Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Дан объект следующей конструкции:

В закрытом доступе имеется массив целого типа и поле его длины. Количество элементов массива четное и больше двух. Объект имеет функциональность:

- Конструктор по умолчанию, вначале работы выдает сообщение;
- Параметризированный конструктор, передается целочисленный параметр. Параметр должен иметь значение больше 2 и быть четным. Вначале работы выдает сообщение;
- Конструктор копии, обеспечивает создание копии объекта в новой области памяти. Вначале работы выдает сообщение;
- Метод деструктор, который в начале работы выдает сообщение;
- Метод который создает целочисленный массив в закрытой области, согласно ранее заданной размерности.
- Метод ввода данных для созданного массива;
- Метод 1, который суммирует значения очередной пары элементов и сумму присваивает первому элементу пары. Например, пусть массив состоит из элементов {1,2,3,4}. В результате суммирования пар получим массив {3,2,7,4};
- Метод 2, который умножает значения очередной пары элементов и результат присваивает первому элементу пары. Например, пусть массив состоит из элементов {1,2,3,4}. В результате умножения пар получим массив {2,2,12,4};
- Метод который, суммирует значения элементов массива и возвращает это значение;
- Метод последовательного вывода содержимого элементов массива,

которые разделены тремя пробелами.

Разработать функцию func, которая имеет один целочисленный параметр, содержащий размерность массива. В функции должен быть реализован алгоритм:

1. Создание локального объекта с использованием параметризированного конструктора.
2. Возврат созданного локального объекта.

В основной функции реализовать алгоритм:

1. Ввод размерности массива.
2. Если размерность массива некорректная, вывод сообщения и завершить работу алгоритма.
3. Вывод значения размерности массива.
4. Создание первого объекта.
5. Присвоение первому объекту результата работы функции func с аргументом, содержащим значение размерности массива.
6. Для первого объекта вызов метода создания массива.
7. Для первого объекта вызов метода ввода данных массива.
8. Для первого объекта вызов метода 2.
9. Инициализация второго объекта первым объектом.
10. Вызов метода 1 для второго объекта.
11. Вывод содержимого массива первого объекта.
12. Вывод суммы элементов массива первого объекта.
13. Вывод содержимого массива второго объекта.
14. Вывод суммы элементов массива второго объекта.

1.1 Описание входных данных

Первая строка:

```
«Целое число»
```

Вторая строка:

```
«Целое число» «Целое число» . . .
```

Пример:

```
4
3 5 1 2
```

1.2 Описание выходных данных

Если введенная размерность массива допустима, то в первой строке выводится это значение:

```
«Целое число»
```

Если введенная размерность массива не больше двух или нечетная, то в первой строке выводится некорректное значение и вопросительный знак:

```
«Целое число»?
```

Конструктор по умолчанию в начале работы с новой строки выдает сообщение:

```
Default constructor
```

Параметризированный конструктор в начале работы с новой строки выдает сообщение:

```
Constructor set
```

Конструктор копии в начале работы с новой строки выдает сообщение:

```
Copy constructor
```

Деструктор в начале работы с новой строки выдает сообщение:

```
Destructor
```

Метод последовательного вывода содержимого элементов массива, с новой строки выдает:

```
«Целое число»   «Целое число»   «Целое число»   . . .
```

Пример вывода:

```
4
Default constructor
Constructor set
Destructor
Copy constructor
15   5   2   2
24
20   5   4   2
31
Destructor
Destructor
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- функция function для создаёт локальный объект с помощью параметрзированного конструктор и его возврат;
- функция main для основного алгоритма функции.

Класс MyClass:

- функционал:
  - метод build() — создаёт целочисленный массив в закрытой области , согласно ранее заданной размерности;
  - метод Sumator() — суммирует значения очередной пары элементов и сумму присваивает первому элементу пары;
  - метод MySpace — умножает значения очередной пары элементов и результат присваивает первому элементу пары;
  - метод Print() — выводит содержимое массива , элементы разделены трему пробелами.

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 build() класса MyClass

Функционал: создаёт целочисленный массив в закрытой области , согласно ранее заданной размерности.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 1.

Таблица 1 – Алгоритм метода build() класса MyClass

№	Предикат	Действия	№ перехода
1		Создание целочисленного массива masive размерности size	∅

3.2 Алгоритм метода Sumator() класса MyClass

Функционал: суммирует значения очередной пары элементов и сумму присваивает первому элементу пары.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода Sumator() класса MyClass

№	Предикат	Действия	№ перехода
1		Инициализация целочисленной переменной i = 0	2

№	Предикат	Действия	№ перехода
2	i < size	masive[i]=masove[i]+masive[i+1]	3
			∅
3		i+=2	2

3.3 Алгоритм метода MySpace класса MyClass

Функционал: умножает значения очередной пары элементов и результат присваивает первому элементу пары.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода MySpace класса MyClass

№	Предикат	Действия	№ перехода
1		инициализация целочисленной переменной i = 0	2
2	i < size	masive[i]=masive[i]*masive[i+1]	3
		i+=2	∅
3			2

3.4 Алгоритм метода Print() класса MyClass

Функционал: выводит содержимое массива , элементы разделены трему пробелами.

Параметры: нет.

Возвращаемое значение: void.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода Print() класса MyClass

№	Предикат	Действия	№ перехода
1		инициализация целочисленной переменной i = 0	2
2	i < size	вывод значения элемента masive[i]	3
			∅
3	i ! = size-1	вывод трёх пробелов	4
			4
4		i+=1	2

3.5 Алгоритм функции function

Функционал: Создаёт локальный объект с помощью параметризированного конструктора и его возврат.

Параметры: int back .

Возвращаемое значение: MyClass.

Алгоритм функции представлен в таблице 5.

Таблица 5 – Алгоритм функции function

№	Предикат	Действия	№ перехода
1		Создание локального объекта loacle с параметром back	2
2		возврат объекта locale	∅

3.6 Алгоритм функции main

Функционал: основной алгоритм программы.

Параметры: нет.

Возвращаемое значение: int.

Алгоритм функции представлен в таблице 6.

Таблица 6 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление целочисленной переменной back	2
2		ввод значения переменной back	3
3	back <= 2 or back%2!=0	вывод значения переменной back "?"	∅
			4
4		вывод значения переменной back	5
5		вывод переноса на новую строку	6
6		создание объекта obj1 класса MyClass	7
7		Присвоение объекту obj1 значение работы функции function(back)	8
8		вывод переноса на новую строку	9
9		вызов метода build объекта obj1	10
10		вызов метода Ellement объекта obj1	11
11		вызов метода MySpace объекта obj1	12
12		инициализация объекта obj2 класса MyClass объектом obj1	13
13		вызов метода Sumator объекта obj2	14
14		вызов метода Print объекта obj1	15
15		вывод результата работы метода Sum объекта obj1	16
16		вывод результата работы метода Sum объекта obj2	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

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

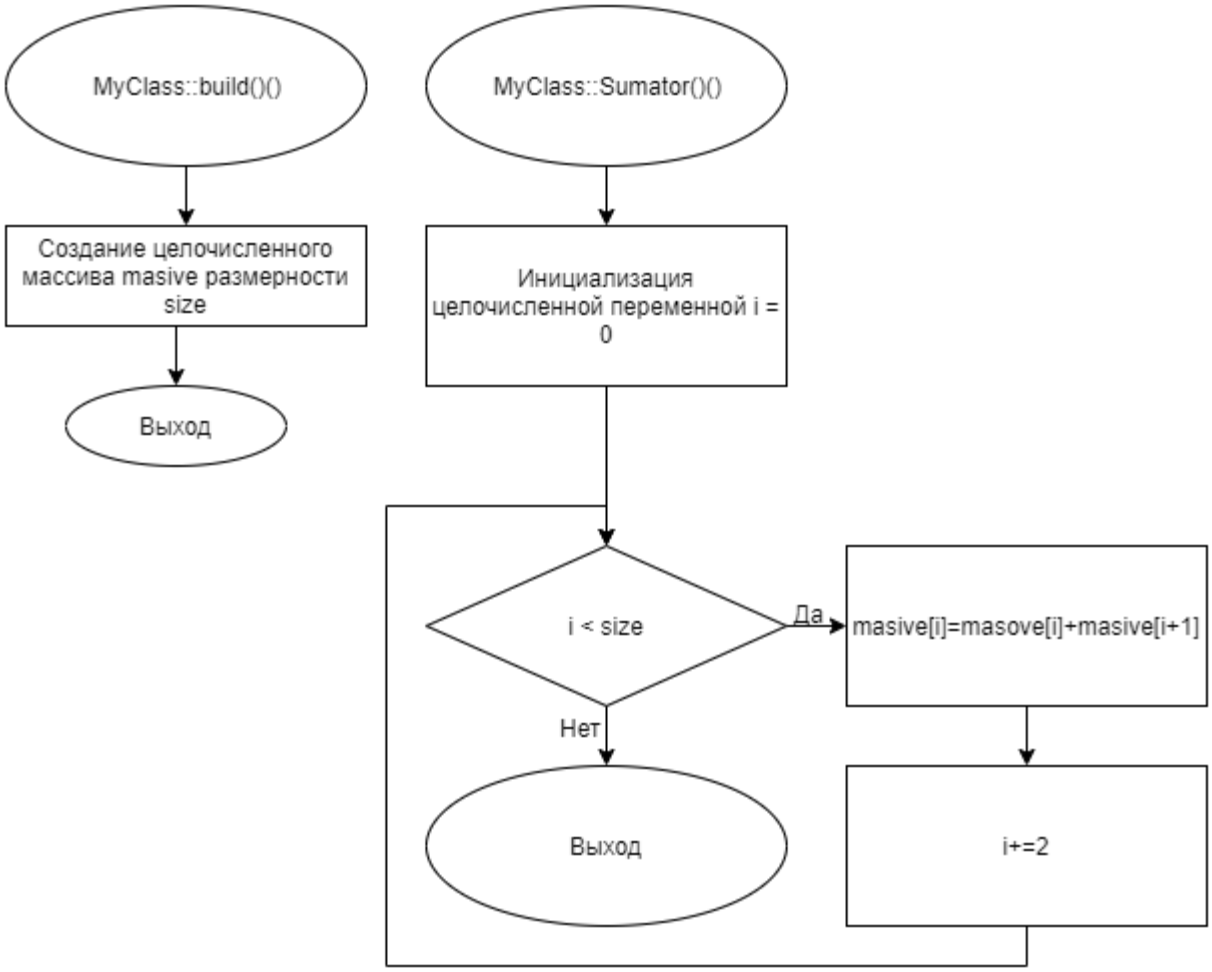

Рисунок 1 – Блок-схема алгоритма

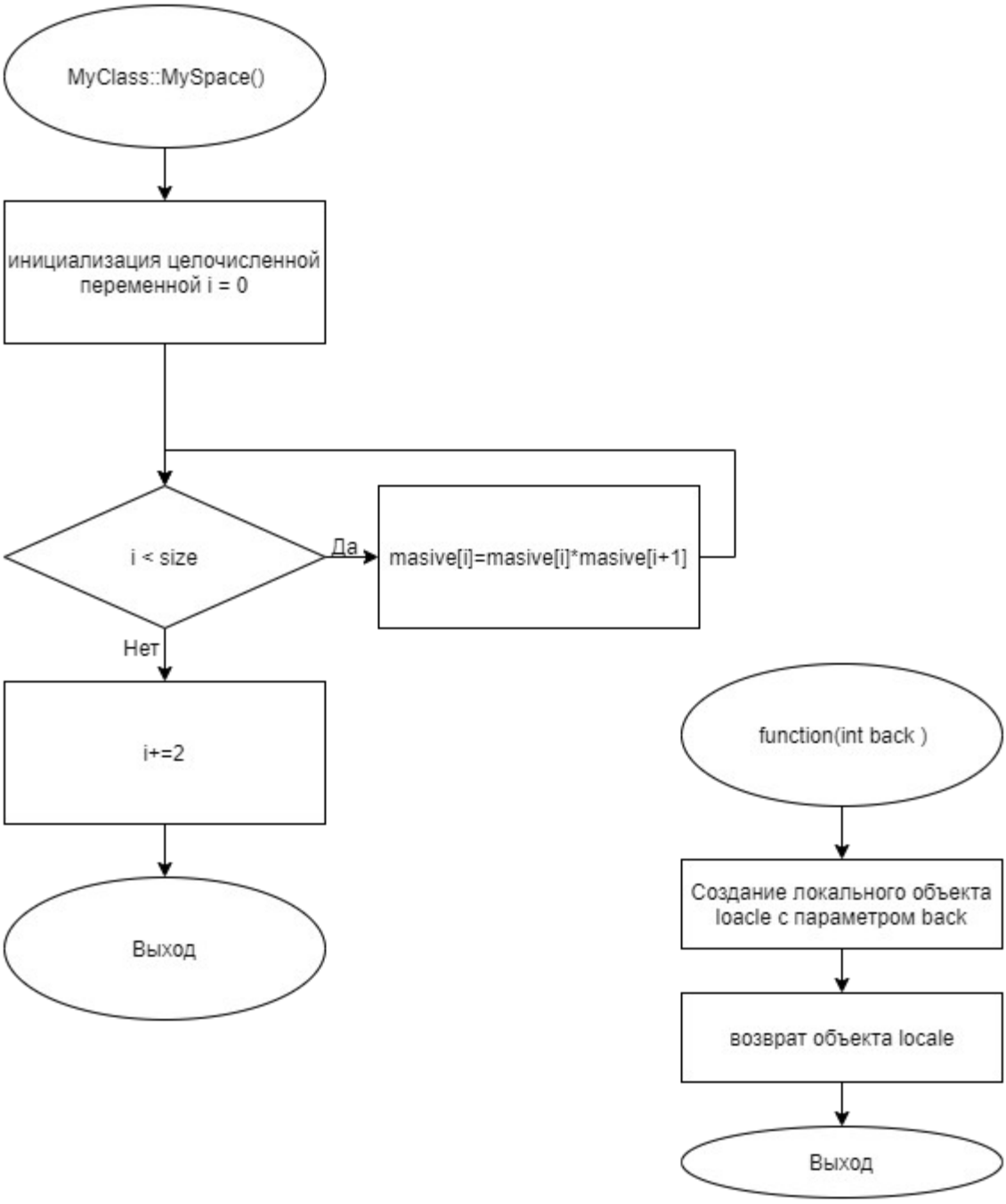

Рисунок 2 – Блок-схема алгоритма

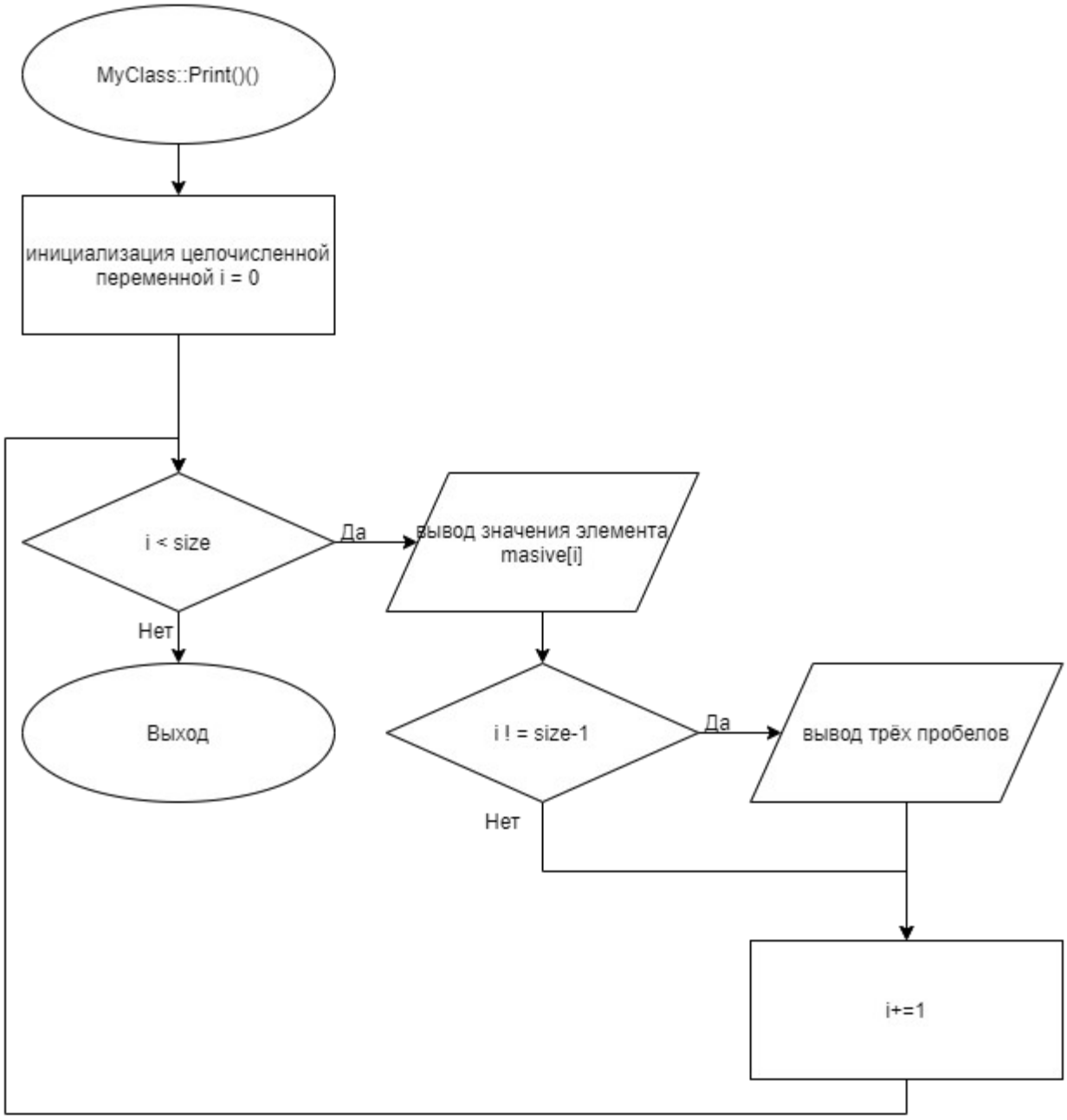

Рисунок 3 – Блок-схема алгоритма

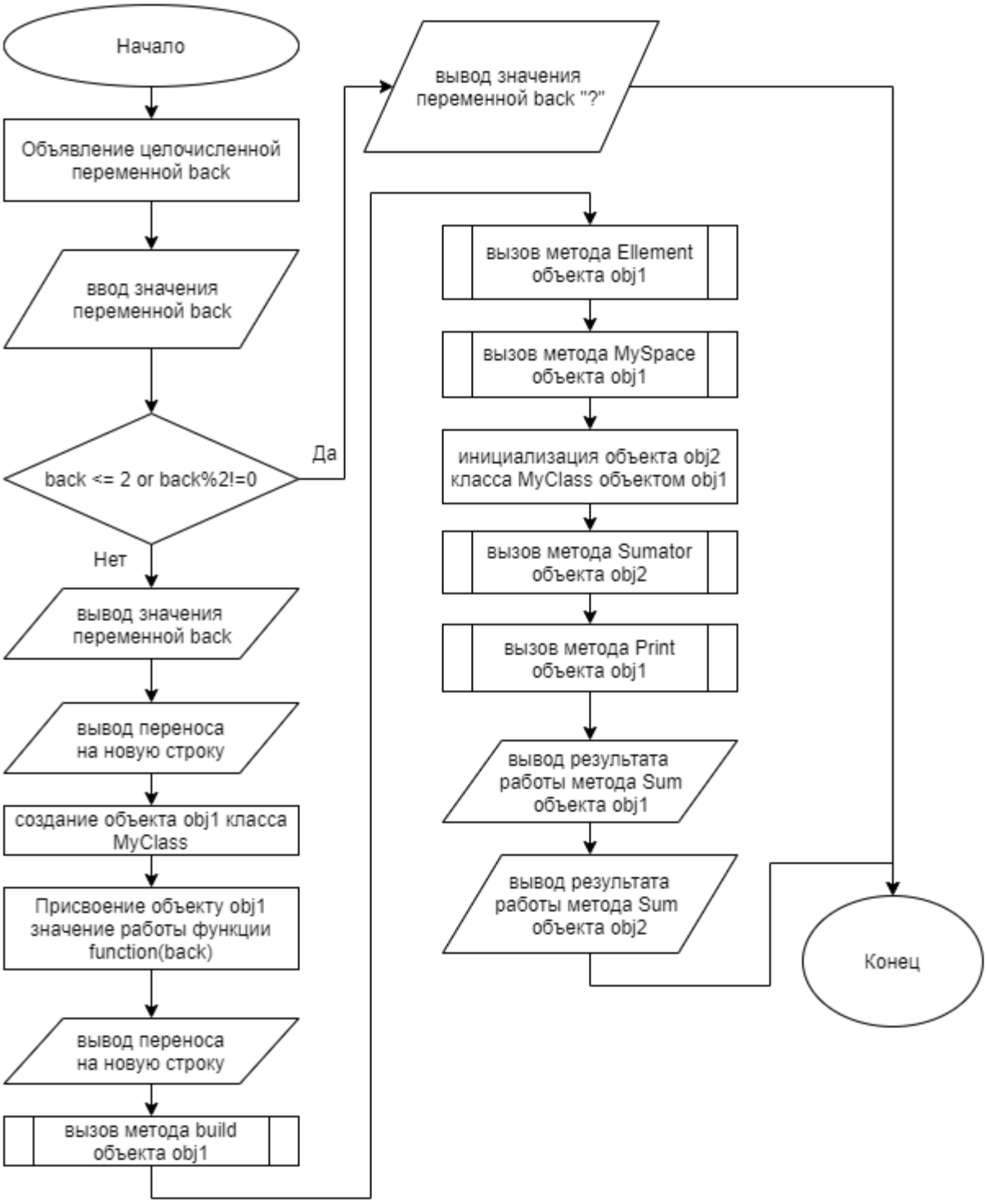

Рисунок 4 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

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

MyClass function(int back){
   MyClass locale(back);
   return locale;
}

int main(){
   int back;
   cin >> back;
   if (back <= 2 || back%2!=0){
      cout <<back << "?";
      return 0;
   }
   cout << back;
   cout <<endl;
   MyClass obj1;
   obj1=function(back);
   cout << endl;
   obj1.build();
   obj1.Ellement();
   obj1.MySpace();
   MyClass obj2=obj1;
   obj2.Sumator();
   obj1.Print();
   cout << endl <<obj1.Sum()<<endl;
   obj2.Print();
   cout << endl<< obj2.Sum();
}
```

5.2 Файл MyClass.cpp

Листинг 2 – MyClass.cpp

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

MyClass::MyClass(){
   cout <<"Default constructor" <<endl;
   masive = nullptr;
}
MyClass::MyClass(int size){
   cout <<"Constructor set";
   masive = new int[size];
   this ->size = size;
}
MyClass::MyClass(const MyClass& object){
   cout << "Copy constructor" << endl;
   size = object.size;
   masive = new int [size];
   for (int i = 0; i<size ; i++){
      masive[i]=object.masive[i];
   }
}
MyClass::~MyClass(){
   cout << endl<< "Destructor";
   if (masive != nullptr){
      delete[] masive;
   }
}
void MyClass::build(){
   masive = new int[size];
}
void MyClass::Ellement(){
   int temp ;
   for (int i =0; i<size; i++){
      cin >> temp;
      masive[i]=temp;
   }
}
void MyClass::Sumator(){
   for (int i =0; i<size; i+=2){
      masive[i]=masive[i] + masive[i+1];
   }
}
void MyClass::MySpace(){
   for (int i =0; i<size; i+=2){
      masive[i]=masive[i] * masive[i+1];
   }
}
int MyClass::Sum(){
   int count=0;
   for (int i =0; i<size; i++){
```

```
            count += masive[i];
         }
         return count;
      }
      void MyClass::Print(){
         for (int i =0; i<size; i++){
            cout<< masive[i];
            if (i!=size-1){
               cout<<"   ";
            }
         }
      }
```

5.3 Файл MyClass.h

Листинг 3 – MyClass.h

```
      #ifndef __MYCLASS__H
      #define __MYCLASS__H

      using namespace std;

      class MyClass{
      private:
         int size;
         int *masive;
      public:
         MyClass();
         MyClass(int size);
         MyClass(const MyClass& object);
         ~MyClass();
         void build();
         void Ellement();
         void Sumator();
         void MySpace();
         int Sum();
         void Print();
      };

      #endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 7.

Таблица 7 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
4 3 5 1 2	4 Default constructor Constructor set Destructor Copy constructor 15 5 2 2 24 20 5 4 2 31 Destructor Destructor	4 Default constructor Constructor set Destructor Copy constructor 15 5 2 2 24 20 5 4 2 31 Destructor Destructor
2	2?	2?
1	1?	1?
3	3?	3?
7	7?	7?
8 1 2 3 4 5 6 7 8	8 Default constructor Constructor set Destructor Copy constructor 2 2 12 4 30 6 56 8 120 4 2 16 4 36 6 64 8 140 Destructor Destructor	8 Default constructor Constructor set Destructor Copy constructor 2 2 12 4 30 6 56 8 120 4 2 16 4 36 6 64 8 140 Destructor Destructor

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).