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, в начале работы выдает сообщение;
- параметризированный конструктор, передается целочисленный параметр. Параметр должен иметь значение больше 2 и быть четным. В начале работы выдает сообщение;
- конструктор копии, обеспечивает создание копии объекта в новой области памяти. В начале работы выдает сообщение;
- деструктор, который в начале работы выдает сообщение;
- метод, который создает целочисленный массив в закрытой области, согласно ранее заданной размерности;
- метод ввода значений элементов 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 двумя пробелами.

Назовём класс описания данного объекта cl_obj.

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

1. Инициализация указателя на объект класса cl_obj адресом объекта, созданного с использованием параметризированного конструктора.
2. С использованием указателя на объект класса cl_obj вызов метода создания массива.
3. С использованием указателя на объект класса cl_obj вызов метода ввода значений элементов массива.
4. С использованием указателя на объект класса cl_obj вызов метода 2.
5. Возврат указателя на объект класса cl_obj.

В основной функции реализовать алгоритм:

6. Ввод размерности массива.
7. Если размерность массива некорректная, вывести сообщение и завершить работу алгоритма.
8. Вывод значения размерности массива.
9. Объявить первый указатель на объект класса cl_obj.
10. Присвоение первому указателю результата работы функции func с аргументом, содержащим значение размерности массива.
11. С использованием первого указателя вызов метода 1.
12. Инициализация второго указателя на объект класса cl_obj адресом объекта, созданного с использованием конструктора копии с аргументом первого объекта.
13. С использованием второго указателя вызов метода 2.
14. Вывод содержимого массива первого объекта.
15. Вывод суммы элементов массива первого объекта.

16. Вывод содержимого массива второго объекта.
17. Вывод суммы элементов массива второго объекта.
18. Второму объекту присвоить первый объект.
19. С использованием первого указателя вызов метода 1.
20. Вывод содержимого массива второго объекта.
21. 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
Constructor set
Copy constructor
20  5  4  2
31
100  5  8  2
115
25  5  6  2
38
Destructor
Destructor
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используются:

Объекты ввода/вывода cin/cout

Объекты new/delete

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 Mas_create класса MyClass

Функционал: Создание динамического массива.

Параметры: Отсутствуют.

Возвращаемое значение: Целый тип.

Алгоритм метода представлен в таблице 1.

Таблица 1 – Алгоритм метода Mas_create класса MyClass

№	Предикат	Действия	№ перехода
1		mas = new int[len]	∅

3.2 Алгоритм функции main

Функционал: Основная функция.

Параметры: Отсутствуют.

Возвращаемое значение: Целый тип.

Алгоритм функции представлен в таблице 2.

Таблица 2 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Инициализация l	2
2	l > 2) && (l % 2 == 0)		3
		Вывод l"?"	∅

№	Предикат	Действия	№ перехода
3		Создание указателя на объект класса obj_a	4
4		Присвоение obj_a значений возравщаймых функцией func(l)	5
5		Вызов метода Met_a() для obj_a	6
6		Создание указателя на объект obj_b типа MyClass с параметрам obj_a	7
7		Вызов метода Met_b() для obj_b	8
8		Вызов метода Print() для obj_a	9
9		Вывод результатов метода Summ() для объекта obj_a	10
10		Вызов метода Print() для obj_b	11
11		Вывод результатов метода Summ() для объекта obj_b	12
12		*obj_b = *obj_a	13
13		Вызов метода Met_a() для obj_a	14
14		Вызов метода Print() для obj_b	15
15		Вывод результатов метода Summ() для объекта obj_b	16
16		Отчистка памяти выделенной под указатели	∅

3.3 Алгоритм функции func

Функционал: Функция из постановки задачи.

Параметры: int l.

Возвращаемое значение: Указатель на MyClass.

Алгоритм функции представлен в таблице 3.

Таблица 3 – Алгоритм функции func

№	Предикат	Действия	№ перехода
1		Создание указателя на объект класса obj_func	2
2		Вызов метода Mas_create()	3
3		Вызов метода Input()	4
4		Вызов метода Met_b()	5
5		Возвращение obj_func	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

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

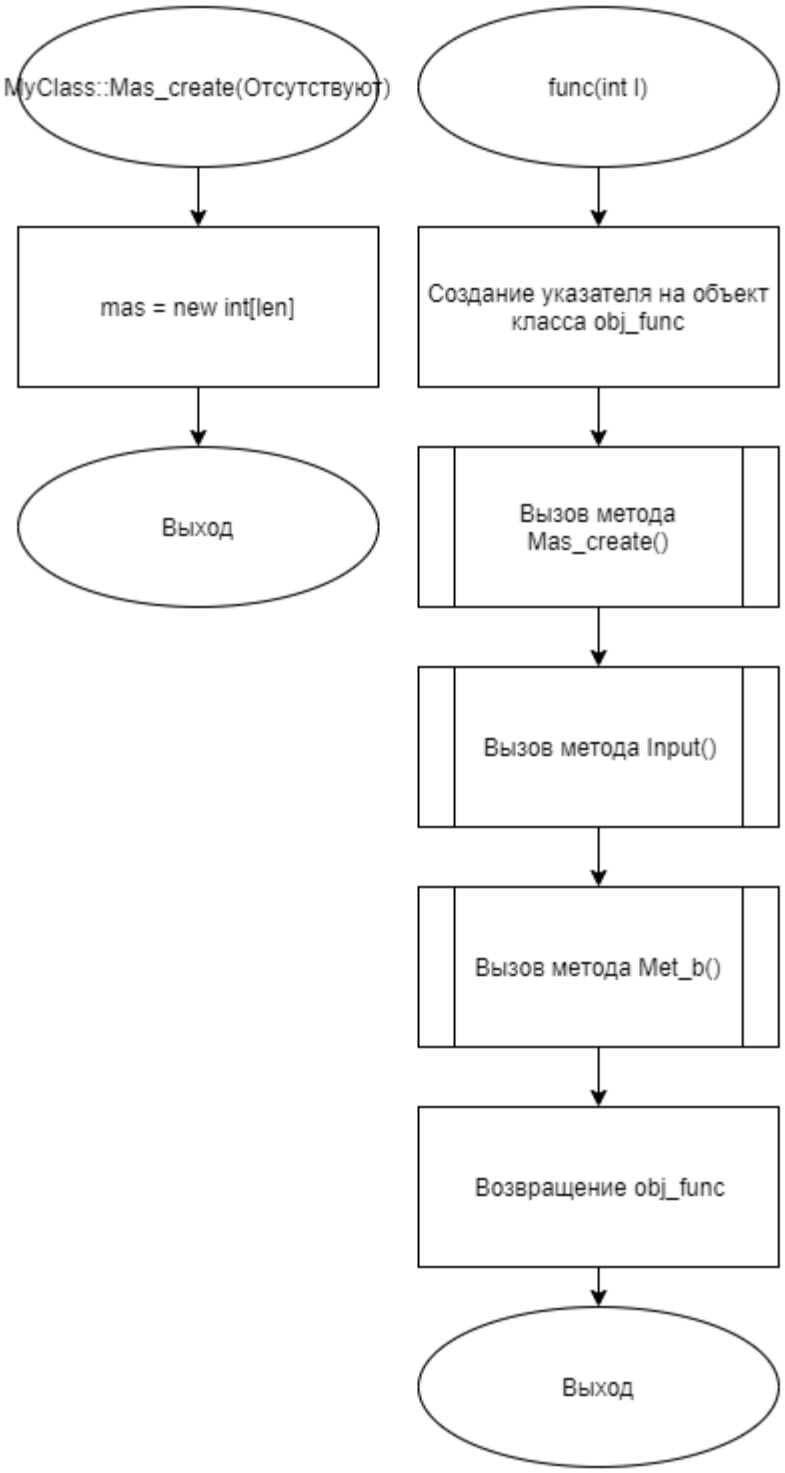

Рисунок 1 – Блок-схема алгоритма

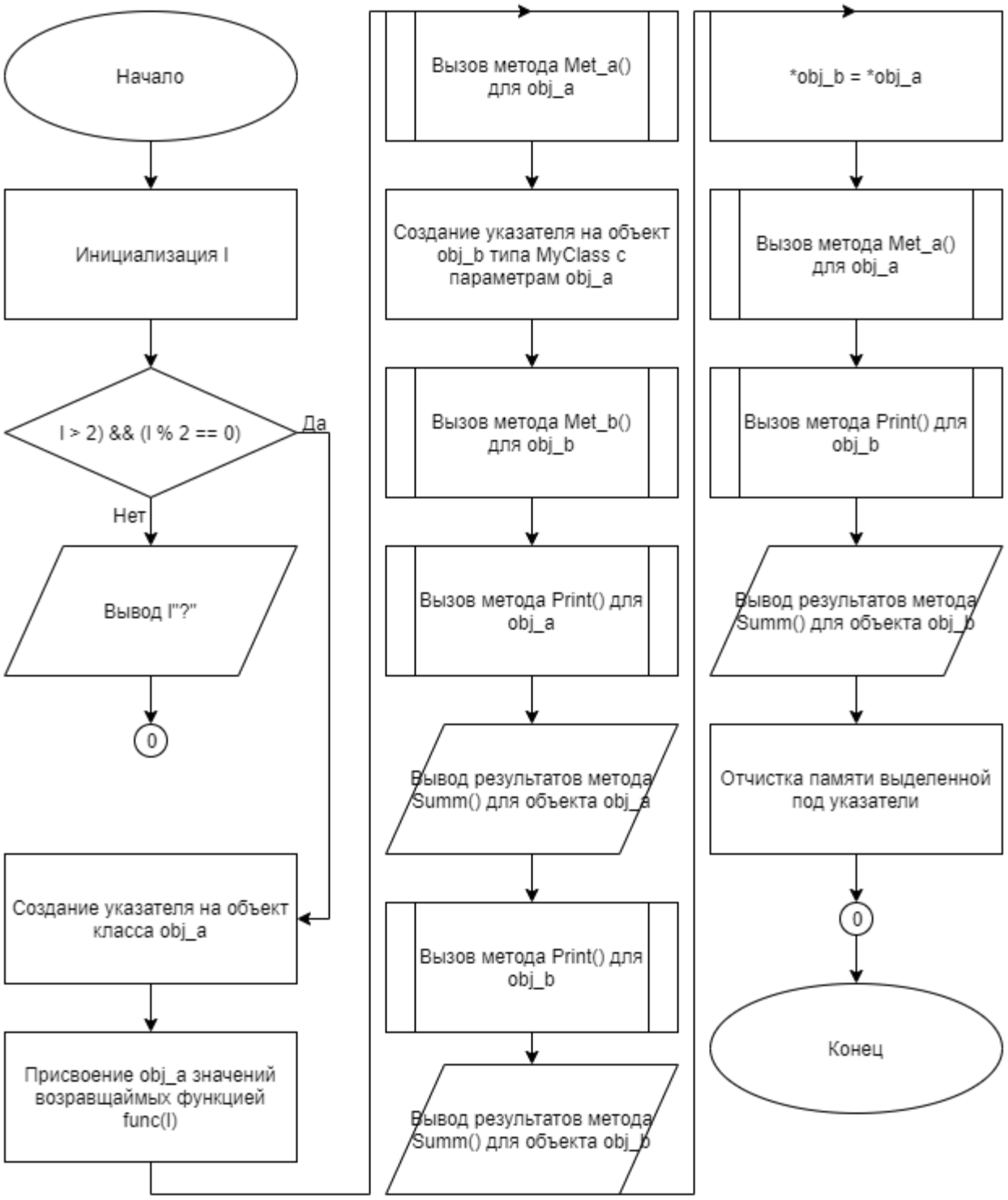

Рисунок 2 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

```
#include "MyClass.h"

MyClass* func(int l)
{
	MyClass* obj_func = new MyClass(l);
	obj_func->Mas_create();
	obj_func->Input();
	obj_func->Met_b();
	return obj_func;
}

int main()
{
	int l;
	cin >> l;
	cout << l;
	if ((l > 2) && (l % 2 == 0))
	{
		MyClass* obj_a;
		obj_a = func(l);
		obj_a->Met_a();
		MyClass* obj_b = new MyClass(obj_a);
		obj_b->Met_b();
		obj_a->Print();
		cout << endl << obj_a->Summ();
		obj_b->Print();
		cout << endl << obj_b->Summ();
		*obj_b = *obj_a;
		obj_a->Met_a();
		obj_b->Print();
		cout << endl << obj_b->Summ();
		delete obj_a;
		delete obj_b;
	}
	else
	{
		cout << "?";
	}
}
```

5.2 Файл MyClass.cpp

Листинг 2 – MyClass.cpp

```
#include "MyClass.h"
int dest = 0;

MyClass::MyClass()
{
	cout << "\nDefault constructor";
}

MyClass::MyClass(int l)
{
	cout << "\nConstructor set";
	len = l;
	mas = new int[len];
}

MyClass::~MyClass()
{
	cout << "\nDestructor";
	if (mas != nullptr && dest == 3)
	{
		delete []mas;
	}
}

MyClass::MyClass(MyClass* obj)
{
	cout << "\nCopy constructor";
	len = obj->len;
	mas = new int[len];
	for (int i = 0; i < len; i++)
	{
		mas[i] = obj->mas[i];
	}
}

void MyClass::Input()
{
	for (int i = 0; i < len; i++)
	{
		cin >> mas[i];
	}
}

void MyClass::Met_a()
{
	for (int i = 0; i+1 < len; i = i + 2)
	{
		mas[i] = mas[i] + mas[i+1];
	}
}
```

```
void MyClass::Met_b()
{
	for (int i = 0; i+1 < len; i = i + 2)
	{
		mas[i] = mas[i] * mas[i+1];
	}
}

int MyClass::Summ()
{
	int summ = 0;
	for (int i = 0; i < len; i++)
	{
		summ = summ + mas[i];
	}
	return summ;
}

void MyClass::Print()
{
	cout << endl;
	for (int i = 0; i < len; i++)
	{
		cout << mas[i];
		if (i != len-1)
		{
			cout << "  ";
		}
	}
}

void MyClass::Mas_create()
{
	mas = new int[len];
}
```

5.3 Файл MyClass.h

Листинг 3 – MyClass.h

```
#ifndef __MYCLASS_H__
#define __MYCLASS_H__
#include <iostream>
using namespace std;
class MyClass
{
public:
	MyClass();
	MyClass(int l);
	MyClass(MyClass* obj);
	~MyClass();
	void Input();
	void Met_a();
```

```
      void Met_b();
      int Summ();
      void Print();
      void Mas_create();
private:
      int *mas, len = 0;
};
#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 4.

Таблица 4 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
4 3 5 1 2	4 Constructor set Copy constructor 20 5 4 2 31 100 5 8 2 115 25 5 6 2 38 Destructor Destructor	4 Constructor set Copy constructor 20 5 4 2 31 100 5 8 2 115 25 5 6 2 38 Destructor Destructor
3 3 2 5 4	3?	3?

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).