Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им.И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление:710400 «Программная инженерия»

ОТЧЕТ

По дисциплине: «Алгоритмы и структуры данных»

Лабораторная работа №3

Тема: «Работа со статическим списком (на основе массива)»

Выполнил: студент группы

ПИ(б)-2-19 Улан уулу Нурдин

Проверила: Валеева А. А.

Бишкек – 2020

**Практическое задание**

1. **Дать определение структуры данных «Список».**

Список — базовая динамическая структура данных в информатике, состоящая из [узлов](https://ru.wikipedia.org/wiki/%D0%A3%D0%B7%D0%B5%D0%BB_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), каждый из которых содержит как собственно [данные](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), так и одну или две [ссылки](https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед [массивом](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

1. **Как классифицируются списки?**

**По количеству полей указателей** различают однонаправленный (односвязный) и двунаправленный (двусвязный) списки.

Связный список, содержащий только один указатель на следующий элемент, называется односвязным.

Связный список, содержащий два поля указателя – на следующий элемент и на предыдущий, называется двусвязным.

1. **Дать определение линейного списка.**

**Односвязный линейный список (ОЛС).**

Каждый узел ОЛС содержит 1 поле указателя на следующий узел. Поле указателя последнего узла содержит нулевое значение (указывает на NULL).

**Двусвязный линейный список (ДЛС).**

Каждый узел ДЛС содержит два поля указателей: на следующий и на предыдущий узел. Поле указателя на следующий узел последнего узла содержит нулевое значение (указывает на NULL). Поле указателя на предыдущий узел первого узла (корня списка) также содержит нулевое значение (указывает на NULL).

1. **Какие основные операции допустимы к спискам?**

Основными операциями над списками являются:

* переход к очередному элементу списка;
* добавление в список нового элемента;
* поиск заданного элемента;
* удаление элемента из списка.

1. **Какой тип данных применяется для описания статического списка в С++?**

Для описания статического списка в С++ используют массив.

1. **Какие описания необходимы для статической реализации списка?**

При использовании списка мы определяем тип LIST как запись, имеющую два поля. Первое поле elements (элементы) — это элементы списка, чей размер считается достаточным для хранения списков любой длины, встречающихся в данной реализации или программе. Второе поле целочисленного типа last (последний) указывает на позицию последнего элемента списка.

1. **Как осуществляется доступ к элементам списка?**

Доступ к элементам списка осуществляется через указатель и выполняются только в конце списка. Список можно расценивать как стопку книг, для того чтобы взять самую нижнюю книгу, нужно убрать сверху, в списке верх, это конец.

1. **Как реализуется операция добавления элемента в список?**

Как и говорилось раннее, список можно расценивать как стопку книг, и чтобы добавить еще одну книгу, нужно для начала убрать все книги, стоящие сверху, если добавить нужно в середину.

1. **Как реализуется операция удаления элемента из списка?**

Для операции удаления все так же, только место добавления, и сдвига всех элементов на право, мы убираем один и сдвигаем все на лево.

1. **Как реализовать поиск элемента в списке?**

Поиск заданного элемента реализуется как обычный поиск в массиве.

1. **Как выполняется проход по статическому списку?**

Проход по статистическому списку выполняется так же, как и в массиве, с помощью цикла.

1. **Какие особые ситуации возможны при статической реализации списка?**

Операции перемещения ячеек массива могут стать трудоемкими, если каждый элемент списка, размещенный в одной ячейке массива представляет собой запись с большим числом объемных полей.

1. **Что такое пустой статический список?**

Список, в котором нет ни 1 элемента, называется пустым статическим списком.

1. **Как в простейшем случае можно отслеживать свободные ячейки массива при реализации статического списка?**

Проверить через счетчик кол-во элементов списка, и сравнить с размером списка.

1. **В чем состоят преимущества и недостатки простейшего способа реализации списков с помощью массивов?**

Преимуществом реализации списков с помощью массива является легкость проверки наличия элемента, поиска и прохода по элементам. Недостатком является трудоемкость перемещения ячеек.

1. **За счет чего можно повысить эффективность простейшей реализации списка с помощью массива?**

Эффективность можно повысить при использовании указателей.

**II. Выполнить упражнения**

**1. Постановка задачи**

Разработать программу моделирования статического линейного списка (на основе массива) с функциями -

* инициализации статического списка;
* добавления элемента в статический список-
* по позиции;
* удаления элемента из статического списка –
* по позиции;
* поиск элемента в статическом списке –
* по позиции;
* по значению;
* вывод элементов статического списка;
* удаление списка.
* добавления элемента в статический список по значению;
* добавление нового элемента после заданного значения;
* добавление нового элемента перед заданным элементом;
* удаления элемента из статического списка по значению;
* удаление элемента после заданного элемента;
* удаление элемента перед заданным элементом;
* нахождение позиции минимального элемента списка;
* упорядочивание списка.

**2. Описание входных и выходных данных**

Структура **List** с параметрами: size, \*mas

Size – размер массива

\* mas – сам массив

\* accept – переменная для принятия адреса искомого элемента

Number – переменная для выбора в меню

A - переменная для выбора в главном меню

X – переменная для хранения значений, например индекса или самого элемента

X1 – переменная для хранения индекса

Q – переменная для проверки кол-во удаленных элементов

K – переменная для проверки, чтобы красиво вывести

Check – переменная для бесконечного цикла в главного меню

Check1 – переменная для бесконечного цикла в вторичного меню

Check2 – переменная для бесконечного цикла при вводе значений

i j – переменные для счетчика цикла

**3. Описание алгоритма**

Сначала идет ввод кол-во элементов массива, далее, пользователю предоставляется выбор: **Ввести элементы**

**Вывести элементы**

**Добавить элементы**

**Удалить элементы**

**Найти элементы**

**Отсортировать элементы**

**Удалить элементы**

**Выход**

Выбрав “Ввести элементы” пользователь войдет в функцию **void initialization(int\* arr, int n),** где он может инициализовать элементы массива.

Выбрав “Вывести элементы” пользователь войдет в функцию **void output(int\* arr, int n),** где он может вывести на экран элементы массива.

Выбрав “Добавить элементы”, пользователь войдет в **case 3,** где ему предоставляется выбор, как добавить элементы: по индексу, до/после заданного элемента, и по значению.

Выбрав “по индексу” пользователь войдет в функцию **int\* add(int\* arr, int n),** где он добавит элемент массива в указанный индекс.

Выбрав “до/после заданного элемента” пользователь войдет в функцию **int\* add\_bef\_aft(int\* arr, int n, int num),** где он добавит элемент массива в до или после заданного элемента.

Выбрав “по значению” пользователь войдет в функцию **int\* add\_bef\_aft(int\* arr, int n),** где он добавит элемент в отсортированный массив.

Выбрав “Удалить элементы”, пользователь войдет в **case 4,** где ему предоставляется выбор, как удалить элементы: по индексу, до или после заданного элемента, и по значению.

Выбрав “по индексу” пользователь войдет в функцию **int\* del(int\* arr, int n1),** где он удалит элемент массива в указанном индексе.

Выбрав “до/после заданного элемента” пользователь войдет в функцию **int\* del\_bef\_aft(int\* arr, int n, int num),** где он удалит элемент массива в до или после заданного элемента.

Выбрав “по значению” пользователь войдет в функцию **int\* del\_znam(int\* arr, int n, int x),** где он удалит заданный элемент в массиве.

Выбрав “Найти элементы”, пользователь войдет в **case 5,** где ему предоставляется выбор, как найти элементы: по индексу, по значению или минимальный в массиве.

Выбрав “по индексу” пользователь войдет в функцию **int\* search(int\* arr, int n, int number),** где он найдет элемент массива в указанном индексе.

Выбрав “по значению” пользователь войдет в функцию **int\* search(int\* arr, int n, int number),** где он найдет первый заданный элемент в массиве.

Выбрав “минимальный в массиве” пользователь войдет в функцию **int\* search\_min(int\* arr, int n),** где он найдет минимальный элемент в массиве.

Выбрав “Отсортировать элементы” пользователь войдет в функцию **void sort(int\* arr, int n),** где элементы массива от сортируются методом пузырька.

Выбрав “Отсортировать элементы” пользователь войдет в функцию **void sort(int\* arr, int n),** где элементы массива от сортируются методом пузырька.

Выбрав “Удалить элементы” пользователь войдет в функцию **void DelMas(int\* arr),** где элементы массива удаляются.

Выбрав “Выход”, пользователь выйдет из программы.

**int main()**

Начало

Cin >>Matrix.size

Matrix.size <= 0

True

False

number = 0, a, x = 0, x1 = 0, q = 0, k = 0;

Check = 0;

Cin >>Matrix.size

Check == 0

Конец

Check1 = 0, Check2 = 0;

Cin >>a

switch (a)

Break

initialization(Matrix.mas, Matrix.size)

case 1

case 2

Break

output(Matrix.mas, Matrix.size)

case 3

Check1 == 0

Break

Cin >>number

switch (number)

case 1

Check1++;

Matrix.mas = add(Matrix.mas, Matrix.size)

Break

Matrix.size++

case 2

Check1++;

Matrix.mas = add\_bef\_aft (Matrix.mas, Matrix.size)

Break

Matrix.size++

case 3

Check1++;

Matrix.mas = add\_bef\_aft (Matrix.mas, Matrix.size)

Break

Matrix.size++

case 4

Check1++;

Matrix.mas = add\_bef\_aft(Matrix.mas, Matrix.size)

Break

Matrix.size++

case 4

Check1 == 0

Break

Cin >>number

switch (number)

case 1

Check1++;

Matrix.mas = del(Matrix.mas, Matrix.size)

Break

Matrix.size--

case 2

Check1++;

Matrix.mas = del\_bef\_aft(Matrix.mas, Matrix.size, number)

Break

Matrix.size--

case 3

Check1++;

Matrix.mas = del\_bef\_aft(Matrix.mas, Matrix.size, number)

Break

Matrix.size--

case 4

Cin >>x

for (i = 0; i < Matrix.size; i++)

if (Matrix.mas[i] == x)

q++

Matrix.mas = del\_znam(Matrix.mas, Matrix.size, x);

for (i = 0; i < q; i++)

Break

Matrix.size--

case 5

Check1 == 0

Break

Cin >>number

switch (number)

case 1

Check1++;

accept = search(Matrix.mas, Matrix.size, number);

Cout << accept

Break

case 2

Check1++;

accept = search(Matrix.mas, Matrix.size, number);

Cout << accept

Break

case 3

Check1++;

accept = search\_min(Matrix.mas, Matrix.size, number);

case 6

Break

Cout << accept

Break

sort(Matrix.mas, Matrix.size)

case 7

Break

DelMas(Matrix.mas, Matrix.size)

case 8

Break

Check++

**void initialization(int\* arr, int n)**

Начало

Конец

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

cin >> arr[i];

**void output(int\* arr, int n)**

Начало

Конец

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

cout >> arr[i];

**int\* add(int\* arr, int n)**

Начало

n++

Check2 == 0

Delete arr

Return arr2

Cin >> x

x <= 0 && x > (n + 1)

False

True

Cout<<”Ошибка!”

Check2++  
j=0;

Cin >> a

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

False

i != (x - 1)

True

arr2[i] = a

arr2[i] = arr[j++]

**int\* add\_bef\_aft(int\* arr, int n, int num)**

Начало

a = 0, k = 0, x = 0, x1 = 0, j = 0

True

False

Num == 2

N++

Cin >> a

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

i != (n - 1)

False

True

arr2[i] = arr[j++]

arr2[i] = a

for (j = 0; j < n; j++)

for (i = 0; i < n - 1; i++)

True

False

arr2[i] > arr2[i + 1]

swap(arr2[i], arr2[i + 1])

False

True

Num == 3

Check2 == 0

Cin >> x

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

arr[i] == x

True

False

x1 = (i + 1)

K++

False

k > (n - 1)

True

Check2++

N++

Cout<<”Ошибка!”

Cin >> a

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

False

i != x1

True

arr2[i] = a

arr2[i] = arr[j++]

Check2 == 0

Cin >> x

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

True

False

arr[i] == x

x1 = (i - 1)

K++

k > (n - 1)

True

False

Check2++

N++

Cout<<”Ошибка!”

Cin >> a

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

True

False

i != x1

arr2[i] = arr[j++]

arr2[i] = a

for (j = 0; j < n; j++)

for (i = 0; i < n - 1; i++)

True

False

arr2[i] > arr2[i + 1]

swap(arr2[i], arr2[i + 1])

Конец

**int\* del(int\* arr, int n)**

Начало

n++

Check2 == 0

Delete arr

Return arr2

Cin >> x

x <= 0 && x > (n + 1)

False

True

Arr[x-1] = 0

Cout << “Ошибка”

for (j = 0; j < n; j++)

for (i = 0; i < n - 1; i++)

True

False

Arr[i] == 0

Swap(arr[i], arr[i + 1])

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

arr2[i] = arr[i]

**int\* del\_bef\_aft(int\* arr, int n, int num)**

Начало

True

False

Num == 2

N--

Check2 == 0

Cin >> x

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

arr[i] == x

True

False

x1 = (i + 1)

K++

False

k > (n - 1)

True

Check2++

K=0

Cout<<”Ошибка”

Arr[x1] = 0

for (j = 0; j < n; j++)

for (i = 0; i < n - 1; i++)

False

True

Arr[i] == 0

Swap(arr[i], arr[i + 1])

True

False

Num == 2

N--

Check2 == 0

Cin >> x

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

False

True

arr[i] == x

K++

x1 = (i - 1)

False

k > (n - 1)

True

Check2++

K=0

Cout<<”Ошибка”

Arr[x1] = 0

for (j = 0; j < n; j++)

for (i = 0; i < n - 1; i++)

False

True

Arr[i] == 0

Swap(arr[i], arr[i + 1])

Delete arr

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

Return arr2

arr2[i] = arr[i]

**int\* del\_znam(int\* arr, int n, int x)**

Начало

Check2 == 0

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

arr[i] == x

True

False

k++

arr[i] = 0

n--

False

True

k-(n-1)

Check2++

Cout<<”Ошибка”

for (j = 0; j < n; j++)

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

False

True

Arr[i] == 0

Swap(arr[i], arr[i + 1])

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

Delete arr

Return arr2

arr2[i] = arr[i]

**int\* search(int\* arr, int n, int number)**

Начало

Num == 1

Check2 == 0

Cin >> x

Return&arr[i]

(x - 1) >= n && (x - 1) < 0

False

True

Check2++

Cout << “Ошибка”

Cout << arr[i], i

Check2 == 0

Конец

Cin >> x

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

arr[i] == x

False

True

K++

X1=i

Cout << arr[i], I, &arr[i]

False

True

k-(n-1)

Check2++  
k=0

Cout << “Ошибка”

Return&arr[i]

**int\* search\_min(int\* arr, int n)**

Начало

Min = arr[0]

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

Cout <<min, imin

Arr[i] <min

False

True

Min = arr[i]  
imin = i

Return&arr[imin]

**void sort(int\* arr, int n)**

Начало

for (j = 0; j < n; j++)

for (i = 0; i < n - 1; i++)

Конец

False

True

arr[i] > arr[i + 1]

swap(arr[i], arr[i + 1])

**void DelMas(int\* arr)**

Начало

Delete arr

Конец

**Декомпозиция**

Начало

Delete list()

Sort elements()

Addition elements()

Search elements()

Delete elements()

Del by value elements()

Del by index elements()

Add by index elements()

Search min element()

Add by value elements()

Add to specified elements()

Search by index elements()

Search by value elements()

Add after the specified

elements()

Del to specified elements()

Del after the specified

elements()

**Текст программы**

#include <iostream>

#include "Windows.h"

using namespace std;

struct list {

int size;

int\* mas;

};

int i, j;

int Check1, Check2, Check = 0;

void initialization(int\* arr, int n);

void output(int\* arr, int n);

int\* add(int\* arr, int n);

int\* add\_bef\_aft(int\* arr, int n, int num);

int\* del(int\* arr, int n);

int\* del\_bef\_aft(int\* arr, int n, int num);

int\* del\_znam(int\* arr, int n, int x);

int\* search(int\* arr, int n, int number);

int\* search\_min(int\* arr, int n);

void sort(int\* arr, int n);

void DelMas(int\* arr);

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

list Matrix;

cout << "Введите размер массива: ";

cin >> Matrix.size;

system("cls");

if (Matrix.size <= 0) {

cout << "Размер массива не может быть <= 0 !!!" << endl;

}

else {

int\* accept = NULL;

int number = 0, a, x = 0, x1 = 0, q = 0, k = 0;

Matrix.mas = new int[Matrix.size];

while (Check == 0) {

Check1 = 0, Check2 = 0;

cout << "-----------------------------------------" << endl;

cout << "| Главное Меню: |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 1.Ввод данных |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 2.Вывод данных |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 3.Добавление элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 4.Удаление элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 5.Поиск элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 6.Упорядочить список |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 7.Удалить список |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 8.Выход. |" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

cout << "Введите номер операции:";

cin >> a;

system("cls");

switch (a) {

case 1: initialization(Matrix.mas, Matrix.size); break;

case 2: output(Matrix.mas, Matrix.size); break;

case 3: while (Check1 == 0) {

cout << "-----------------------------------------" << endl;

cout << "| Добавление элемента: |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 1.По позиции |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 2.По значению |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 3.После заданного элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 4.Перед заданным элементом |" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

cout << "Введите номер операции:";

cin >> number;

system("cls");

switch (number) {

case 1:

Check1++;

Matrix.mas = add(Matrix.mas, Matrix.size);

Matrix.size++; break;

case 2:

Check1++;

Matrix.mas = add\_bef\_aft(Matrix.mas, Matrix.size, number);

Matrix.size++; break;

case 3:

Check1++;

Matrix.mas = add\_bef\_aft(Matrix.mas, Matrix.size, number);

Matrix.size++; break;

case 4:

Check1++;

Matrix.mas = add\_bef\_aft(Matrix.mas, Matrix.size, number);

Matrix.size++; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}; break;

case 4: while (Check1 == 0) {

cout << "-----------------------------------------" << endl;

cout << "| Удаление элемента: |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 1.По позиции |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 2.После заданного элемента |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 3.Перед заданным элементом |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 4.По значению |" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

cout << "Введите номер операции:";

cin >> number;

system("cls");

switch (number) {

case 1:

Check1++;

Matrix.mas = del(Matrix.mas, Matrix.size);

Matrix.size--; break;

case 2:

Check1++;

Matrix.mas = del\_bef\_aft(Matrix.mas, Matrix.size, number);

Matrix.size--; break;

case 3:

Check1++;

Matrix.mas = del\_bef\_aft(Matrix.mas, Matrix.size, number);

Matrix.size--; break;

case 4:

Check1++;

cout << "Введите элемент который удалить:";

cin >> x;

system("cls");

for (i = 0; i < Matrix.size; i++) {

if (Matrix.mas[i] == x) {

q++;

}

}

Matrix.mas = del\_znam(Matrix.mas, Matrix.size, x);

for (i = 0; i < q; i++) {

Matrix.size--;

}; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}; break;

case 5: while (Check1 == 0) {

cout << "-----------------------------------------" << endl;

cout << "| Поиск элемента: |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 1.По позиции |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 2.По значению |" << endl;

cout << "-----------------------------------------" << endl;

cout << "| 3.Минимальный элемент |" << endl;

cout << "-----------------------------------------" << endl;

cout << endl;

cout << "Введите номер операции:";

cin >> number;

system("cls");

switch (number) {

case 1:

Check1++;

accept = search(Matrix.mas, Matrix.size, number);

cout << "И находиться по адресу " << accept << endl; break;

case 2:

Check1++;

accept = search(Matrix.mas, Matrix.size, number);

break;

case 3:

Check1++;

accept = search\_min(Matrix.mas, Matrix.size);

cout << "И находиться по адресу " << accept << endl; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}; break;

case 6: sort(Matrix.mas, Matrix.size); break;

case 7: DelMas(Matrix.mas); Matrix.size = 0; break;

case 8: Check++; break;

default: cout << "Такой операции не существует!!!" << endl << "Введите заново." << endl; break;

}

}

}

return 0;

}

void initialization(int\* arr, int n) {

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

cin >> arr[i];

}

}

void output(int\* arr, int n) {

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

cout << arr[i] << endl;

}

}

int\* add(int\* arr, int n) {

int x, a;

n++;

int\* arr2 = new int[n];

while (Check2 == 0) {

cout << "Введите индекс куда добавить:";

cin >> x;

system("cls");

if (x <= 0 && x > (n + 1)) {

cout << "Индекс не может быть <= 0 или > введенного размера в начале !!!" << endl;

}

else {

Check2++;

j = 0;

cout << "Введите элемент:";

cin >> a;

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

if (i != (x - 1)){

arr2[i] = arr[j++];

}

else {

arr2[i] = a;

}

}

}

}

delete arr;

return arr2;

}

int\* add\_bef\_aft(int\* arr, int n, int num){

int a = 0, k = 0, x = 0, x1 = 0;

int\* arr2 = new int[n + 1];

j = 0;

if (num == 2){

n++;

cout << "Введите элемент:";

cin >> a;

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

if (i != (n - 1)) {

arr2[i] = arr[j++];

}

else {

arr2[i] = a;

}

}

for (j = 0; j < n; j++) {

for (i = 0; i < n - 1; i++) {

if (arr2[i] > arr2[i + 1]) {

swap(arr2[i], arr2[i + 1]);

}

}

}

}

else if (num == 3){

while (Check2 == 0) {

cout << "Введите элемент после которого добавить:";

cin >> x;

system("cls");

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

if (arr[i] == x) {

x1 = (i + 1);

}

else{

k++;

}

}

if (k > (n - 1)) {

cout << "Такого элемента не существует" << endl;

}

else{

n++;

Check2++;

cout << "Введите элемент:";

cin >> a;

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

if (i != x1) {

arr2[i] = arr[j++];

}

else {

arr2[i] = a;

}

}

}

}

}

else {

while (Check2 == 0){

cout << "Введите элемент до которого добавить:";

cin >> x;

system("cls");

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

if (arr[i] == x) {

x1 = (i - 1);

}

else{

k++;

}

}

if (k > (n - 1)) {

cout << "Такого элемента не существует" << endl;

}

else{

n++;

Check2++;

cout << "Введите элемент:";

cin >> a;

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

if (i != x1) {

arr2[i] = arr[j++];

}

else {

arr2[i] = a;

}

}

for (j = 0; j < n; j++) {

for (i = 0; i < n - 1; i++) {

if (arr2[i] > arr2[i + 1]) {

swap(arr2[i], arr2[i + 1]);

}

}

}

}

}

arr2[0] = a;

}

delete arr;

return arr2;

}

int\* del(int\* arr, int n) {

int x;

n--;

int\* arr2 = new int[n];

while (Check2 == 0) {

cout << "Введите индекс откуда удалить:";

cin >> x;

system("cls");

if (x <= 0 && x > n) {

cout << "Индекс не может быть <= 0 или > введенного размера в начале !!!" << endl;

}

else {

Check2++;

arr[x - 1] = 0;

for (j = 0; j < n; j++) {

for (i = 0; i < (n - 1); i++) {

if (arr[i] == 0) {

swap(arr[i], arr[i + 1]);

}

}

}

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

arr2[i] = arr[i];

}

}

}

delete arr;

return arr2;

}

int\* del\_bef\_aft(int\* arr, int n, int num) {

int x1, x, k = 0;

if (num == 2) {

n--;

while (Check2 == 0) {

cout << "Введите элемент после которого удалить:";

cin >> x;

system("cls");

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

if (arr[i] == x) {

x1 = (i + 1);

}

else {

k++;

}

}

if (k > (n - 1)) {

cout << "Такого элемента не существует" << endl;

k = 0;

}

else

{

Check2++;

k = 0;

arr[x1] = 0;

for (j = 0; j < (n + 1); j++) {

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

if (arr[i] == 0) {

swap(arr[i], arr[i + 1]);

}

}

}

}

}

}

else if (num == 3) {

n--;

while (Check2 == 0) {

cout << "Введите элемент после которого удалить:";

cin >> x;

system("cls");

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

if (arr[i] == x) {

x1 = (i - 1);

}

else {

k++;

}

}

if (k > (n - 1)) {

cout << "Такого элемента не существует" << endl;

k = 0;

}

else{

Check2++;

k = 0;

arr[x1] = 0;

for (j = 0; j < (n + 1); j++) {

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

if (arr[i] == 0) {

swap(arr[i], arr[i + 1]);

}

}

}

}

}

}

int\* arr2 = new int[n];

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

arr2[i] = arr[i];

}

delete arr;

return arr2;

}

int\* del\_znam(int\* arr, int n, int x) {

int k = 0;

while (Check2 == 0) {

system("cls");

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

if (arr[i] == x) {

arr[i] = 0;

n--;

}

else {

k++;

}

}

if (k > (n - 1)) {

cout << "Такого элемента не существует" << endl;

k = 0;

}

else {

Check2++;

for (j = 0; j < n; j++) {

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

if (arr[i] == 0) {

swap(arr[i], arr[i + 1]);

}

}

}

}

}

int\* arr2 = new int[n];

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

arr2[i] = arr[i];

}

delete arr;

return arr2;

}

int\* search(int\* arr, int n, int number){

int x;

if (number == 1) {

while (Check2 == 0) {

cout << "Введите индекс элемента: ";

cin >> x;

system("cls");

if ((x - 1) >= n && (x - 1) < 0) {

cout << "Такого индекса нет !!!" << endl << "Введите заново." << endl;

}

else {

Check2++;

cout << "Элемент №:" << arr[x - 1] << " находиться на " << x << " индексе" << endl;

}

}

return&arr[x];

}

else {

int k = 0, x1 = 0;

while (Check2 == 0) {

cout << "Введите значение элемента: ";

cin >> x;

system("cls");

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

if (arr[i] == x) {

x1 = i;

cout << "Элемент №:" << arr[i] << " находиться на " << i + 1 << " индексе" << endl;

cout << "И находиться по адресу " << &arr[i] << endl;

}

else {

k++;

}

}

if (k > (n - 1)) {

cout << "Такого элемента нет !!!" << endl << "Введите заново." << endl;

k = 0;

}

else

{

Check2++;

k = 0;

return&arr[x1];

}

}

}

}

int\* search\_min(int\* arr, int n) {

int imin = 0, min = arr[0];

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

if (arr[i] < min) {

min = arr[i];

imin = i;

}

}

cout << "Элемент №:" << min << " находиться на " << imin + 1 << " индексе" << endl;

return&arr[imin];

}

void sort(int\* arr, int n) {

for (j = 0; j < n; j++) {

for (i = 0; i < n - 1; i++) {

if (arr[i] > arr[1 + i]) {

swap(arr[i], arr[i + 1]);

}

}

}

}

void DelMas(int\* arr) {

delete arr;

}