Здесь будет титульник, листай ниже

**СОДЕРЖАНИЕ**

[1 ПОСТАНОВКА ЗАДАЧИ 5](#__RefHeading___Toc2645_3274402494)

[1.1 Описание входных данных 7](#__RefHeading___Toc2647_3274402494)

[1.2 Описание выходных данных 7](#__RefHeading___Toc2649_3274402494)

[2 МЕТОД РЕШЕНИЯ 9](#__RefHeading___Toc2651_3274402494)

[3 ОПИСАНИЕ АЛГОРИТМОВ 10](#__RefHeading___Toc2653_3274402494)

[3.1 Алгоритм функции main 10](#__RefHeading___Toc2655_3274402494)

[3.2 Алгоритм функции func 11](#__RefHeading___Toc2657_3274402494)

[3.3 Алгоритм метода GetArr класса Object 12](#__RefHeading___Toc2659_3274402494)

[3.4 Алгоритм метода SetArr класса Object 12](#__RefHeading___Toc2661_3274402494)

[4 БЛОК-СХЕМЫ АЛГОРИТМОВ 13](#__RefHeading___Toc2663_3274402494)

[5 КОД ПРОГРАММЫ 15](#__RefHeading___Toc2665_3274402494)

[5.1 Файл main.cpp 15](#__RefHeading___Toc2667_3274402494)

[5.2 Файл Object.cpp 16](#__RefHeading___Toc2669_3274402494)

[5.3 Файл Object.h 17](#__RefHeading___Toc2671_3274402494)

[6 ТЕСТИРОВАНИЕ 19](#__RefHeading___Toc2673_3274402494)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#__RefHeading___Toc2675_3274402494)

# 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 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;
* метод, который возвращает значение указателя на массив из закрытой области;
* метод, который присваивает значение указателя массива из закрытой области.

Назовём класс описания данного объекта cl\_obj (для примера, у вас он может называться иначе).

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

* Инициализация указателя на объект класса cl\_obj адресом объекта, созданного с использованием параметризированного конструктора.
* С использованием указателя на объект класса cl\_obj вызов метода создания массива.
* С использованием указателя на объект класса cl\_obj вызов метода ввода значений элементов массива.
* С использованием указателя на объект класса cl\_obj вызов метода 2.
* Возврат указателя на объект класса cl\_obj.

В основной функции реализовать алгоритм:

* Ввод размерности массива.
* Если размерность массива некорректная, вывод сообщения и завершить работу алгоритма.
* Вывод значения размерности массива.
* Объявить первый указатель на объект класса cl\_obj.
* Присвоение первому указателю результата работы функции func с аргументом, содержащим значение размерности массива.
* С использованием первого указателя вызов метода 1.
* Инициализация второго указателя на объект класса cl\_obj адресом объекта, созданного с использованием конструктора копии с аргументом первого объекта.
* С использованием второго указателя вызов метода 2.
* Вывод содержимого массива первого объекта.
* Вывод суммы элементов массива первого объекта.
* Вывод содержимого массива второго объекта.
* Вывод суммы элементов массива второго объекта.
* Второму объекту присвоить первый объект.
* С использованием первого указателя вызов метода 1.
* Вывод содержимого массива второго объекта.
* Вывод суммы элементов массива второго объекта.
* Удалит первый объект.
* Удалить второй объект.

Добавить в этот алгоритм пункты, которые обеспечат корректное завершение работы программы.

## 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

100 5 8 2

115

Destructor

Destructor

# 2 МЕТОД РЕШЕНИЯ

Для решения задачи используется то же, что и в предыдущей задаче 3\_1\_4, однако класс Object, функции main и func переработаны под условия задачи.

Класс Object:

* Свойства/поля:
  + Поля arr и arrSize класса Object остались неизменными.
* Методы:
  + Те же методы, что и в предыдущей задаче 3\_1\_4.
  + Метод GetArr
    - Функционал - Используется для возвращения значения указателя на массив из закрытой области.
  + Метод SetArr
    - Функционал - Используется для присваивания значения указателя массиву из закрытой области.

# 3 ОПИСАНИЕ АЛГОРИТМОВ

Согласно этапам разработки, после определения необходимого инструментария в разделе «Метод», составляются подробные описания алгоритмов для методов классов и функций.

## Алгоритм функции main

Функционал: Основной алгоритм программы.

Параметры: Отсутствуют.

Возвращаемое значение: int - индикатор корректности завершения программы.

Алгоритм функции представлен в таблице 1.

Таблица 1 – Алгоритм функции main

| № | Предикат | Действия | № перехода |
| --- | --- | --- | --- |
| 1 |  | Начало работы алгоритма из предыдущей задачи 3\_1\_4 | 2 |
| 2 |  | Создание указателя на объект obj1 класса Object | 3 |
| 3 |  | Присваивание объекту obj1 результата функции func с аргументом size | 4 |
| 4 |  | Вызов метода Method1 объекта obj1 | 5 |
| 5 |  | Создание указателя на объект obj2 класса Object и выделение под него памяти с помощью оператора new | 6 |
| 6 |  | Вызов метода Method2 объекта obj2 | 7 |
| 7 |  | Вызов метода OutputArray объекта obj1 | 8 |
| 8 |  | Вывод результата вызова метода Sum объекта obj1 на экран | 9 |
| 9 |  | Вызов метода OutputArray объекта obj2 | 10 |
| 10 |  | Вывод результата вызова метода Sum объекта obj2 на экран | 11 |
| 11 |  | Присваивание объекту obj2 значение объекта obj1 | 12 |
| 12 |  | Вызов метода Method2 объекта obj1 | 13 |
| 13 |  | Вызов метода OutputArray объекта obj2 | 14 |
| 14 |  | Вывод результата вызова метода Sum объекта obj2 на экран | 15 |
| 15 |  | Удаление объекта obj1 с помощью оператора delete | 16 |
| 16 |  | Удаление объекта obj1 с помощью оператора delete | 17 |
| 17 |  | Окончание работы алгоритма из предыдущей задачи 3\_1\_4 | ∅ |

## Алгоритм функции func

Функционал: Создание локального объекта с использованием параметризированного конструктора.

Параметры: int size - параметр для задания размера массива создаваемого объекта.

Возвращаемое значение: Object\* - возврат указателя на созданный объект класса Object.

Алгоритм функции представлен в таблице 2.

Таблица 2 – Алгоритм функции func

| № | Предикат | Действия | № перехода |
| --- | --- | --- | --- |
| 1 |  | Создание указателя на объект obj класса Object и выделение под него памяти с помощью оператора new | 2 |
| 2 |  | Вызов метода CreateArray объекта obj | 3 |
| 3 |  | Вызов метода FillArray объекта obj | 4 |
| 4 |  | Вызов метода Method2 объекта obj | 5 |
| 5 |  | Возврат указателя на объект obj | ∅ |

## Алгоритм метода GetArr класса Object

Функционал: Используется для возвращения значения указателя на массив из закрытой области.

Параметры: Отсутствуют.

Возвращаемое значение: int\* - значение указателя на массив.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода GetArr класса Object

| № | Предикат | Действия | № перехода |
| --- | --- | --- | --- |
| 1 |  | Возврат значения поля arr текущего объекта | ∅ |

## Алгоритм метода SetArr класса Object

Функционал: Используется для присваивания значения указателя массиву из закрытой области.

Параметры: int\* arr - указатель на новое значение для поля arr текущего объекта.

Возвращаемое значение: Отсутствует.

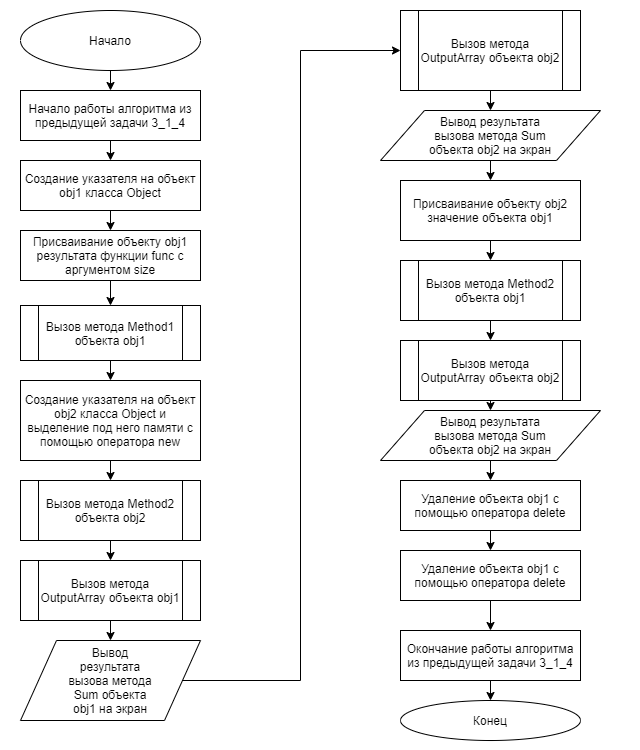
Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода SetArr класса Object

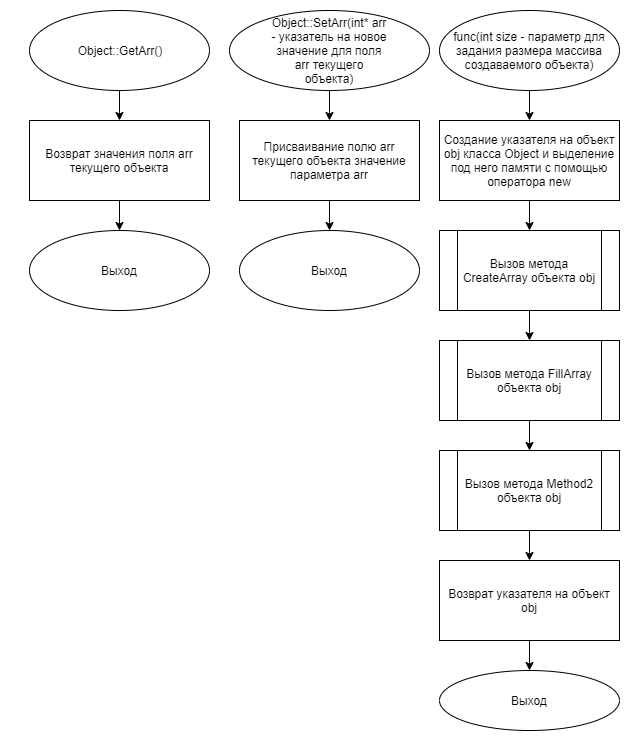
| № | Предикат | Действия | № перехода |
| --- | --- | --- | --- |
| 1 |  | Присваивание полю arr текущего объекта значение параметра arr | ∅ |

# 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.

## **Файл** **main.cpp**

Листинг 1 – main.cpp

|  |
| --- |
| #include <iostream>  #include "Object.h"  using namespace std;  Object\* func(int size)  {  Object\* obj = new Object(size);  obj->CreateArray();  obj->FillArray();  obj->Method2();  return obj;  }  int main()  {  int size;  cin >> size;  if(size > 2 && size%2 == 0)  {  cout << size << endl;  Object\* obj1;  obj1 = func(size);  obj1->Method1();  Object\* obj2 = new Object(\*obj1);  obj2->Method2();  obj1->OutputArray();  cout << obj1->Sum() << endl;  obj2->OutputArray();  cout << obj2->Sum() << endl;  \*obj2 = \*obj1;  obj1->Method2();  obj2->OutputArray();  cout << obj2->Sum();  delete obj1;  delete obj2;  }  else  {  cout << size << "?";  }  return(0);  } |

## **Файл** **Object.cpp**

Листинг 2 – Object.cpp

|  |
| --- |
| #include "Object.h"  Object::Object()  {  cout << "Default constructor" << endl;  }  Object::Object(int arrSize)  {  cout << "Constructor set";  this->arrSize = arrSize;  }  Object::Object(const Object &obj)  {  cout << "\nCopy constructor" << endl;  arrSize = obj.arrSize;  arr = new int[arrSize];  for(int i = 0; i < arrSize; i++)  {  arr[i] = obj.arr[i];  }  }  Object::~Object()  {  cout << "\nDestructor";  }  void Object::FillArray()  {  for(int i = 0; i < arrSize; i++)  {  cin >> arr[i];  }  }  void Object::CreateArray()  {  arr = new int[arrSize];  }  void Object::OutputArray()  {  cout << arr[0];  for(int i = 1; i < arrSize; i++)  {  cout << " " << arr[i];  }  cout << endl;  }  int\* Object::GetArr()  {  return arr;  }  void Object::SetArr(int\* arr)  {  this->arr = arr;  }  int Object::Sum()  {  int sum = 0;  for(int i = 0; i < arrSize; i++)  {  sum += arr[i];  }  return sum;  }  void Object::Method1()  {  for(int i = 0; i < arrSize; i+=2)  {  arr[i] += arr[i+1];  }  }  void Object::Method2()  {  for(int i = 0; i < arrSize; i+=2)  {  arr[i] \*= arr[i+1];  }  } |

## **Файл** **Object.h**

Листинг 3 – Object.h

|  |
| --- |
| #ifndef OBJECT\_H  #define OBJECT\_H  #include <iostream>  using namespace std;  class Object  {  int\* arr;  int arrSize;  public:  Object();  Object(int arrSize);  Object(const Object &obj);  ~Object();  void FillArray();  void CreateArray();  void OutputArray();  int\* GetArr();  void SetArr(int\*);  int Sum();  void Method1();  void Method2();  };  #endif |

# 6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 5.

Таблица 5 – Результат тестирования программы

| Входные данные | Ожидаемые выходные данные | Фактические выходные данные |
| --- | --- | --- |
| 4  1 2 3 4 | 4  Constructor set  Copy constructor  4 2 16 4  26  8 2 64 4  78  8 2 64 4  78  Destructor  Destructor | 4  Constructor set  Copy constructor  4 2 16 4  26  8 2 64 4  78  8 2 64 4  78  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. С++: 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).