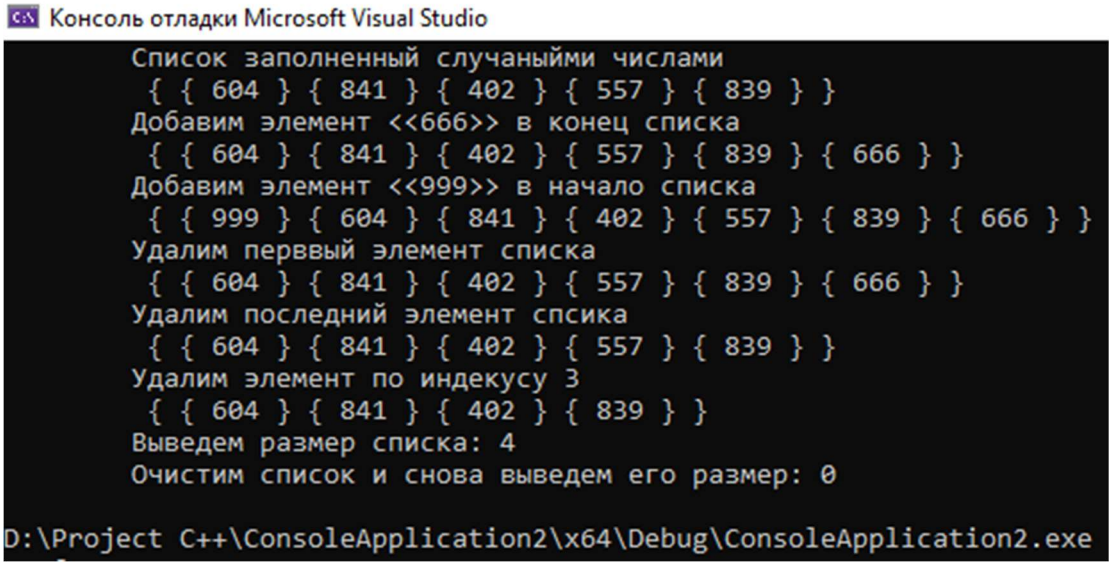
Задача 1. Объект представляет собой динамический массив.



Исходный код:

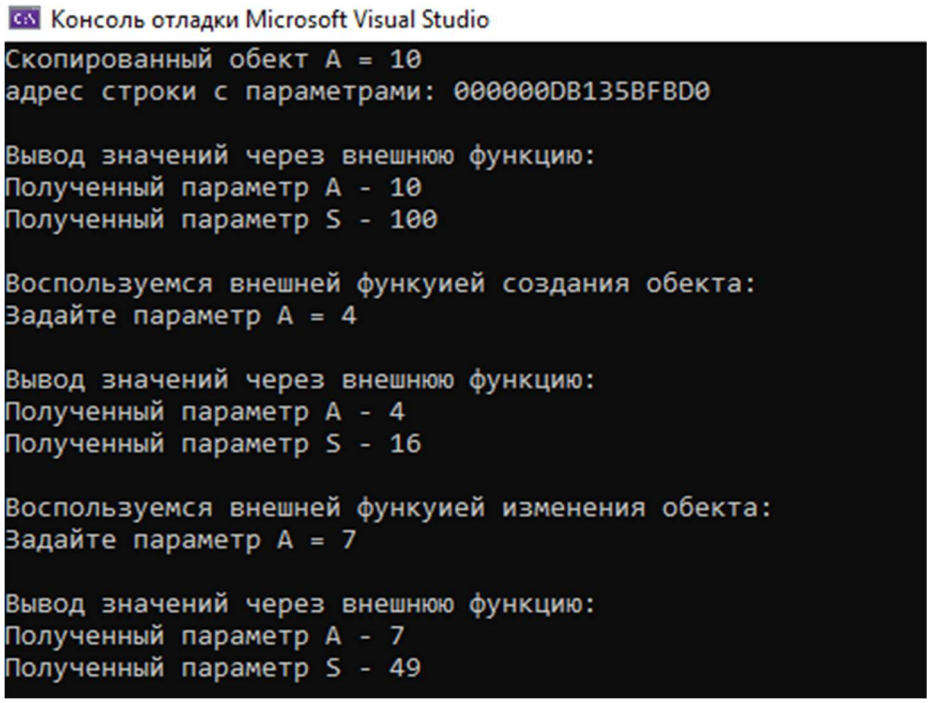
```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <string>

using namespace std;

class List
{
public:
    List();  // конструктор
    ~List(); // дистуктор

    void pop_front(); // удаление первого элемента
    void pop_back();  // удаление послднего элемента
    void push_back(int data);  // добавление новой ячейки к концу
    void push_front(int data); // добавление новой ячейки к началу
    void removeAt(int index); // удаление по указанному индексу
    void print_list(); // печать списка
    void clear(); // очистка списка
    int GetSize() { return Size; }; // показать кол-во элементов в списке

    /* Перрегрузка [] */
    int& operator[](const int index);

private:
    class Node
    {
    public:
        Node* pNext;
        int data;

        Node(int data = int(), Node* pNext = nullptr)
        {
            this->data = data;
            this->pNext = pNext;
        }
    };

    int Size;
    Node* head;
```

```
};

List::List()
{
    Size = 0;
    head = nullptr;
}

List::~List()
{
    clear();
}

void List::pop_front()
{
    Node* temp = head;
    head = head->pNext;
    delete temp;
    Size--;
}

void List::pop_back()
{
    removeAt(Size - 1);
}

void List::push_back(int data)
{
    if (head == nullptr)
    {
        head = new Node(data);
    }
    else
    {
        Node* current = this->head;
        while (current->pNext != nullptr)
        {
            current = current->pNext;
        }
        current->pNext = new Node(data);
    }

    Size++;
}

void List::push_front(int data)
{
    head = new Node(data, head);
    Size++;
}

void List::removeAt(int index)
{
    if (index == 0)
    {
        pop_front();
    }
    else
    {
        Node* previous = this->head;

        for (int i = 0; i < index - 1; i++)
        {
            previous = previous->pNext;
        }
```

```
        Node* toDelete = previous->pNext;
        previous->pNext = toDelete->pNext;

        delete toDelete;
        Size--;
    }
}

void List::print_list()
{
    Node* current = this->head;
    cout << "\t { ";
    for (int i = 0; i < GetSize(); i++)
    {
        if (i < 9)
        {
            cout << "{ " << current->data << " } ";
        }
        else
        {
            cout << "{ " << current->data << " } ";
        }
        current = current->pNext;
    }
    cout << "} " << endl;
}

void List::clear()
{
    while (Size)
    {
        pop_front();
    }
}

int& List::operator[](const int index)
{
    int counter = 0;
    Node* current = this->head;

    while (current != nullptr)
    {
        if (counter == index)
        {
            return current->data;
        }
        current = current->pNext;
        counter++;
    }
}

void CreatingFillRandomList(List& MyList, int N)
{
    for (int i = 0; i < N; i++)
    {
        MyList.push_back(rand()%888 + 101);
    }
}


int main()
{
    setlocale(LC_ALL, "ru");
    srand(time(NULL));
```

```
    List MyList;
    CreatingFillRandomList(MyList, 5);
    cout << "\tСписок заполненный случаныйми числами\n";
    MyList.print_list();

    cout << "\tДобавим элемент <<666>> в конец списка\n";
    MyList.push_back(666);
    MyList.print_list();

    cout << "\tДобавим элемент <<999>> в начало списка\n";
    MyList.push_front(999);
    MyList.print_list();

    cout << "\tУдалим перввый элемент списка\n";
    MyList.pop_front();
    MyList.print_list();

    cout << "\tУдалим последний элемент спсика\n";
    MyList.pop_back();
    MyList.print_list();

    cout << "\tУдалим элемент по индекусу 3\n";
    MyList.removeAt(3);
    MyList.print_list();

    cout << "\tВыведем размер списка: " << MyList.GetSize() <<"\n";
    MyList.clear();
    cout << "\tОчистим список и снова выведем его размер: " << MyList.GetSize() <<
"\n";
}
```

Задача 2. (Про квадрат)

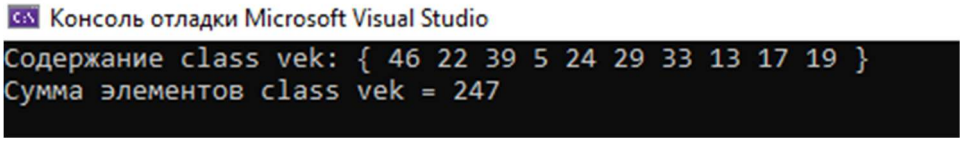


Исходный код:

```
#include <iostream>
#include <string>
#include <stdlib.h>

using namespace std;

class Square {
public:
    int A; // Сторона a
    int S; // Площадь
    string SquareParameters;

    // Конструктор
    Square(int a = 5) {
        A = a;
        S = GetS();
    }
    // Перегруженный конструктор копирования
    Square(const Square& _Square) {
        A = _Square.A;
        S = _Square.S;
    }
    // Возвращает строку параметров Square
    string* GetParameters() {
        SquareParameters = "Параметры квадрата: Сторона квадарата = "
            + to_string(A)
            + "; Площадь квадрата = "
            + to_string(S);
        return &SquareParameters;
    }
    // Метод расчета площади
    int GetS() {
        return A * A;
    }
```

```
};

// Получает объект в качестве параметра и печатает его поля
void Print(Square& _Square) {
    cout << "Полученный параметр A - " << _Square.A << endl;
    cout << "Полученный параметр S - " << _Square.S << endl;
}

// Возвращает локальный объект с параметром «сторона», заданным в этой функции
Square* LocalObject() {
    cout << "Задайте параметр A = ";
    int A; cin >> A;
    Square Kvadrat(A);
    return &Kvadrat;
}

// Принимает ссылку на объект, изменяет его и возвращает ссылку на объект
Square* LocalObject(Square& _Square) {
    cout << "Задайте параметр A = ";
    int A; cin >> A;
    _Square.A = A;
    _Square.S = _Square.GetS();
    return &_Square;
}

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

    Square Kvadrat(10); // Сработал обычный конструткор
    Square KvadratCopy(Kvadrat); // Сработал конструктор копирования
    cout << "Скопированный обект A = 10\nадрес строки с параметрами: ";
    cout << KvadratCopy.GetParameters() << endl; // Показали адрес втроенной функции
    cout << "\nВывод значений через внешнюю функцию:\n"; Print(KvadratCopy);

    cout << "\nВоспользуемся внешней функуией создания обекта:\n";
    Square KvadratPtr = *LocalObject();
    cout << "\nВывод значений через внешнюю функцию:\n"; Print(KvadratPtr);

    cout << "\nВоспользуемся внешней функуией изменения обекта:\n";
    KvadratPtr = *LocalObject(KvadratPtr);
    cout << "\nВывод значений через внешнюю функцию:\n"; Print(KvadratPtr);
}
```

Задача 3 Дружественный класс




```
#include <iostream>
#include <string>
#include <stdlib.h>

using namespace std;

class vek
{
private:
    static const int n = 10;
    int mas[n];

public:
    vek() {
        for (int i = 0; i < n; i++) mas[i] = rand() % 50 + 5;
    }

    ~vek() { }

    void show() {
        cout << "Содержание class vek: { ";
        for (int i = 0; i < n; i++) {
            cout << mas[i] << " ";
        }
        cout << "}\n";
    }

    friend class work;
};

class work {
private:
    int SumElements = 0;

public:
    work(vek ClassVek) {
        for (int i = 0; i < ClassVek.n - 1; i++) {
            SumElements += ClassVek.mas[i];
        }
    }

    ~work() { }

    void PrintSumElements()  {
        cout << "Сумма элементов class vek = " << SumElements << "\n";
    }
};

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

    vek ClassVek = vek();
    ClassVek.show();

    work ClassWork = work(ClassVek);
    ClassWork.PrintSumElements();
}
```