# Код

#include <iostream>

using namespace std;

class IntArray {

private:

int size;

int\* array;

public:

IntArray();

IntArray(int, int);

~IntArray();

void inputArray();

void printArray();

int countEven(int);

int countOdd(int);

};

IntArray::IntArray() {

size = 0;

}

IntArray::IntArray(int size\_, int element) {

size = size\_;

array = new int[size];

for (int i = 0;i < size;i++) {

array[i] = element;

}

}

IntArray::~IntArray() {

delete[] array;

array = nullptr;

}

void IntArray::inputArray() {

cout << "Enter array size:\n";

cin >> size;

array = new int[size];

cout << "Enter array elements:\n";

for (int i = 0;i < size;i++) {

cin >> array[i];

}

}

void IntArray::printArray() {

cout << "Array is:\n";

for (int i = 0;i < size;i++) {

cout << array[i] << " ";

}

cout << endl;

}

int IntArray::countEven(int n) {

int count = 0;

for (int i = 0;(i < n) && (i < size);i++) {

if (array[size - 1 - i] % 2 == 0) {

++count;

}

}

return count;

}

int IntArray::countOdd(int n) {

int count = 0;

for (int i = 0;(i < n) && (i < size);i++) {

if (array[size - 1 - i] % 2 == 1) {

++count;

}

}

return count;

}

int main() {

IntArray A;

A.inputArray();

A.printArray();

cout << "Even - " << A.countEven(5) << endl << "Odd - " << " " << A.countOdd(5);

return 0;

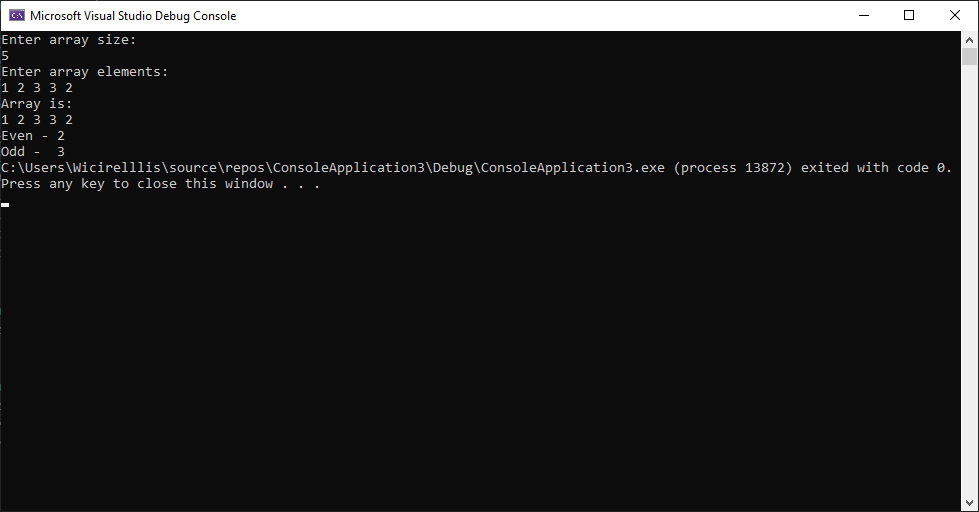
}

# 2.

Проверим работу программы на каком-нибудь входе. Пусть ввод слеующий:

5(размер массива)

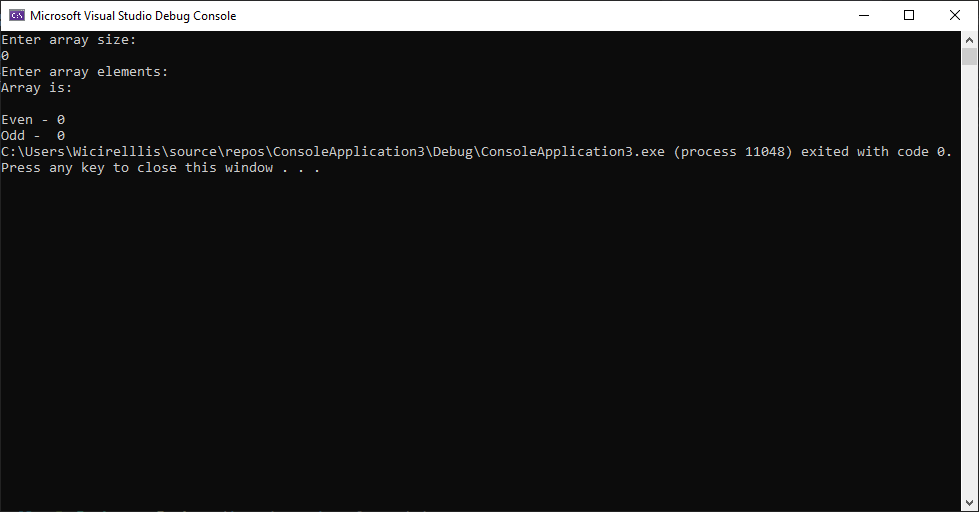
1 2 3 3 2(элементы)

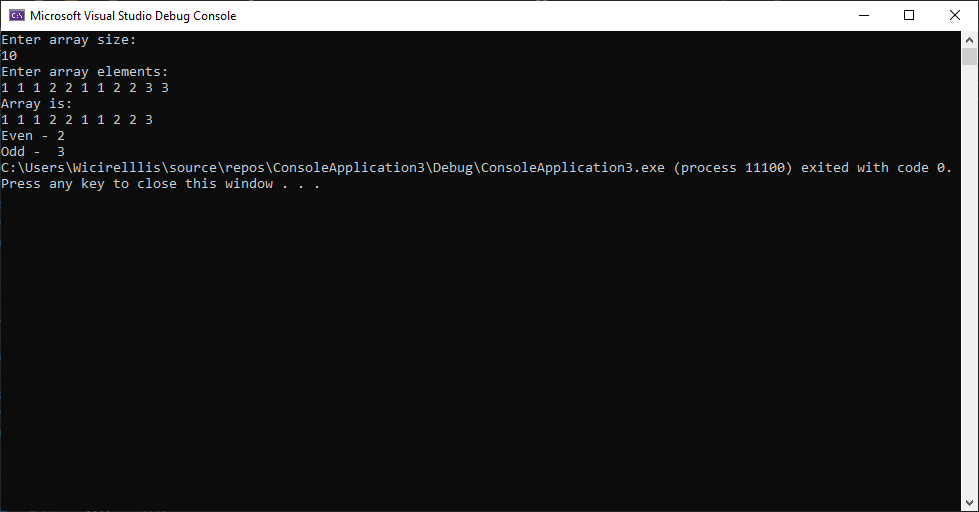


Работает (по крайней мере на данном входе) верно.

# 3.

Несколько скринов вывода





# 4. Вопросы

1) Как осуществляется определение массива произвольной длины?

Массив – суть набор чисел, определяется количеством элементов и самими элементами. Для массива произвольной длинны заранее не известно сколько памяти понадобится, поэтому сначала узнаем размер, а затем выделяем память.

2) Важно ли до ввода массива задавать размерность массива? Почему?  
Да, т.к. массив лежит в памяти непрерывным куском и мы не можем для массива из 20 элементов сначала выделить место для 10 элементов, а потом еще для 10, т.к. это будут два куска памяти, вообще говоря, не лежащие рядом.

В принципе, можно и не спрашивать размер заранее, а выделять сколько-нибудь места (например под 10 элемнтов) и если не хватило, то выделять заново больше (например под 20 элемнтов), коприровать первые 10 элементов и удалять оригинальные 10 элемннтов (чтобы не тратить излишнюю память). std::vector вроде, равно так и работает.

3) Как удалить массив?

delete[] array;

Это освобождение памяти, в которой лежат элементы.