Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ.....5
1.1 Описание входных данных.....6
1.2 Описание выходных данных.....6
2 МЕТОД РЕШЕНИЯ.....7
3 ОПИСАНИЕ АЛГОРИТМОВ.....8
3.1 Алгоритм функции main.....8
3.2 Алгоритм конструктора класса ArrayObject.....9
3.3 Алгоритм деструктора класса ArrayObject.....9
3.4 Алгоритм метода print_array класса ArrayObject.....10
3.5 Алгоритм метода get_array класса ArrayObject.....10
3.6 Алгоритм метода set_array класса ArrayObject.....11
4 БЛОК-СХЕМЫ АЛГОРИТМОВ.....12
5 КОД ПРОГРАММЫ.....16
5.1 Файл ArrayObject.cpp.....16
5.2 Файл ArrayObject.h.....17
5.3 Файл main.cpp.....18
6 ТЕСТИРОВАНИЕ.....20
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....21

1 ПОСТАНОВКА ЗАДАЧИ

Дан объект следующей конструкции:

В закрытом доступе имеется указатель на массив целого типа.

Конструктору объекта передается целочисленный параметр. Параметр должен иметь значение больше 4. По значению параметра определяется размерность целочисленного массива из закрытой области и каждому элементу присваивается это же значение.

Объект имеет функциональность, по которой выводит содержимое целочисленного массива. Вывод производит последовательно, разделяя значения двумя пробелами.

Функциональность объекта можно расширить по усмотрению разработчика не более чем на два метода.

Спроектировать систему, которая содержит два объекта. Для построения системы последовательно, с новых строк вводятся целочисленные значения. Если значение меньше или равно 4, то создание системы прекращается и выводится сообщение. Если система построена, то посредством параметризованного конструктора создаются объекты.

Далее система функционирует по алгоритму:

1. . . .
2. Первому объекту присвоить второй объект.
3. . . .
4. С первой строки вывести содержимое массива первого объекта.
5. . . .
6. Со второй строки вывести содержимое массива второго объекта.

1.1 Описание входных данных

Первая строка:

«Целое число»

Вторая строка:

«Целое число»

Пример.

```
5
8
```

1.2 Описание выходных данных

Если система была построена, то в первой строке:

«Целое число» «Целое число» . . .

Во второй строке:

«Целое число» «Целое число» . . .

Если система не была построена, то в первой строке выводится некорректное значение и вопросительный знак:

«Целое число»?

Пример вывода.

```
5  5  5  5  5
8  8  8  8  8  8  8  8
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект first_object класса ArrayObject предназначен для предназначен для функционирования системы;
- объект second_object класса ArrayObject предназначен для предназначен для функционирования системы;
- функция main для предназначен для функционирования системы;
- cout - объект потокового вывода;
- cin - объект потокового ввода;
- оператор освобождения памяти delete[].

Класс ArrayObject:

- свойства/поля:
    - поле поле, которое хранит указатель на массив:
        - наименование — array;
        - тип — указатель на массив;
        - модификатор доступа — public;
    - поле параметр конструктора, определяющее размер масссива:
        - наименование — size;
        - тип — целочисленный;
        - модификатор доступа — private;
- функционал:
    - метод ArrayObject — Конструктор;
    - метод ~ArrayObject — Деструктор;
    - метод print_array — Вывод значений;
    - метод get_array — Получение значения указателя;
    - метод set_array — Установка значения указателя.

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 функции main

Функционал: основной алгоритм работы программы.

Параметры: отсутсвуют.

Возвращаемое значение: int - индикатор корректности завершения работы программы.

Алгоритм функции представлен в таблице 1.

Таблица 1 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		объявление целочисленных input_one, input_two	2
2		ввод значения переменной input_one	3
3	input_one <= 4	вывод значение переменной input_one , "?"	∅
			4
4		создание объекта first_object класса "ArrayObject" и инициализация значением input_one	5
5		ввод значения переменной input_two	6
6	input_two <= 4	вывод значение переменной input_two, "?"	∅
			7
7		создание объекта second_object класса "ArrayObject" и инициализация значением input_two	8
8		вызов метода "print_array()" объекта first_object, который выводит содержимое массива "array"	9

№	Предикат	Действия	№ перехода
		этого объекта	
9		вызов метода "print_array()" объекта second_object, который выводит содержимое массива "array" этого объекта	∅

3.2 Алгоритм конструктора класса ArrayObject

Функционал: конструктор.

Параметры: int size.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса ArrayObject

№	Предикат	Действия	№ перехода
1		по адресу указателя создаем динамический массив размерностью size+1	2
2		инициализация целочисленной переменной i = 0	3
3	i < size	i++, присваивание каждому элементу значение 'size'	3
			4
4		присваивание элементу массива 'array' с индексом 'size' значение -1	∅

3.3 Алгоритм деструктора класса ArrayObject

Функционал: Деструктор.

Параметры: отсутсвуют .

Алгоритм деструктора представлен в таблице 3.

Таблица 3 – Алгоритм деструктора класса ArrayObject

№	Предикат	Действия	№ перехода
1		освобождение памяти, выделенную для массива 'array'	∅

3.4 Алгоритм метода print_array класса ArrayObject

Функционал: Вывод значений.

Параметры: public.

Возвращаемое значение: отсутвуют.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода print_array класса ArrayObject

№	Предикат	Действия	№ перехода
1		определение метода 'print_array()'	2
2		инициализация переменной i = 0	3
3	array[i] != -1	вывод текущего элемента массива 'array'	3
			4
4	array[i + 1] != -1	выводим пробел после элемента массива	4
			5
5		оператор endl	∅

3.5 Алгоритм метода get_array класса ArrayObject

Функционал: Получение значения указателя.

Параметры: отсутсвуют.

Возвращаемое значение: указатель на массив array.

Алгоритм метода представлен в таблице 5.

Таблица 5 – Алгоритм метода get_array класса ArrayObject

№	Предикат	Действия	№ перехода
1		определятся метод 'get_array()', который вовзращает указатель на массив целых чисел	2
2		вовзращается указатель на массив 'array'	∅

3.6 Алгоритм метода set_array класса ArrayObject

Функционал: Установка значения указателя.

Параметры: указатель на целочисленное значение.

Возвращаемое значение: отсутсвует.

Алгоритм метода представлен в таблице 6.

Таблица 6 – Алгоритм метода set_array класса ArrayObject

№	Предикат	Действия	№ перехода
1		определяется метод "set_array()" класса ArrayObject, который устанавливает значение указателя 'array' равным переданному указателю 'ptr'	2
2		устанавливается значение указателя 'array' равным значению указателю 'ptr'	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

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

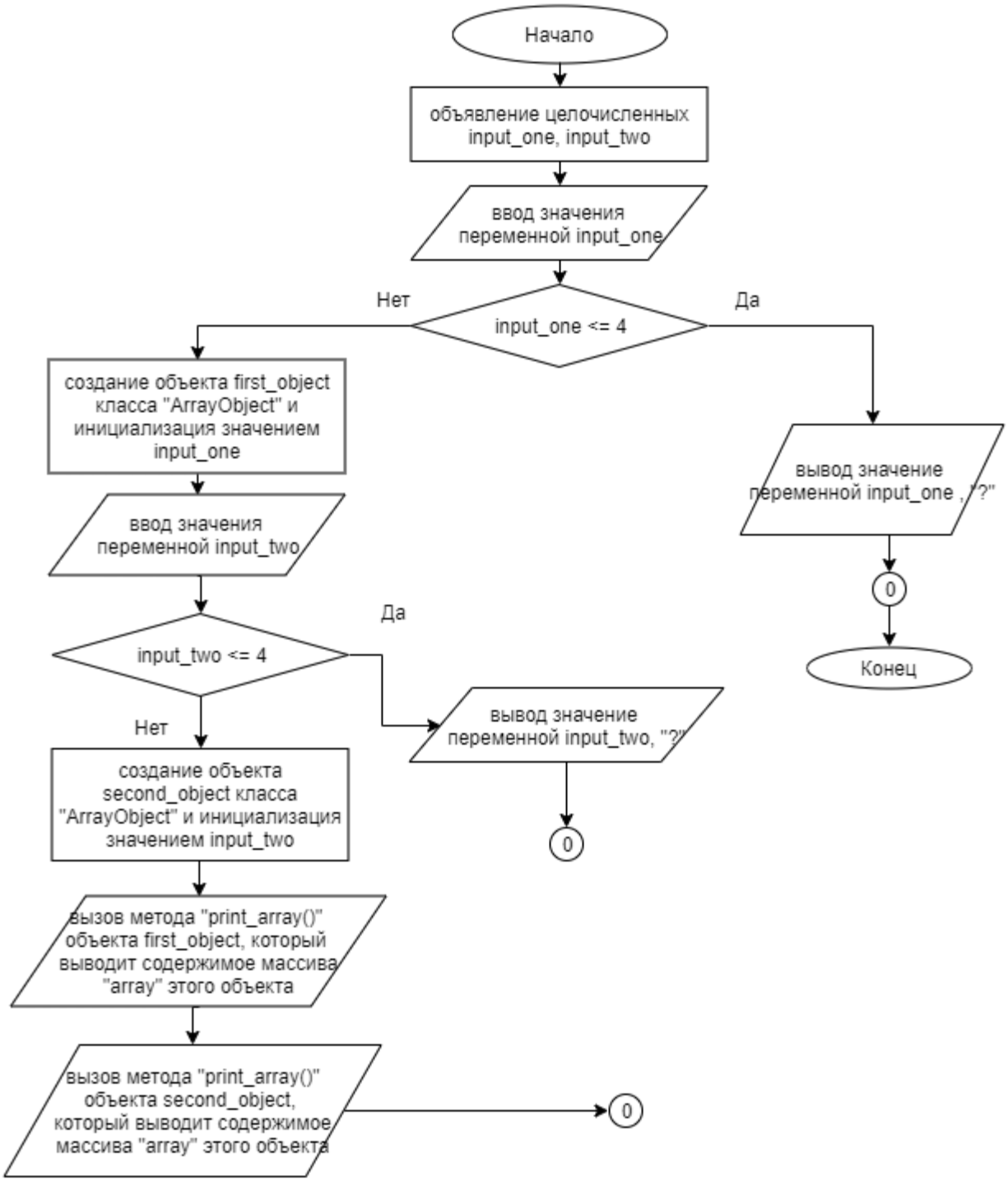

Рисунок 1 – Блок-схема алгоритма

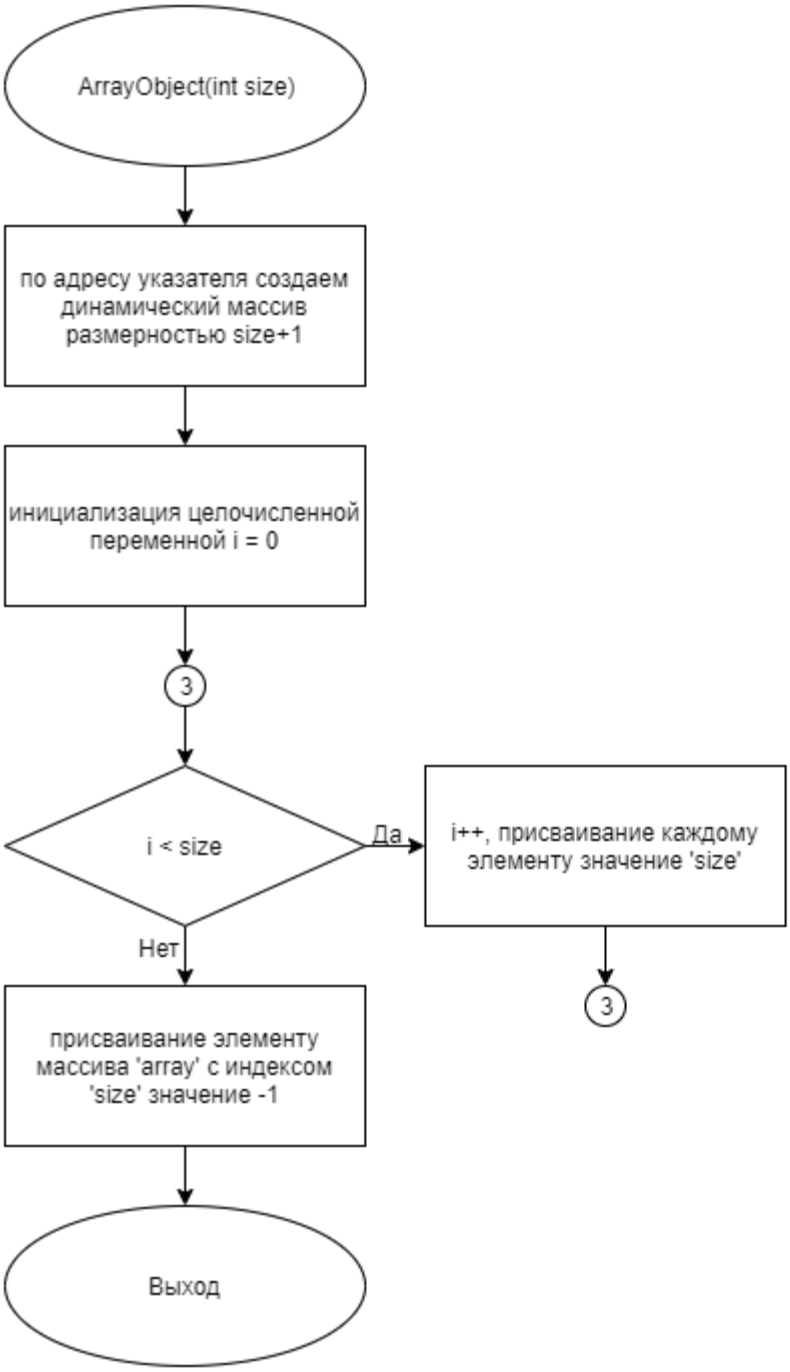

Рисунок 2 – Блок-схема алгоритма

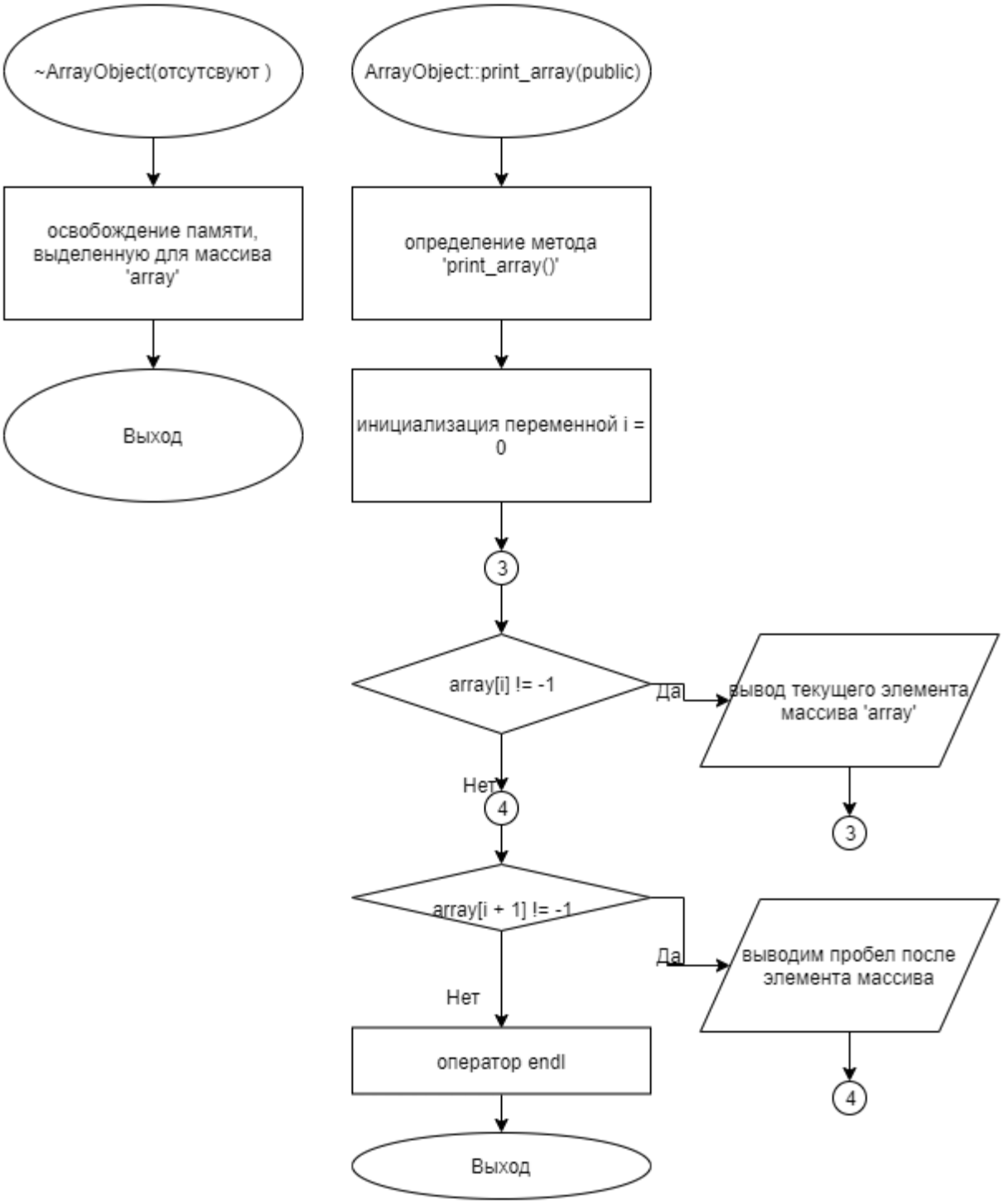

Рисунок 3 – Блок-схема алгоритма

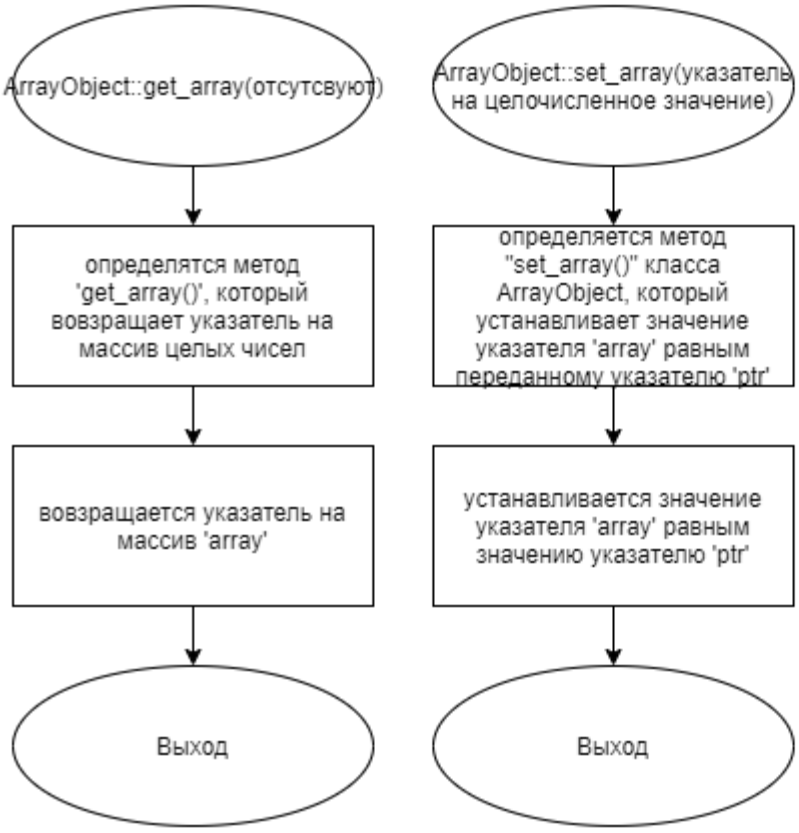

Рисунок 4 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл ArrayObject.cpp

Листинг 1 – ArrayObject.cpp

```
#include "ArrayObject.h"

using namespace std;

ArrayObject::ArrayObject(int size) {
   array = new int[size + 1];
   for(int i = 0; i < size; ++i) {
      array[i] = size;
   }
   array[size] = -1;
}

ArrayObject::~ArrayObject() {
   delete[] array;
}

void ArrayObject::print_array() const {
   int i = 0;
   while(array[i] != -1) {
      cout << array[i];
      if(array[i + 1] != -1) {
         cout << "  ";
      }
      ++i;
   }
   cout << endl;
}

//void ArrayObject::set_element(int index, int value) {
// if(index >= 0 && index < array[index] != -1) {
//    array[index] = value;
// }
//}

//int ArrayObject::get_element(int index) const {
// if(index >= 0 && index < array[index] != -1) {
//    return array[index];
// }
// return -1;
```

```
//}

int* ArrayObject::get_array() const{
   return array;
}

void ArrayObject::set_array(int* ptr) {
   array = ptr;
}

//#include "ArrayObject.h"
//using namespace std;
//ArrayObject::ArrayObject(int size) {
// array = new int[size + 1];
// for(int i = 0; i < size; ++i) {
//    array[i] = size;
// }
// array[size] = -1;
//}

//ArrayObject::~ArrayObject() {
// delete[] array;
//}

//void ArrayObject::print_array() const {
// int i = 0;
// while(array[i] != -1) {
//    cout << array[i];
//    if(array[i + 1] != -1) {
//       cout << "  ";
//    }
//    ++i;
// }
// cout << endl;
//}
```

5.2 Файл ArrayObject.h

Листинг 2 – ArrayObject.h

```
#ifndef __ARRAYOBJECT__H
#define __ARRAYOBJECT__H

#include <iostream>

using namespace std;

class ArrayObject {
private:
   int* array;
```

```
public:
   ArrayObject(int size);
   ~ArrayObject();
   void print_array() const;
   //void set_element(int index, int value);
   //int get_element(int index) const;

   int* get_array() const;

   void set_array(int* ptr);
};

#endif
```

5.3 Файл main.cpp

Листинг 3 – main.cpp

```
#include "ArrayObject.h"
#include <iostream>
#include <stdlib.h>
#include <stdio.h>

using namespace std;

int main() {
   int input_one, input_two;

   cin >> input_one;

   if (input_one <= 4) {
      cout << input_one << "?";
      return 0;
   }

   ArrayObject first_object(input_one);

   cin >> input_two;

   if (input_two <= 4) {
      cout << input_two << "?";
      return 0;
   }

   ArrayObject second_object(input_two);

   first_object.print_array();
   second_object.print_array();

   return 0;
```

```
}

//#include "ArrayObject.h"
//#include <iostream>
//#include <stdlib.h>
//#include <stdio.h>

//using namespace std;

//int main() {
// int input_one, input_two;
// cin >> input_one >> input_two;

// if (input_one <= 4 || input_two <= 4) {
//    cout << (input_one <= 4 ? input_one : input_two) << "?";
//    return 0;
// }

// ArrayObject first_object(input_one);
// ArrayObject second_object(input_two);

// int* temp_pointer = first_object.array;

// first_object = second_object;

// first_object.array = temp_pointer;

// first_object.print_array();
// second_object.print_array();

// return 0;
//}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 7.

Таблица 7 – Результат тестирования программы

Входные данные	**Ожидаемые выходные данные**	**Фактические выходные данные**
5 8	5 5 5 5 5 8 8 8 8 8 8 8 8	5 5 5 5 5 8 8 8 8 8 8 8 8
5 6	5 5 5 5 5 6 6 6 6 6 6	5 5 5 5 5 6 6 6 6 6 6

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).