Примеры решения задач к экзамену (2-ой семестр)

Все варианты рассмотрены на примере задачи:

Вычислить количество положительных элементов в исходном массиве вещественных чисел. Сформировать новый массив из элементов исходного массива, значения которых больше найденного числа — количества положительных элементов.

1. **Реализация программы создания класса, не содержащего элементов-функций. Обработка массива представителей такого класса**

Постановка задачи

Разработать программу обработки динамического массива, элементы которого представлены объектами пользовательского класса без методов.

Требования к реализации

- Реализовать класс `Number`, содержащий:

Поля:

- вещественное значение.

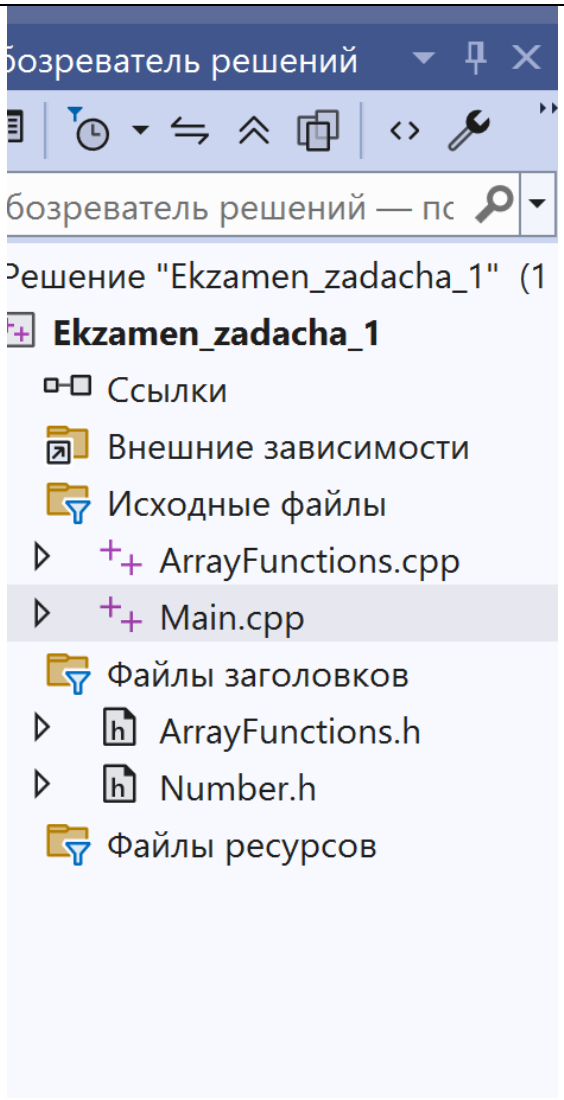
Особенности:

- класс не должен содержать методов;
- доступ к данным осуществляется напрямую.

Обработка массива

Реализовать функции:

- создание динамического массива объектов класса;
- заполнение массива случайными значениями;
- вывод массива;
- подсчет количества положительных элементов;
- формирование нового массива:
- из элементов, значения которых больше количества положительных элементов исходного массива.



Number.h

```h
#pragma once

class Number
{
public:
    double value;
};
```

ArrayFunctions.h

```h
#pragma once
#include "Number.h"


Number* create_array(int size);
void fill_array(Number* arr, int size);
void print_array(Number* arr, int size);
int count_positive(Number* arr, int size);
Number* build_new_array(Number* arr, int size, int pos_count, int& new_size);
```

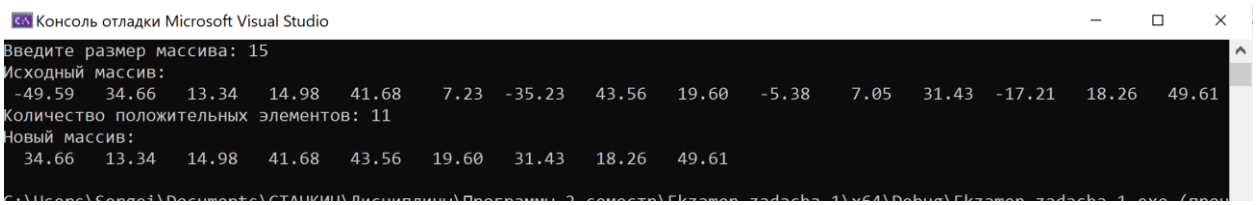
ArrayFunctions.cpp

```cpp
#include "ArrayFunctions.h"
#include <iostream>
#include <cstdlib>

using namespace std;

// Выделение памяти
Number* create_array(int size)
{
    return new Number[size];
}

// Заполнение массива
void fill_array(Number* arr, int size)
{
    for (int i = 0; i < size; i++) {
        arr[i].value = (rand() % 10001 - 5000) / 100.0;
    }
}

// Вывод массива
void print_array(Number* arr, int size)
{
    for (int i = 0; i < size; i++) {
        cout.width(7);
        cout << fixed;
        cout.precision(2);
        cout << arr[i].value << " ";
    }
    cout << endl;
}

// Подсчет положительных элементов
int count_positive(Number* arr, int size)
{
    int count = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i].value > 0) {
            count++;
        }
    }
    return count;
}

// Формирование нового массива
Number* build_new_array(Number* arr, int size,int pos_count, int& new_size)
{
    int count = 0;

    for (int i = 0; i < size; i++) {
        if (arr[i].value > pos_count) {
            count++;
        }
    }
    new_size = count;
    Number* new_arr = new Number[count];
    int j = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i].value > pos_count) {
            new_arr[j++].value = arr[i].value;
        }
    }
```

```
    return new_arr;
}
```

Main.cpp

```
#include "ArrayFunctions.h"
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    setlocale(LC_ALL, "Russian");

    int n;
    cout << "Введите размер массива: ";
    cin >> n;

    Number* arr = create_array(n);

    fill_array(arr, n);

    cout << "Исходный массив:\n";
    print_array(arr, n);

    int pos_count = count_positive(arr, n);

    cout << "Количество положительных элементов: " << pos_count << endl;

    int new_size;
    Number* new_arr = build_new_array(arr, n, pos_count, new_size);

    cout << "Новый массив:\n";
    print_array(new_arr, new_size);

    delete[] arr;
    delete[] new_arr;

    return 0;
}
```



2. Реализация класса, содержащего элементы-функции

Постановка задачи

Разработать программу обработки динамического массива с использованием пользовательского класса, содержащего методы для работы с данными.

Требования к реализации

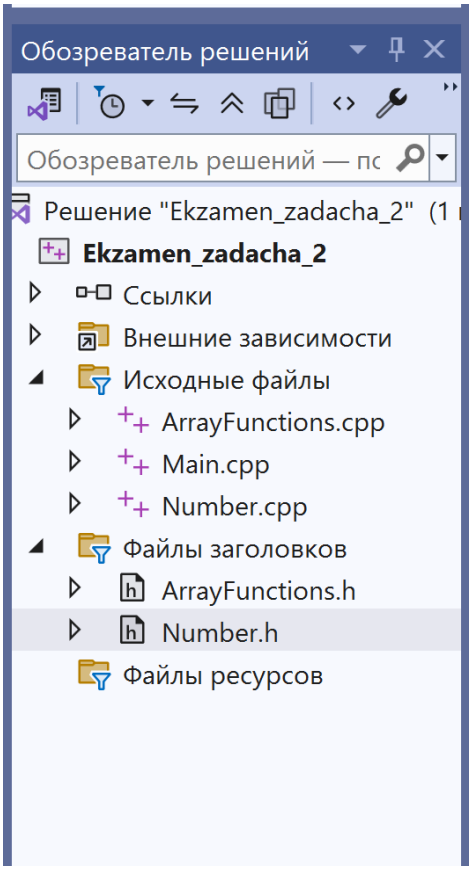
- Реализовать класс `Number`, содержащий:

Поля:

- вещественное значение.

Методы:

- установка значения;
- получение значения;
- проверка, является ли число положительным;
- проверка, больше ли значение заданного числа;
- вывод значения.



Number.h

```h
#pragma once
#include <iostream>

class Number
{
private:
    double value;

public:
    Number();
    Number(double v);

    void setValue(double v);
    double getValue() const;

    bool isPositive() const;
    bool greaterThan(double x) const;

    void print() const;
};
```

ArrayFunctions.h

```h
#pragma once
#include "Number.h"

Number* create_array(int size);
void fill_array(Number* arr, int size);
void print_array(Number* arr, int size);
int count_positive(Number* arr, int size);
Number* build_new_array(Number* arr, int size,int pos_count, int& new_size);
```

ArrayFunctions.cpp

```cpp
#include "ArrayFunctions.h"
#include <iostream>
#include <cstdlib>

using namespace std;

Number* create_array(int size)
{
    return new Number[size];
}

void fill_array(Number* arr, int size)
{
    for (int i = 0; i < size; i++) {
        double val = (rand() % 10001 - 5000) / 100.0;
        arr[i].setValue(val);
    }
}

void print_array(Number* arr, int size)
{
    for (int i = 0; i < size; i++) {
        arr[i].print();
    }
    cout << endl;
}
```

```cpp
int count_positive(Number* arr, int size)
{
    int count = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i].isPositive()) {
            count++;
        }
    }
    return count;
}

Number* build_new_array(Number* arr, int size, int pos_count, int& new_size)
{
    int count = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i].greaterThan(pos_count)) {
            count++;
        }
    }
    new_size = count;
    Number* new_arr = new Number[count];
    int j = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i].greaterThan(pos_count)) {
            new_arr[j++] = arr[i];
        }
    }
    return new_arr;
}
```

Number.cpp

```cpp
#include "Number.h"
using namespace std;

Number::Number() : value(0) {}

Number::Number(double v) : value(v) {}

void Number::setValue(double v)
{
    value = v;
}

double Number::getValue() const
{
    return value;
}

bool Number::isPositive() const
{
    return value > 0;
}

bool Number::greaterThan(double x) const
{
    return value > x;
}

void Number::print() const
{
    cout.width(7);
    cout << fixed;
```

```
    cout.precision(2);
    cout << value << " ";
}
```

Main.cpp

```
#include <iostream>
#include <cstdlib>
#include "ArrayFunctions.h"

using namespace std;

int main()
{
    setlocale(LC_ALL, "Russian");

    int n;
    cout << "Введите размер массива: ";
    cin >> n;

    Number* arr = create_array(n);

    fill_array(arr, n);

    cout << "Исходный массив:\n";
    print_array(arr, n);
    int pos_count = count_positive(arr, n);
    cout << "Количество положительных элементов: " << pos_count << endl;
    int new_size;
    Number* new_arr = build_new_array(arr, n, pos_count, new_size);

    cout << "Новый массив:\n";
    print_array(new_arr, new_size);

    delete[] arr;
    delete[] new_arr;
    return 0;
}
```

3. Реализация класса, содержащего элементы-функции, осуществляющие перегрузку операций

Постановка задачи

Разработать программу, выполняющую обработку динамического массива чисел с использованием пользовательского класса.

Требования к реализации

- Реализовать класс `Number`, содержащий:

Поля:

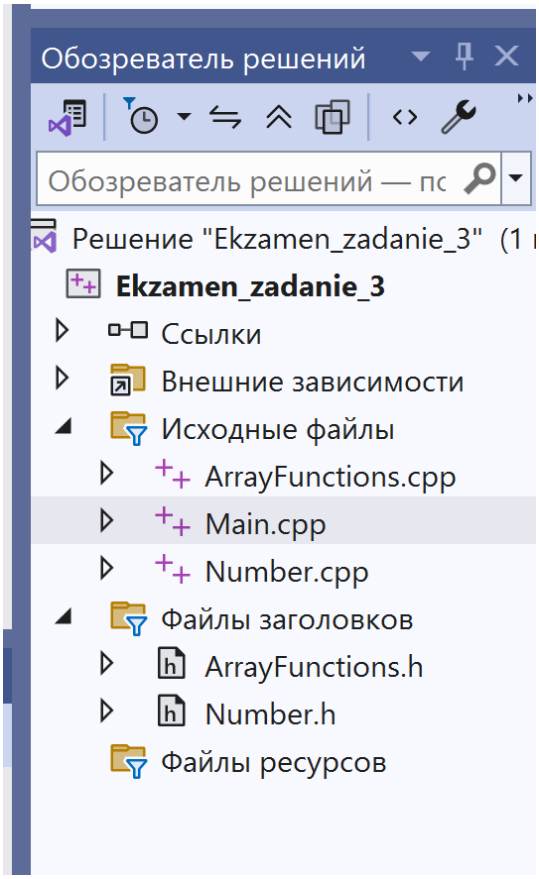
- вещественное значение.

Методы:

- установка значения;
- получение значения;
- вывод значения.

Перегруженные операторы:

- оператор > для сравнения:
- с вещественным числом;
- с другим объектом класса.



Number.h

```
{
private:
    double value;

public:
    Number();
    Number(double v);


    void setValue(double v);
    double getValue() const;
    bool operator>(double x) const;
    bool operator>(const Number& other) const;
    void print() const;
};
```

ArrayFunctions.h

```
#pragma once
#include "Number.h"

Number* create_array(int size);
void fill_array(Number* arr, int size);
void print_array(Number* arr, int size);
int count_positive(Number* arr, int size);
Number* build_new_array(Number* arr, int size, int pos_count, int& new_size);
```

Number.cpp

```cpp
#include "Number.h"
using namespace std;

Number::Number() : value(0) {}

Number::Number(double v) : value(v) {}

void Number::setValue(double v)
{
    value = v;
}

double Number::getValue() const
{
    return value;
}

bool Number::operator>(double x) const
{
    return value > x;
}

bool Number::operator>(const Number& other) const
{
    return value > other.value;
}

void Number::print() const
{
    cout.width(7);
    cout << fixed;
    cout.precision(2);
    cout << value << " ";
}
```

ArrayFunctions.cpp

```cpp
#include "ArrayFunctions.h"
#include <iostream>
#include <cstdlib>

using namespace std;

Number* create_array(int size)
{
    return new Number[size];
}

void fill_array(Number* arr, int size)
{
    for (int i = 0; i < size; i++) {
        double val = (rand() % 10001 - 5000) / 100.0;
        arr[i].setValue(val);
    }
}


void print_array(Number* arr, int size)
{
    for (int i = 0; i < size; i++) {
        arr[i].print();
```

```
    }
    cout << endl;
}

int count_positive(Number* arr, int size)
{
    int count = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] > 0) {   // ← перегруженный оператор
            count++;
        }
    }
    return count;
}

Number* build_new_array(Number* arr, int size,
    int pos_count, int& new_size)
{
    int count = 0;

    for (int i = 0; i < size; i++) {
        if (arr[i] > pos_count) {   // ← перегруженный оператор
            count++;
        }
    }

    new_size = count;
    Number* new_arr = new Number[count];

    int j = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] > pos_count) {
            new_arr[j++] = arr[i];
        }
    }

    return new_arr;
}
```

Main.cpp

```
#include <iostream>
#include <cstdlib>
#include "ArrayFunctions.h"

using namespace std;

int main()
{
    setlocale(LC_ALL, "Russian");

    int n;
    cout << "Введите размер массива: ";
    cin >> n;

    Number* arr = create_array(n);

    fill_array(arr, n);

    cout << "Исходный массив:\n";
    print_array(arr, n);

    int pos_count = count_positive(arr, n);
```

```
    cout << "Количество положительных элементов: " << pos_count << endl;

    int new_size;
    Number* new_arr = build_new_array(arr, n, pos_count, new_size);

    cout << "Новый массив:\n";
    print_array(new_arr, new_size);

    delete[] arr;
    delete[] new_arr;

    return 0;
}
```