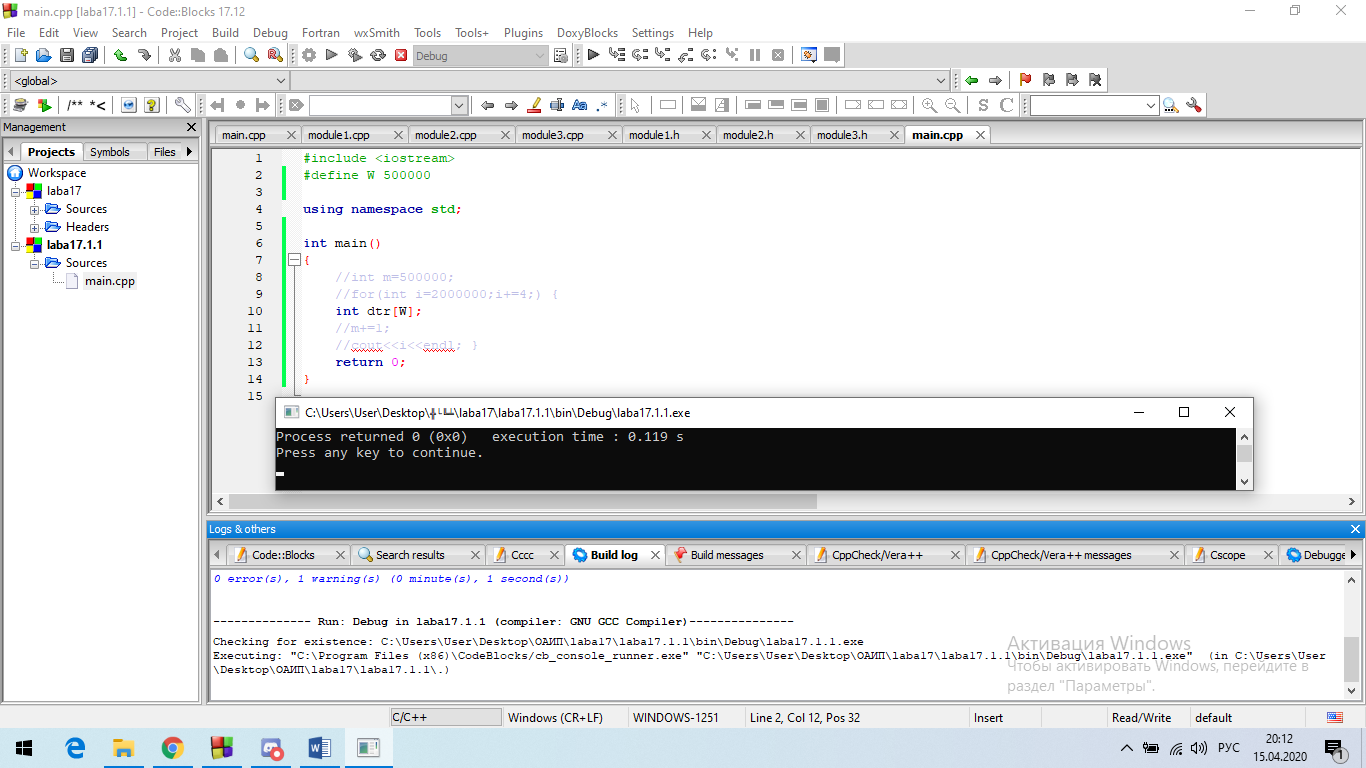
**Головачев Д.А. Y2231 Лабораторная работа номер 17.**

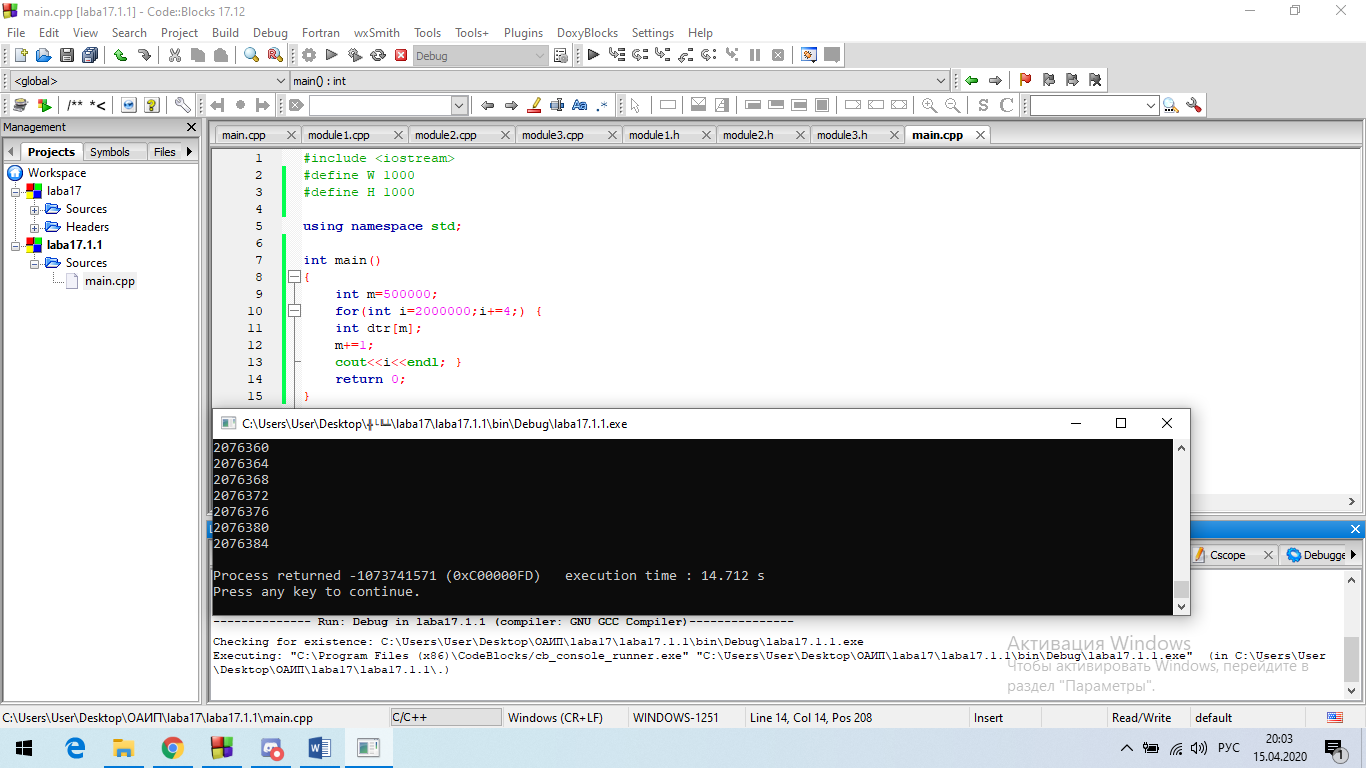
**Задание 1.1.**

Быстрее всего у моего ноутбука получилось переполнить стек за 0.119 секунды, я вызвал это тем, что создал массив на миллион элементов. Память под массив выделялась в стеке, он переполнился, программа выдала ошибку.



Задание 1.**2.**

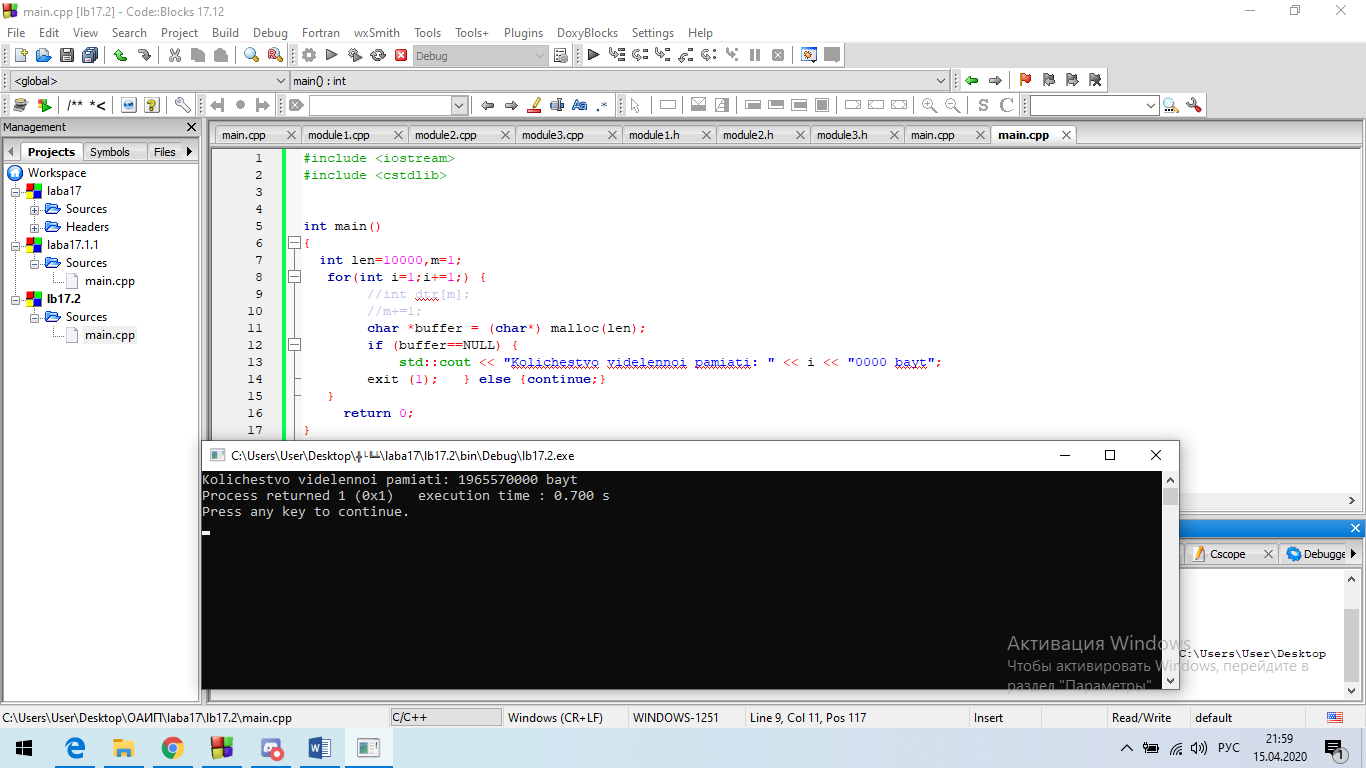
Самое точное переполнение: как видно в коде, я сделал бесконечный цикл, которые пересоздавал массив из предыдущего примера и на каждой новой итерации число элементов увеличивалось на 1, помять под 1 элемент выделялась в 4 байта, поэтому я сделал счетчик, который начинался с 2 000 000 байт (500 000 элементов) и с каждым новым элементом увеличивался на 4. Таким образом у меня получилось заполнить стек максимально приближенно в выделенной памяти: 2 076 384 байта ~ 2мб.



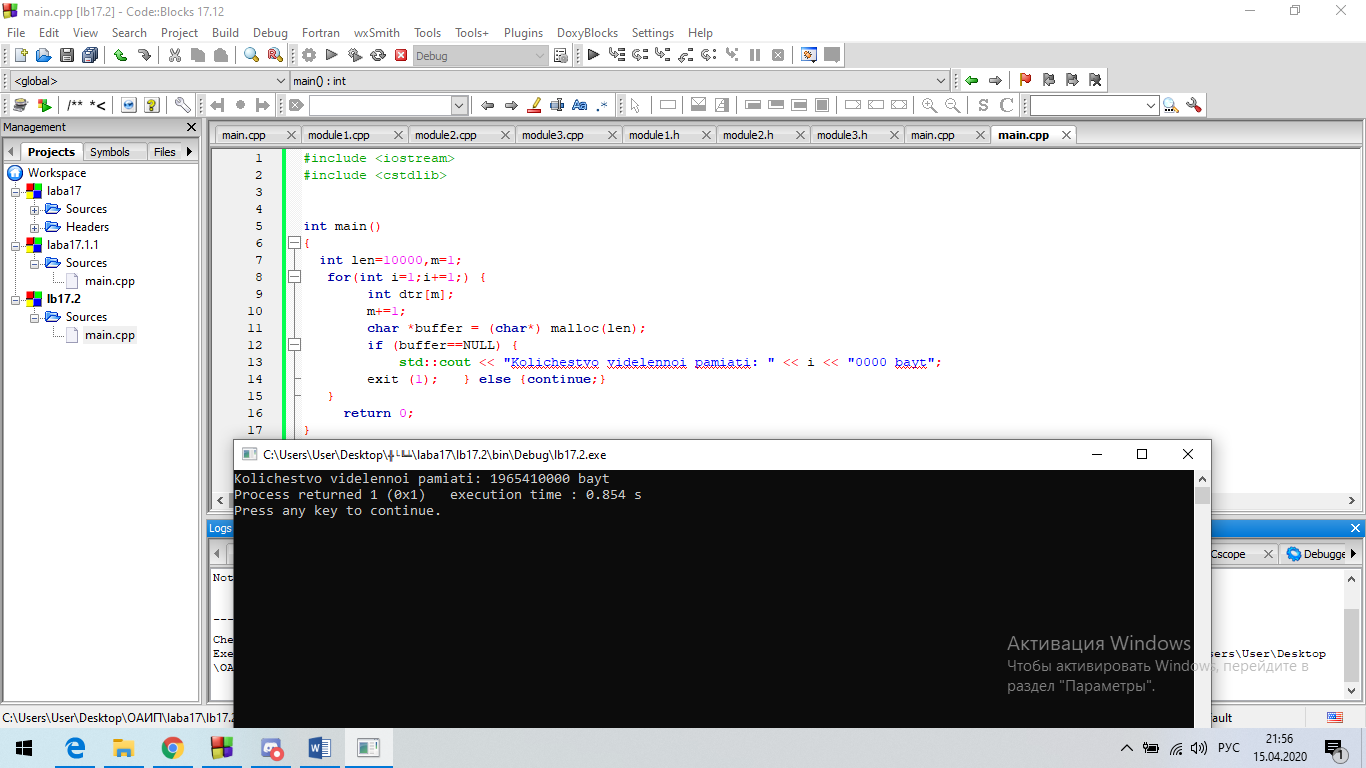
Вывод: когда стек переполняется, программа возвращает случайное значение и выдает ошибку. При создании массива с названием, которое уже принадлежит другому массиву, старый массив исчезает и на его место встает новый.

**Задание 2.**

На скриншоте куча заполнена на 1 965 570 000 байтах.



На втором скриншоте куча заполнена на 1 965 410 000 байт.



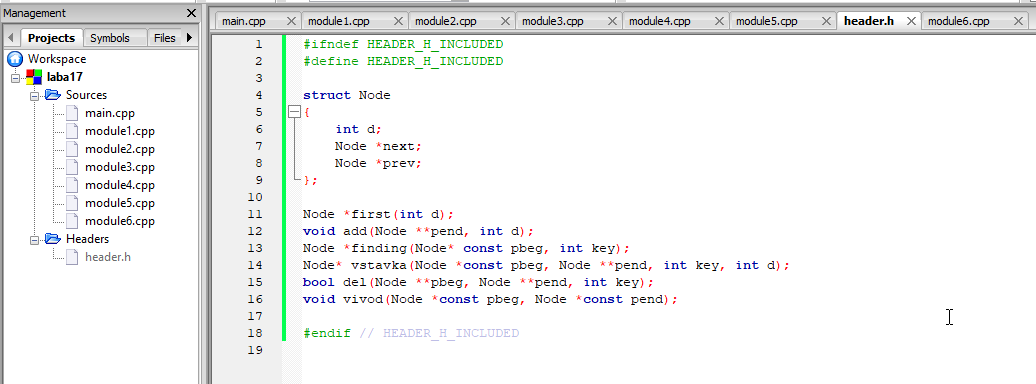
И на первом, и на втором скриншотах погрешность 10 000 байт, так был проведен эксперимент, как сильно заполнение стека повлияет на заполнение кучи. Как видно по скриншотам, там, где стек заполнялся, памяти в кучу было выделено немного меньше, значит они связаны (как и говорилось на лекции)

**Вывод:** куча занимает чуть меньше 2гб памяти, она почти в 1000 раз больше стека (интернет это подтверждает). Если выделять место под одну и ту же переменную, но не обнулять это место потом, то куча переполнится, причем переполнится мусором, к которому не будет доступа, ведь никаких адресов мы не оставляем.

**Задание 3.**

Объединение кодов из лекции, добавление меню.

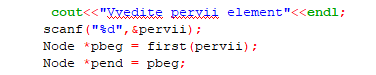
Я сделал 1 головной файл для всех шести модулей, структуру тоже в нем создал.



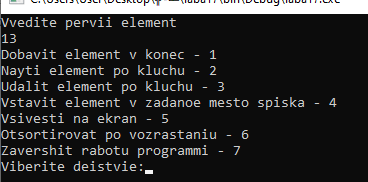
Коды в модулях 1-5 точно такие же, как в лекции, так что скриншотов с ними в этом отчете не будет.

Но зато будут скриншоты того, как передаются в них значения из main:

**1.Ввод первого элемента и то, что выдает программа в самом начале.**

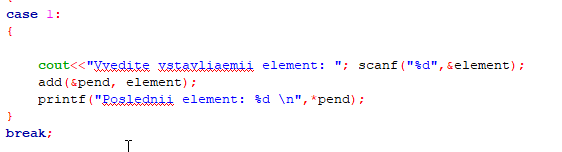
https://skr.sh/i/190420/QpFlDoXF.png?download=1

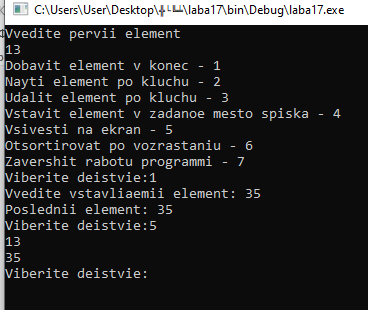
**2.Выдается после ввода первого элемента:**



Как видно по скриншоту, после ввода 1-ого элемента у пользователя есть 7 вариантов продолжения (один из этих вариантов, сортировка, из 4-ого задания).

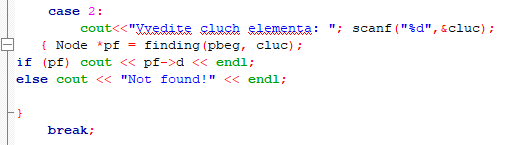
**3.Добавление элемента в конец списка.**

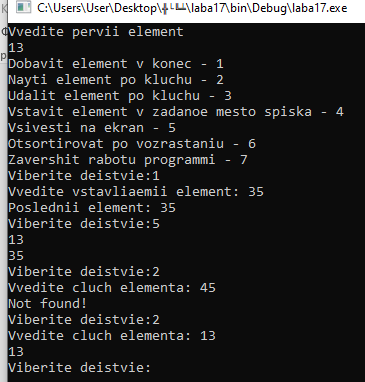




Тут я сначала выбираю, что хочу добавить элемент в конец списка, потом ввожу сам элемент, программа выдает, какой теперь последний элемент, и я ввожу запрос на показ всех элементов.

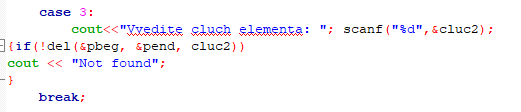
**4.Найти элемент по ключу.**

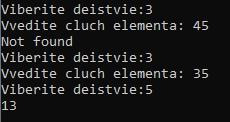




Я ввел запрос на поиск элемента, потом попытался найти элемент 45, но его не существует, после этого я повторил процедуру, но искал уже 13, и нашел.

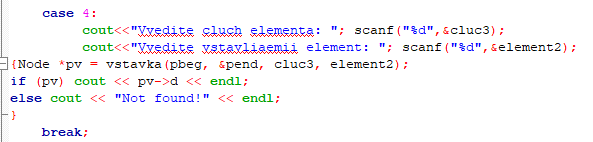
**4.Удаление элемента по ключу.**

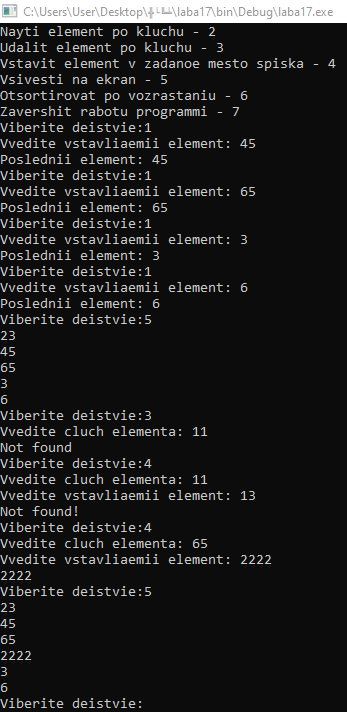




Я попытался удалить 45, не получилось (его нет), потом попытался удалить 35, получилось, потом запрос на вывод всех элементов.

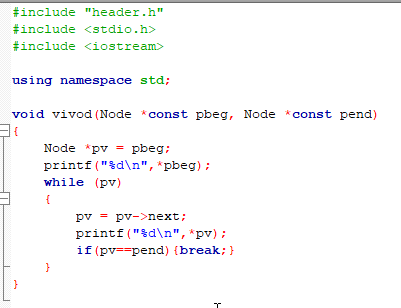
**5.Вставка элемента в заданное место списка.**

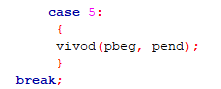
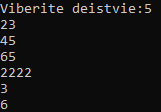




Я создал 5 элементов, после этого перепутал 3 и 4 на клавиатуре, потом все-таки послал запрос на добавление элемента после элемента 11, не получилось, потом послал запрос на добавление элемента после элемента 65, получилось, добавил элемент 2222, вывел все элементы на экран.

**6.Вывод на экран. (уже несколько раз демонстрировался)**





**Задание 4. (Задание 3 + сортировка)**

Код модуля сортировки:

#include "header.h"

#include <stdio.h>

#include <iostream>

using namespace std;

void sort(Node\* const pbeg, Node\* const pend)

{

int N=1;

Node \*p = pbeg;

while(p!=pend) {

p=p->next;

N+=1;

}

cout << endl;

cout<<N<<endl;

cout << endl;

Node \*p1 = pbeg;

Node \*p2 = pbeg;

int temp;

for (int i = 0; i <N; i++)

{

for (int j = i; j < N; j++)

{

if ((p1->d) >= (p2->d))

{

temp = p1->d;

p1->d = p2->d;

p2->d = temp;

}

p2 = p2-> next;

}

p1 = p1->next;

p2 = p1;

}

delete p2;

p1 = pbeg;

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

cout << (\*p1).d << endl;

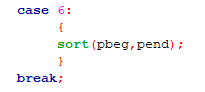
p1 = p1->next;

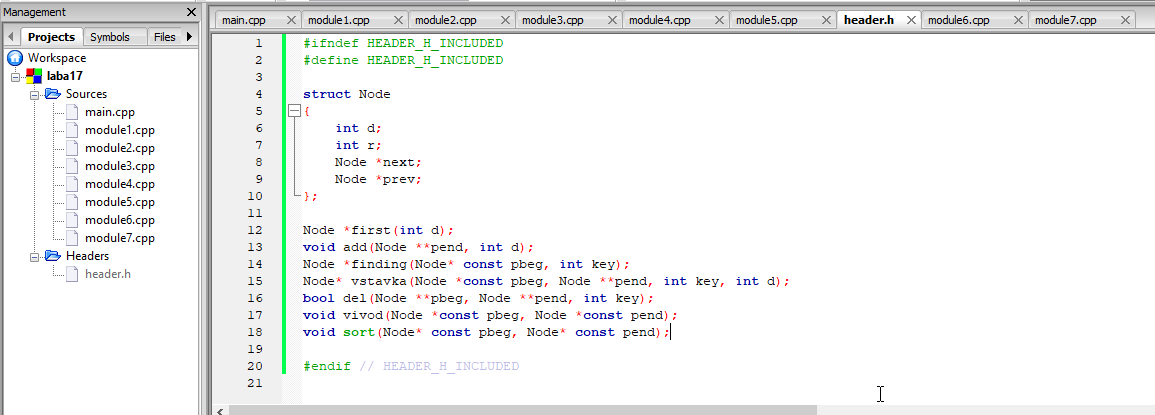
}

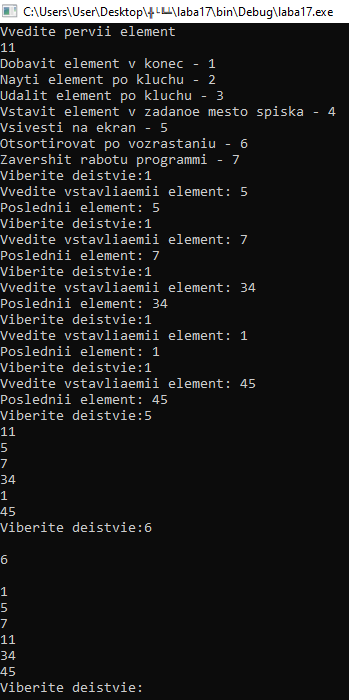
delete p1;

}

На скриншот код модуля сортировки не влез.







Я создал несколько элементов, после этого я вывел их на экран, потом я пустил запрос на сортировку, и программа вывела количество элементов и сами элементы в порядке возрастания.

**Задание 5.**

**Способ номер 1:** то, что я делал во втором задании (создание динамической переменной с именем, которое уже принадлежало другой динамической переменной).

**Способ номер 2:** у нас сесть список из нескольких элементов, каждый элемент ссылается на следующий элемент, а мы меняем ссылку от первого ко второму на от первого к третьему и у нас получается, что на второй элемент больше ничего не ссылается, мы не можем им пользоваться, он мусор.

**Способ номер 3:** мы в модуле/функции создали динамическую переменную и не отчистили её, после использования функции и до завершения работы программы там будет мусор.