МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационные системы»

**Лабораторная работа №1**по дисциплине «Основы алгоритмизации и программирования»

Выполнил:  
студент(ка) гр. ПИбд-12 Дозорова Алена

Ульяновск  
2021

Задание на лабораторную работу:

1. Реализовать АТД на структуре данных заданной по варианту:

Вариант 6: реализовать дек на базе односвязного списка

1. 2.Реализовать 2 сортировки по варианту:

Вариант 6: пузырьковая и быстрая сортировка

Краткое описание используемых алгоритмов и структур

В программе используется класс Deque, в котором определена структура узла для списка и функции, для работы с деком: проверка на пустоту, добавление в начало, добавление в конец, возвращение первого элемента, возвращение последнего элемента, возвращение размера дека, элемента по индексу и проверка на наличие элемента.

Для реализации сортировок используется массив, который случайным образом заполняется в цикле, затем в цикле происходит сортировка: перестановка местами элементов в соответствии с алгоритмом (попарная перестановка в пузырьковой и относительно центрального элемента в быстрой).

Код лабораторной работы:

Сортировки:

int my\_bubble\_sort(int mas [], int n)

{

int time = clock();

bool sort;

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

{

sort = true;

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

{

if (mas[i] > mas[j])

{

int t = mas[i];

mas[i] = mas[j];

mas[j] = t;

sort = false;

}

}

if (sort)

{

break;

}

}

return clock() / 1000.0;

}

void my\_q\_sort(int mas[], int n) {

int begin = 0;

int end = n - 1;

int center = mas[n / 2];

do {

while (mas[begin] < center) {

begin++;

}

while (mas[end] > center) {

end--;

}

if (begin <= end) {

int tmp = mas[begin];

mas[begin] = mas[end];

mas[end] = tmp;

begin++;

end--;

}

} while (begin <= end);

if (end > 0) {

my\_q\_sort(mas, end + 1);

}

if (begin < n) {

my\_q\_sort(&mas[begin], n - begin);

}

}

Класс Deque

class Deque

{

private:

Node\* head;

public:

Deque()

{

head = NULL;

}

void Push\_back(int d)

{

Node\* nd = new Node;

nd->data = d;

nd->next = NULL;

if (head == NULL)

head = nd;

else

{

Node\* current = head;

while (current->next != NULL)

current = current->next;

current->next = nd;

}

}

void Push\_front(int d)

{

Node\* nd = new Node;

nd->data = d;

nd->next = NULL;

if (head == NULL)

head = nd;

else

{

nd->next = head;

head = nd;

}

}

int front()

{

return head->data;

}

int back()

{

Node\* current = head;

while (current->next != NULL)

current = current->next;

return current->data;

}

int At(int index)

{

int i = 0;

Node\* current = head;

while (i != index && current->next != NULL)

{

current = current->next;

i++;

}

return current->data;

}

int Size()

{

int i = 1;

Node\* current = head;

while (current->next != NULL)

{

i++;

current = current->next;

}

return i;

}

void Print()

{

Node\* current = head;

while (current != NULL)

{

cout << current->data << "->";

current = current->next;

}

cout << "NULL";

}

bool Empty()

{

if (head == NULL)

{

return true;

}

return false;

}

bool Has(int d)

{

Node\* current = head;

while (current != NULL)

{

if (current->data == d)

{

return true;

}

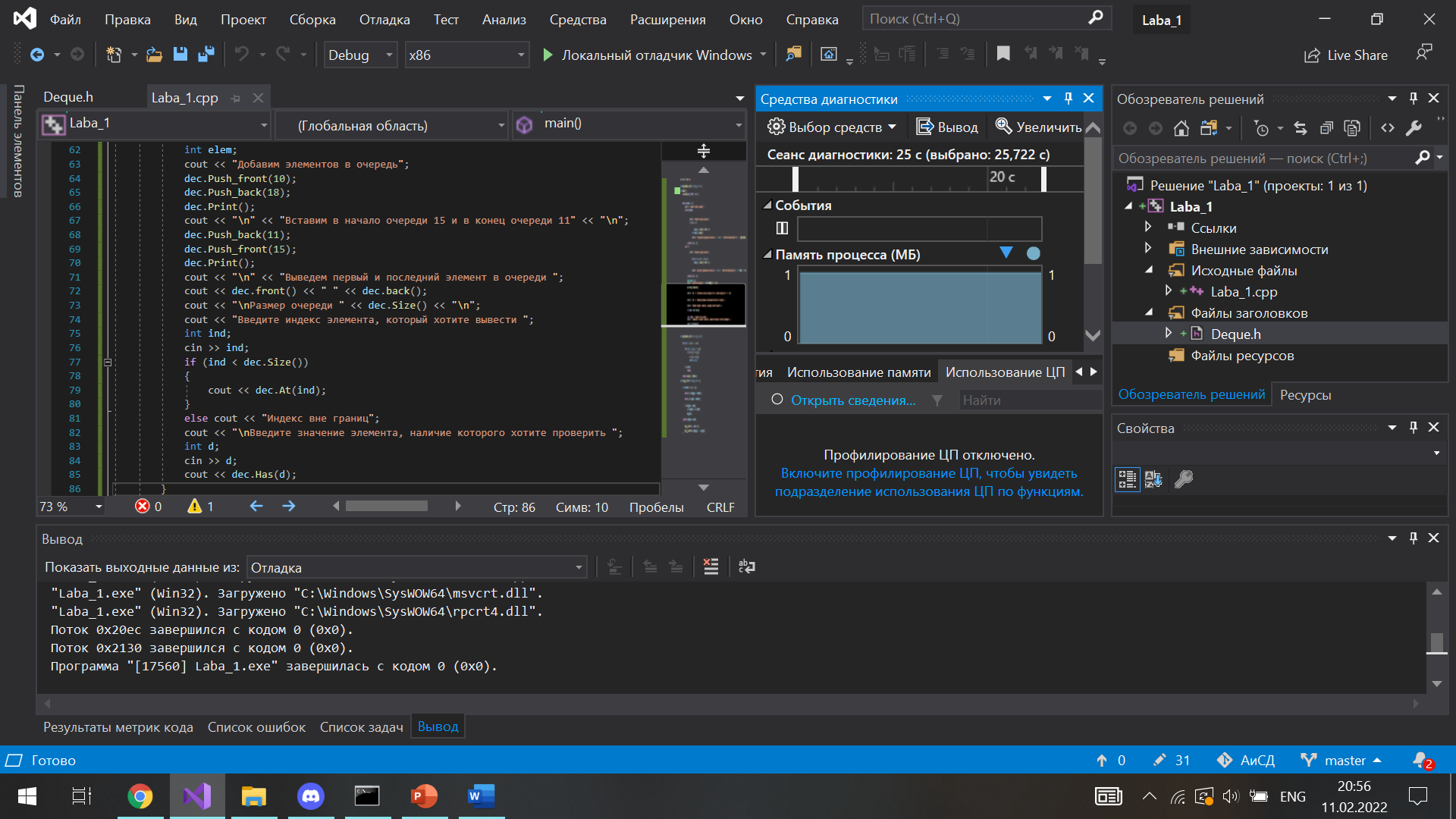
else current = current->next;

}

return false;

}

Результаты тестирования:



Выводы:

Алгоритмы сортировок были выполнены очень приближенно к эталонным, реализация дека оказалась более громоздкой, чем в коллекции.