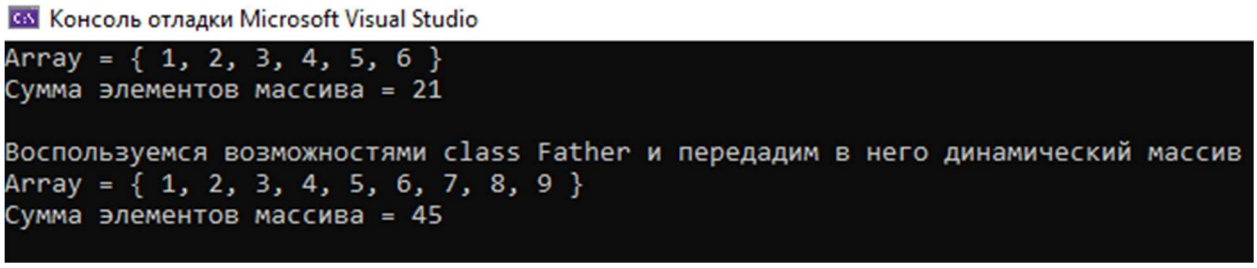
Задание 1

Результат работы программы:



Main.cpp

```cpp
#include <iostream>

/* Задание 1
* Базовый класс – Father (отец)
* Наследуемый класс – Son (сын)
*
* Я немного забежал вперед и усложнил задачу – моя программа в калссе Son
* прнинимает в себя ряд чисел, как и было сказано в условии, а вот класс
* Father работает уже с одномерным массивом. Таким образом, класс Son
* теперь более осмысленный: он не только передает значения, но еще и
* интерпретирует их, а класс Father стал более универсален, что и показывает
* функция SetDataFather() в корторую мы можеме передать массив любой длинны
*/

using namespace std;

class Father
{
protected:
    int* Array;
    int Size;

public:
    Father(int NewSize = NULL, int* NewArray = NULL)
    {
        Size = NewSize;
        Array = NewArray;
    }

    void PrintArray()
    {
        cout << "Array = { ";
        for (int i = 0; i < Size; i++)
        {
            if (i != Size - 1 ? cout << Array[i] << ", " : cout << Array[i] << " ");
        }
        cout << "}";
    }

    int GetSumArray()
    {
        int SumArray = 0;

        for (int i = 0; i < Size; i++)
        {
            SumArray += Array[i];
        }

        return SumArray;
```

```
    }

    ~Father() {}
};

class Son : private Father
{
    int A, B, C, D, E, F;

public:
    Son(int NewA, int NewB, int NewC, int NewD, int NewE, int NewF)
    {
        int Size = 6;
        int* MyArray = new int[Size];

        A = NewA; MyArray[0] = NewA;
        B = NewB; MyArray[1] = NewB;
        C = NewC; MyArray[2] = NewC;
        D = NewD; MyArray[3] = NewD;
        E = NewE; MyArray[4] = NewE;
        F = NewF; MyArray[5] = NewF;

        int* PointerMyArray = MyArray;

        Father::Size = Size;              // Передаем в функцию длинну массива
        Father::Array = PointerMyArray;   // Создаем указатель для передачи в
функцию
    }
    ~Son() {}

    void SetDataFather(int NewSize, int* NewArray)
    {
        Father::Size = NewSize;
        Father::Array = NewArray;
    }

    // Восстанавливаем область видимости
    Father::PrintArray;
    Father::GetSumArray;
};



void main()
{
      setlocale(LC_ALL, "Rus");

    Son MyClass(1, 2, 3, 4, 5, 6);      // Строго как в задании
    MyClass.PrintArray();
    cout << "\nСумма элементов массива = " << MyClass.GetSumArray() << "\n";

    cout << "\nВоспользуемся возможностями class Father и передадим в него
динамический массив\n";
    int MyArray[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 }; // Любой динамический массив, в
нашем случае заданный в ручную

    int LenMyArray = (sizeof(MyArray) / sizeof(*MyArray)); // Узнаем длинну для
передачи в функцию
    int* PointerMyArray = MyArray;                         // Создаем указатель для
передачи в функцию

    MyClass.SetDataFather(LenMyArray, PointerMyArray);
    MyClass.PrintArray();
    cout << "\nСумма элементов массива = " << MyClass.GetSumArray() << "\n";
};
```

Задание 2

В задании просили использовать модульный принцип построения программ. Поэтому программа разделена на три файла:

Main.cpp
MyClass.cpp
HeaderMyClass.h

Результат работы программы:

Консоль отладки Microsoft Visual Studio

```
Случано заполненный массив в постоянной памяти (статический массив)
      Array = { 23, 29, 68, 48, 31, 24, 98, 31, 94, 12,
                41, 18, 54, 46, 55, 69, 12, 94, 89, 32,
                72, 82, 76, 48, 91, 73, 73, 23, 62, 18,
                61, 56, 50, 70, 39, 69, 18, 34, 87, 13,
                24, 59, 73, 12, 76, 84, 18, 51, 29, 23, }
                Сумма элементов массива = 2532

Динамический массив содержащий только нечетные элементы статического массива
  Odd Array = { 23, 29, 31, 31, 41, 55, 69, 89, 91, 73,
                73, 23, 61, 39, 69, 87, 13, 59, 73, 51,
                29, 23 }
                Сумма элементов массива = 1132

Отсортированный методом пузырька статический массив
      Array = { 12, 12, 12, 13, 18, 18, 18, 18, 23, 23,
                23, 24, 24, 29, 29, 31, 31, 32, 34, 39,
                41, 46, 48, 48, 50, 51, 54, 55, 56, 59,
                61, 62, 68, 69, 69, 70, 72, 73, 73, 73,
                76, 76, 82, 84, 87, 89, 91, 94, 94, 98, }

Отсортированный методом выбора динамический массив
  Odd Array = { 13, 23, 23, 23, 29, 29, 31, 31, 39, 41,
                51, 55, 59, 61, 69, 69, 73, 73, 73, 87,
                89, 91 }
```

Main.cpp

```cpp
#include <iostream>
#include <stdlib.h>
#include <time.h>

#include "HeaderMyClass.h"

using namespace std;

void main()
{
	setlocale(LC_ALL, "Rus");
    srand(time(NULL));

    SortArray StartDataMyArray; // Запустили работать ветку классов

    cout << "Случано заполненный массив в постоянной памяти (статический массив)\n";
    StartDataMyArray.PrintArray();
    cout << "\n\t\tСумма элементов массива = " << StartDataMyArray.GetSummArray();

    cout << "\n\nДинамический массив содержащий только нечетные элементы
статического массива\n";
    StartDataMyArray.PrintOddArray();
    cout << "\n\t\tСумма элементов массива = " <<
StartDataMyArray.GetSummOddArray();

    cout << "\n\nОтсортированный методом пузырька статический массив\n";
    StartDataMyArray.BubbleSorting();
    StartDataMyArray.PrintArray();

    cout << "\n\nОтсортированный методом выбора динамический массив\n";
    StartDataMyArray.ChoiceSorting();
    StartDataMyArray.PrintOddArray();
};
```

MyClass.cpp

```cpp
#include <iostream>
#include <stdlib.h>
#include <time.h>

#include "HeaderMyClass.h"

using namespace std;

/*    БАЗОВЫЙ КЛАСС ARRAY    */
ARRAY::ARRAY()
{
    for (int i = 0; i < Size; i++)
    {
        Array[i] = 10 + rand() % 89;
    }
}
ARRAY::~ARRAY() {}

void ARRAY::PrintArray()
{
    int NumberItemsInRow = 0;

    cout << "      Array = { ";
    for (int i = 0; i < Size; i++)
    {
        if (NumberItemsInRow < 10)
        {
            cout << Array[i] << ", ";
            NumberItemsInRow++;
        }
        else
        {
            if (i != Size - 1 ? cout << "\n\t\t" << Array[i] << ", " : cout << "\n"
<< Array[i] << " ");
            NumberItemsInRow = 1;
        }
    }
    cout << "}";
}

int ARRAY::GetSummArray()
{
    int SummArray = 0;
    for (int i = 0; i < Size; i++)
    {
        SummArray += Array[i];
    }
    return SummArray;
}


/*    НАСЛЕДУЕМЫЙ КЛАСС OddArrayElement    */
OddArrayElement::OddArrayElement() : ARRAY()
{
    SizeDynamic = CountOddArrayElement();
    ArrayDynamic = new int[SizeDynamic];

    int Pen = 0;
    for (int i = 0; i < Size; i++)
    {
        if (Array[i] % 2 != 0)
        {
            ArrayDynamic[Pen] = Array[i];
            Pen++;
```

```
        }
    }
}
OddArrayElement::~OddArrayElement() {}

int OddArrayElement::CountOddArrayElement()
{
    int CountOdd = 0;
    for (int i = 0; i < Size; i++)
    {
        if (Array[i] % 2 != 0)
        {
            CountOdd++;
        }
    }
    return CountOdd;
}

int OddArrayElement::GetSummOddArray()
{
    int SummArray = 0;
    for (int i = 0; i < SizeDynamic; i++)
    {
        SummArray += ArrayDynamic[i];
    }
    return SummArray;
}


void OddArrayElement::PrintOddArray()
{
    int NumberItemsInRow = 0;
    cout << "  Odd Array = { ";
    for (int i = 0; i < SizeDynamic; i++)
    {
        if (NumberItemsInRow < 10)
        {
            if (i != SizeDynamic - 1 ? cout << ArrayDynamic[i] << ", " : cout <<
ArrayDynamic[i] << " ");
            NumberItemsInRow++;
        }
        else
        {
            if (i != SizeDynamic - 1 ? cout << "\n\t\t" << ArrayDynamic[i] << ", " :
cout << "\n\t\t" << ArrayDynamic[i] << " ");
            NumberItemsInRow = 1;
        }
    }
    cout << "}";
}


/*    НАСЛЕДУЕМЫЙ КЛАСС SortArray   */
SortArray::SortArray() : OddArrayElement() {}
SortArray::~SortArray() {}

void SortArray::BubbleSorting()
{
    for (int i = 0; i < Size; i++)
    {
        bool flag = true;
        for (int j = 0; j < Size - (i + 1); j++)
        {
            if (Array[j] > Array[j + 1])
            {
```

```
                flag = false;
                swap(Array[j], Array[j + 1]);
            }
        }
        if (flag)
        {
            break;
        }
    }
}

void SortArray::ChoiceSorting()
{
    for (int repeat_counter = 0; repeat_counter < SizeDynamic; repeat_counter++)
    {
        for (int element_counter = repeat_counter + 1; element_counter <
SizeDynamic; element_counter++)
        {
            if (ArrayDynamic[repeat_counter] > ArrayDynamic[element_counter])
            {
                swap(ArrayDynamic[repeat_counter], ArrayDynamic[element_counter]);
            }
        }
    }
}
```

HeaderMyClass.h

```
#pragma once

using namespace std;

class ARRAY
{
protected:
    static const int Size = 50;
    int Array[Size];

public:
    ARRAY();
    ~ARRAY();
    void PrintArray();  // Печать массива
    int GetSummArray(); // Сумма элементов массива
};

class OddArrayElement : private ARRAY
{
protected:
    int* ArrayDynamic;
    int SizeDynamic;

public:
    ARRAY::Array;
    ARRAY::Size;

    OddArrayElement();
    ~OddArrayElement();

    ARRAY::PrintArray;
    ARRAY::GetSummArray;

    /* CountOddArrayElement - Вспопогательная функция необходимая для
    подсчета нечетных элементов в Array[Size] чтобы в послдствии задать
    динамический массив нужной длинны */
    int CountOddArrayElement();
    void PrintOddArray();       // Печать массива
    int GetSummOddArray();      // Сумма элементов массива
};

class SortArray : private OddArrayElement
{
public:
    SortArray();
    ~SortArray();

    OddArrayElement::Array;
    OddArrayElement::Size;
    OddArrayElement::ArrayDynamic;
    OddArrayElement::SizeDynamic;


    OddArrayElement::PrintArray;
    OddArrayElement::PrintOddArray;
    OddArrayElement::GetSummArray;
    OddArrayElement::GetSummOddArray;
    void BubbleSorting(); // Сортировка с применением метода пузырька
    void ChoiceSorting(); // Сортировка с применением метода выбора наименьшего
};
```