m_name = name	2
2		Инициализация указателя на объект inner класса Inner	∅

3.4 Алгоритм метода SetOther класса Outer

Функционал: Инициализирует указатель на объект класса m_other.

Параметры: Outer* outer.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода SetOther класса Outer

№	Предикат	Действия	№ перехода
1		Инициализация m_other = outer	∅

3.5 Алгоритм метода PassValue класса Outer

Функционал: Передает значение параметра другому внешнему объекту посредством вызова метода и последующего суммирования.

Параметры: int num.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода PassValue класса Outer

№	Предикат	Действия	№ перехода
1	m_other != nullptr	Вызов m_other->inner->Sum(num)	∅
2			∅

3.6 Алгоритм деструктора класса Outer

Функционал: Выводит имя объекта.

Параметры: нет.

Алгоритм деструктора представлен в таблице 7.

Таблица 7 – Алгоритм деструктора класса Outer

№	Предикат	Действия	№ перехода
1		Вывод "Object_" m_name " "	2
2		Удаление объекта inner с помощью оператора delete с освобождением памяти указателя	∅

3.7 Алгоритм функции main

Функционал: Входная точка программы.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм функции представлен в таблице 8.

Таблица 8 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление переменной name строкового типа	2
2		Ввод name	3
3		Инициализация объекта outer1 класса Outer с передачей name в качестве параметра	4

№	Предикат	Действия	№ перехода
4		Ввод name	5
5		Инициализация объекта outer2 класса Outer с передачей name в качестве параметра	6
6		Вызов outer1->SetOther(outer2)	7
7		Вызов outer2->SetOther(outer1)	8
8		Объявление целочисленных переменных objNumber и value	9
9	true	Ввод objNumber	10
			13
10	objNumber = 0		13
			11
11		Ввод value	12
12	objNumber = 1	Вызов outer1->PassValue(value)	9
	objNumber = 2	Вызов outer2->PassValue(value)	9
13		Удаление объекта outer2 с помощью оператора delete с освобождением памяти указателя	14
14		Удаление объекта outer1 с помощью оператора delete с освобождением памяти указателя	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-5.

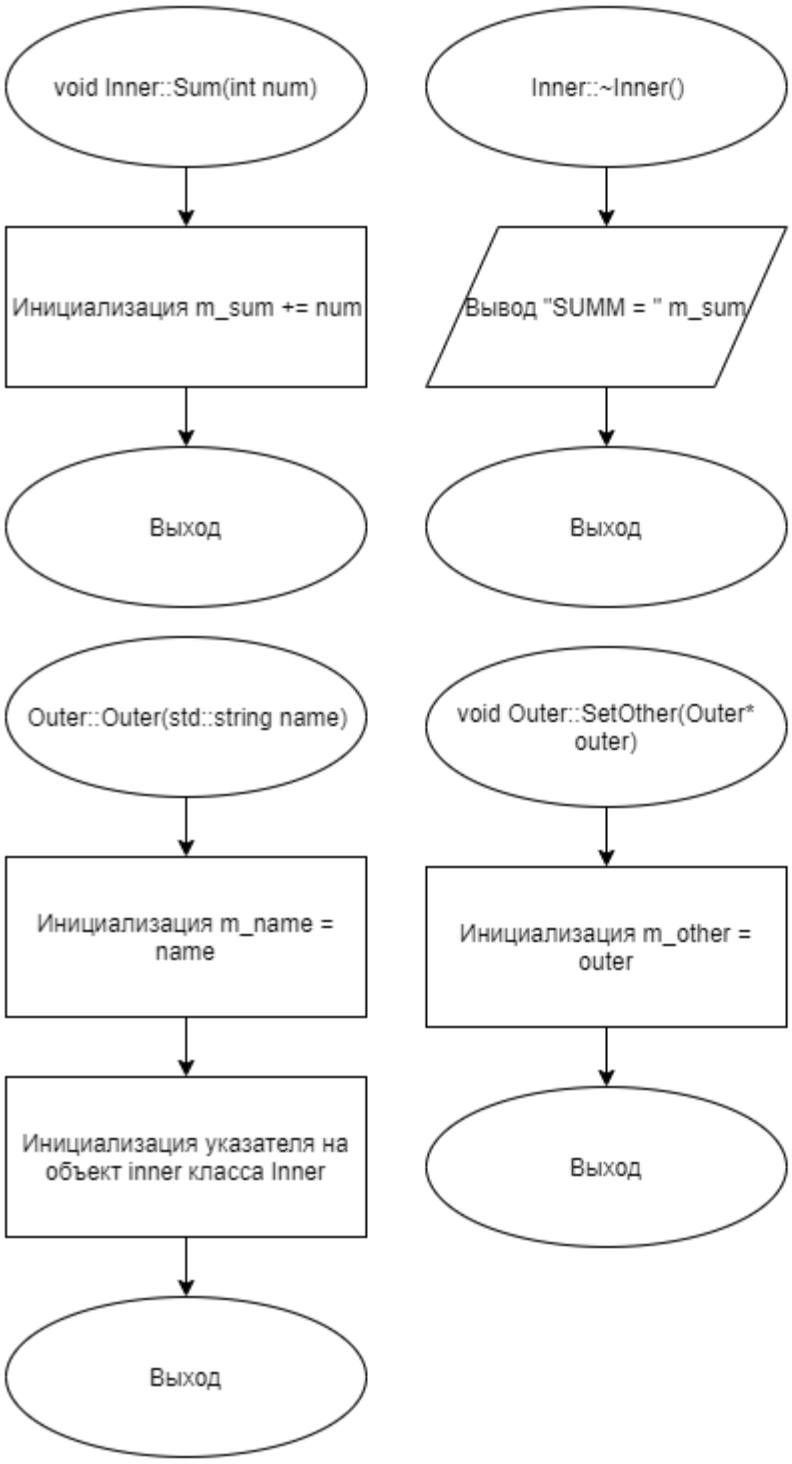

Рисунок 1 – Блок-схема алгоритма

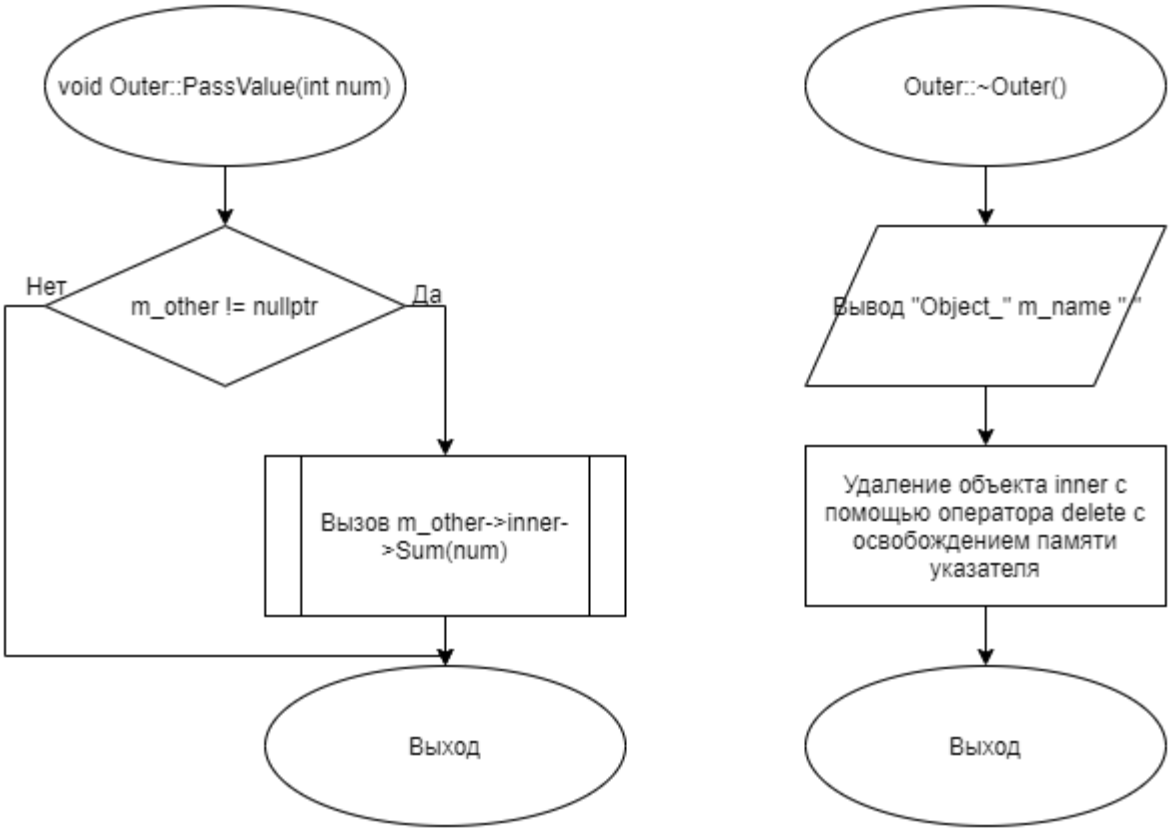

Рисунок 2 – Блок-схема алгоритма

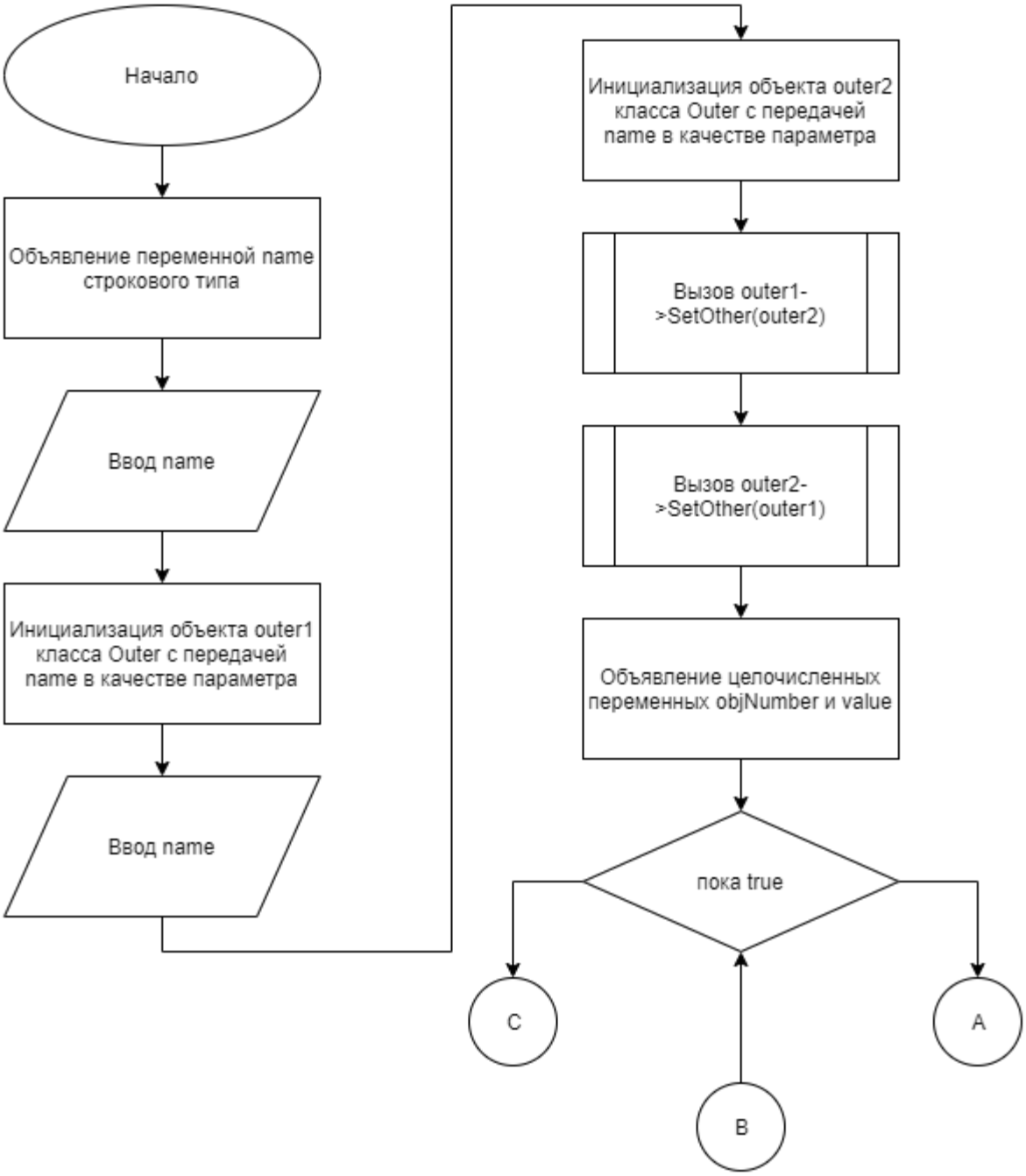

Рисунок 3 – Блок-схема алгоритма

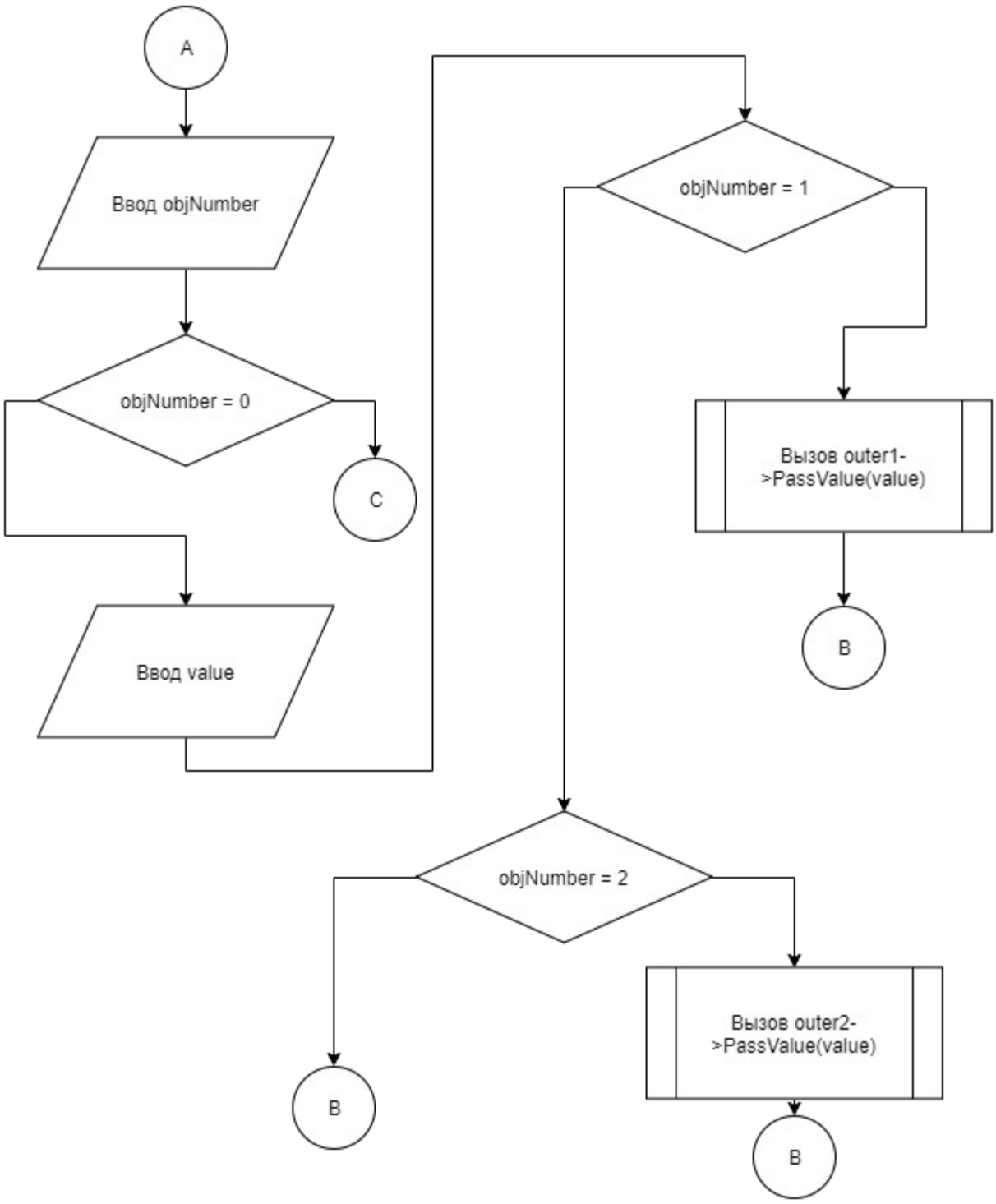

Рисунок 4 – Блок-схема алгоритма

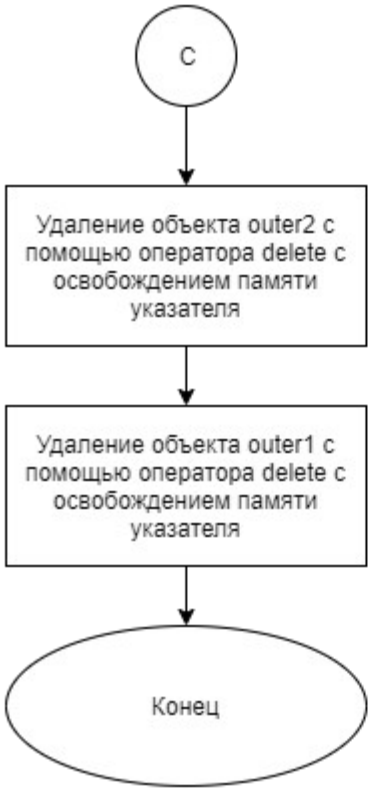

Рисунок 5 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл Inner.cpp

Листинг 1 – Inner.cpp

```
#include "Inner.h"

void Inner::Sum(int num)
{
   m_sum += num;
}

Inner::~Inner()
{
   std::cout << "SUMM = " << m_sum << std::endl;
}
```

5.2 Файл Inner.h

Листинг 2 – Inner.h

```
#ifndef __INNER__H
#define __INNER__H

#include <string>
#include <iostream>

class Inner
{
   int m_sum;
public:
   void Sum(int num);

   ~Inner();
};

#endif
```

5.3 Файл main.cpp

Листинг 3 – main.cpp

```
#include "Outer.h"

int main()
{
   std::string name;

   std::cin >> name;
   Outer* outer1 = new Outer(name);

   std::cin >> name;
   Outer* outer2 = new Outer(name);

   outer1->SetOther(outer2);
   outer2->SetOther(outer1);

   int objNumber, value;

   while (true)
   {
      std::cin >> objNumber;
      if (objNumber == 0) break;

      std::cin >> value;

      if (objNumber == 1)
      {
         outer1->PassValue(value);
      }
      else if (objNumber == 2)
      {
         outer2->PassValue(value);
      }
   }

   delete outer2;
   delete outer1;

   return 0;
}
```

5.4 Файл Outer.cpp

Листинг 4 – Outer.cpp

```
#include "Outer.h"
```

```
Outer::Outer(std::string name) : m_name(name)
{
   inner = new Inner();
}

void Outer::SetOther(Outer* outer)
{
   m_other = outer;
}

void Outer::PassValue(int num)
{
   if (m_other != nullptr)
   {
      m_other->inner->Sum(num);
   }
}

Outer::~Outer()
{
   std::cout << "Object_" << m_name << "    ";
   delete inner;
}
```

5.5 Файл Outer.h

Листинг 5 – Outer.h

```
#ifndef __OUTER__H
#define __OUTER__H

#include "Inner.h"

class Outer
{
   std::string m_name;

   Outer* m_other = nullptr;
   Inner* inner;
public:
   Outer(std::string name);

   void SetOther(Outer* outer);
   void PassValue(int num);

   ~Outer();
};
```

```
    #endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 9.

Таблица 9 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
ffff ssss 1 1 2 2 2 2 1 4 0	Object_ssss SUMM = 5 Object_ffff SUMM = 4	Object_ssss SUMM = 5 Object_ffff SUMM = 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-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).