**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №2**

**по дисциплине «Программирование»**

Тема: "Линейные структуры данных"

|  |  |  |
| --- | --- | --- |
| Студент гр. 0324 |  | Гордиенко Т.Е. |
| Преподаватель |  | Глущенко А.Г |

Санкт-Петербург

2020

**Цели работы**: изучение свойств и организация динамических массивов и двусвязных списков; получение практических навыков в работе с динамическими массивами и двусвязными списками; проведение сравнительной характеристики скорости вставки, получения и удаления элементов из них.

Необходимо реализовать программу, которая выполняет следующие действия.

1.   Формирование целочисленного одномерного массива размерности *N*, где:

a) пользователь вводит количество элементов в массиве, который будет автоматически заполняться случайными числами (0 до 99);

б) пользователь вводит в консоль элементы массива, *N* определяется автоматически по количеству введенных элементов;

в) \* массив считывается с файла, *N* определяется как количество элементов массива в файле.

2.   Определение скорости создания динамического массива п. 1.

3.   Вставка, удаление и получение элемента массива. Удаление и получение элемента необходимо реализовать по индексу и по значению.

4.   Определение скорости вставки, удаления и получения элемента массива п. 3.

5.   Формирование двусвязного списка размерности *N*, где:

a) пользователь вводит количество элементов в списке, который будет автоматически заполняться случайными числами (0 до 99);

б) пользователь вводит в консоль элементы списка, *N* определяется автоматически по количеству введенных элементов;

в) \* список считывается с файла, *N* определяется как количество элементов списка в файле.

6.   Определение скорости создания двусвязного списка п. 5.

7.   Вставка, удаление и получение элемента двусвязного списка. Удаление и получение элемента необходимо реализовать по индексу и по значению.

8.   Определение скорости вставки, удаление и получения элемента двусвязного списка п. 7.

Должна быть возможность запуска каждого пункта многократно, если есть возможность (если в списке/массиве нет элементов, то нельзя ничего удалить и об этом нужно сообщить пользователю). Необходимо сравнить результаты. Для этого пункты 1–4 и 5–8 должны принимать одинаковые значения.

**#include <iostream>**

**#include <string>**

**#include <fstream>**

**#include <chrono>**

**int arr\_size = 0;//Переменная для хранения размера массива**

**struct list {//Структура двусвязного списка**

**list\* head;**

**int data;**

**list\* tail;**

**};**

**int print\_list(list\* begin) {//Вывод списка на экран**

**list\* curr = begin;**

**if (!begin) {//если передан пустой указатель**

**std::cout << "List is empty" << '\n';**

**}**

**else {**

**while (curr) {//пока указатель не будет пустым**

**std::cout << curr->data << " ";//выводим эелемент**

**curr = curr->tail;//переходим к следующему элементу**

**}**

**std::cout << '\n';**

**}**

**return 1;**

**}**

**int print\_array(int\* array) {//Вывод массива на экран**

**if (!array || arr\_size == 0) {//если указатель пустой иди размер массива равен нулю**

**std::cout << "Array is empty" << '\n';**

**}**

**else {**

**for (int i = 0; i != arr\_size; i++) {//проходим по элементам массива и выводим их**

**std::cout << array[i] << " ";**

**}**

**std::cout << '\n';**

**}**

**return 1;**

**}**

**int\* create\_rand\_array() {//массив рандомов**

**int\* arr = new int[arr\_size];//выделяем память под массив**

**for (int i = 0; i != arr\_size; i++) {**

**arr[i] = rand() % 100;//присваиваем значения**

**}**

**return arr;//возвращаем указатель на массив**

**}**

**int\* input\_array() {//ввод чисел в массив**

**int\* arr = new int[0];**

**int\* temp;**

**int n = 0, num;**

**std::cout << "Введите элементы списка через пробел (закончите ввод любой буквой):" << '\n';**

**std::cin >> num;//вводится значение**

**while (!std::cin.fail()) {//пока введёное значение можно отнести к int**

**temp = new int[n + 1];//выделяем память под новый массив, который на 1 элемент больше чем старый**

**for (int i = 0; i != n; i++) {//копируем значения из старого**

**temp[i] = arr[i];**

**}**

**temp[n] = num;//последнему элементу присваиваем введённое значение**

**delete[] arr;//удаляем старый массив**

**arr = temp;//переносим указатель**

**n++;//счётчик элементов**

**std::cin >> num;//вводим следующий элемент**

**}**

**std::cin.clear();//очищаем потом ввода**

**std::cin.ignore(32767, '\n');**

**arr\_size = n;//запоминаем размер массива**

**return arr;//возвращаем указатель на массив**

**}**

**int\* read\_array() {//чтение из файла в массив**

**std::string path;**

**std::cout << "Путь до файла:" << '\n';**

**getline(std::cin, path);//получаем путь до файла**

**std::ifstream file(path);**

**if (!file.is\_open()) {//если файл не открыт**

**std::cout << "Ошибка чтения" << '\n';**

**return 0;**

**}**

**int\* arr = new int[0];**

**int\* temp;**

**int n = 0, num;**

**while (!file.eof()) {//пока не дойдём до конца файла**

**temp = new int[n + 1];**

**for (int i = 0; i != n; i++) {**

**temp[i] = arr[i];**

**}**

**file >> temp[n];//заносим число в массив**

**delete[] arr;**

**arr = temp;**

**n++;**

**}**

**arr\_size = n;**

**file.close();//закрываем файл**

**return arr;**

**}**

**int\* add(int\* array, int item) {//добавление в массив**

**int\* temp = new int[arr\_size + 1];**

**for (int i = 0; i != arr\_size; i++) {**

**temp[i] = array[i];**

**}**

**temp[arr\_size] = item;**

**delete[] array;**

**array = temp;**

**arr\_size++;**

**return array;**

**}**

**int\* remove(int\* array, unsigned int posicion) {//удаление элемента по позиции**

**if (!array || arr\_size == 0) {//если указатель пустой или размер массива равен нулю**

**std::cout << "Array is empty" << '\n';**

**return array;**

**}**

**if (posicion > arr\_size || posicion == 0) {//если введённая позиция отсутствует**

**std::cout << "Out of range" << '\n';**

**return array;**

**}**

**int n = arr\_size;**

**int\* temp = new int[n - 1];**

**for (int i = 0; i != posicion - 1; i++) {//копируем**

**temp[i] = array[i];**

**}**

**for (int i = posicion - 1; i != n - 1; i++) {//пропускаем позицию и копируем следующие**

**temp[i] = array[i + 1];**

**}**

**delete[] array;//удаляем старый массив**

**std::cout << "Элемент удалён" << '\n';**

**arr\_size--;//уменьшаем размер на 1**

**array = temp;**

**return array;**

**}**

**int\* remove\_items(int\* array, int item) {//удаление эелемента по значению**

**if (!array) {**

**std::cout << "Array is empty" << '\n';**

**return array;**

**}**

**for (int i = 0; i != arr\_size; i++) {**

**if (array[i] == item) {//если нашелся элемент**

**array = remove(array, i + 1);//удаляем его**

**}**

**}**

**return array;**

**}**

**int take(int\* array, unsigned int posicion) {//возврат значения по позиции**

**if (posicion > arr\_size || posicion == 0) {**

**std::cout << "Out of range" << '\n';**

**return 0;**

**}**

**std::cout << array[posicion - 1] << '\n';**

**return array[posicion - 1];**

**}**

**int where\_item(int\* array, int item) {//возвращениее позиции по значению**

**for (int i = 0; i != arr\_size; i++) {**

**if (array[i] == item) {**

**std::cout << i + 1 << " ";**

**}**

**}**

**std::cout << '\n';**

**return 0;**

**}**

**list\* create\_rand\_list(unsigned int length) {//список из рандомов**

**if (length == 0) {**

**return 0;**

**}**

**list\* curr = 0;**

**list\* temp = 0;**

**list\* begin = 0;**

**temp = new list;**

**temp->head = NULL;**

**temp->tail = 0;**

**temp->data = rand() % 100;**

**curr = temp;**

**begin = temp;**

**for (int i = 1; i != length; i++) {**

**temp = new list;**

**temp->head = curr;**

**temp->tail = 0;**

**temp->data = rand() % 100;**

**curr->tail = temp;**

**curr = temp;**

**}**

**return begin;**

**}**

**list\* input\_list() {//ввод списка**

**list\* curr = 0;**

**list\* temp = 0;**

**list\* begin = 0;**

**int num;**

**std::cout << "Введите эелементы списка через пробел (закончите любой буквой):" << '\n';**

**std::cin >> num;**

**if (!std::cin.fail()) {**

**temp = new list;**

**temp->head = curr;**

**temp->tail = 0;**

**temp->data = num;**

**curr = temp;**

**begin = temp;**

**std::cin >> num;**

**}**

**while (!std::cin.fail()) {**

**temp = new list;**

**temp->head = curr;**

**temp->tail = 0;**

**temp->data = num;**

**curr->tail = temp;**

**curr = temp;**

**std::cin >> num;**

**}**

**std::cin.clear();**

**std::cin.ignore(32767, '\n');**

**return begin;**

**}**

**list\* read\_list() {//чтение из файла в список**

**list\* curr = 0;**

**list\* temp = 0;**

**list\* begin = 0;**

**std::string path;**

**std::cout << "Введите путь до файла:" << '\n';**

**getline(std::cin, path);**

**std::ifstream file(path);**

**if (!file.is\_open()) {**

**std::cout << "Ошибка чтения" << '\n';**

**return 0;**

**}**

**if (!file.eof()) {**

**temp = new list;**

**temp->head = curr;**

**temp->tail = 0;**

**file >> temp->data;**

**curr = temp;**

**begin = temp;**

**}**

**while (!file.eof()) {**

**temp = new list;**

**temp->head = curr;**

**temp->tail = 0;**

**file >> temp->data;**

**curr->tail = temp;**

**curr = temp;**

**}**

**return begin;**

**}**

**list\* add(list\* begin, int item) {//добавление к списку**

**list\* temp = begin;**

**if (temp == 0) {**

**temp = new list;**

**temp->head = 0;**

**temp->tail = 0;**

**temp->data = item;**

**begin = temp;**

**return begin;**

**}**

**else {**

**while (temp->tail) {**

**temp = temp->tail;**

**}**

**temp->tail = new list;**

**temp->tail->head = temp;**

**temp->tail->tail = 0;**

**temp->tail->data = item;**

**return begin;**

**}**

**}**

**list\* remove(list\* begin, unsigned int posicion) {//удаление по позиции**

**if (!begin) {**

**std::cout << "List is empty" << '\n';**

**return begin;**

**}**

**if (posicion == 0) {**

**return begin;**

**}**

**list\* curr = begin;**

**list\* temp;**

**int size = 0;**

**while (curr) {**

**curr = curr->tail;**

**size++;**

**}**

**if (posicion > size) {**

**std::cout << "Invalid posicion" << '\n';**

**return begin;**

**}**

**else if (size == 0) {**

**std::cout << "List is empty" << '\n';**

**return 0;**

**}**

**curr = begin;**

**for (int i = 1; i != posicion; i++) {**

**curr = curr->tail;**

**}**

**if (!curr->head && !curr->tail) {**

**begin = 0;**

**delete curr;**

**std::cout << "Элемент удалён" << '\n';**

**}**

**else if(!curr->head) {**

**temp = curr;**

**curr = curr->tail;**

**curr->head = 0;**

**begin = curr;**

**delete temp;**

**std::cout << "Элемент удалён" << '\n';**

**}**

**else if (!curr->tail) {**

**temp = curr;**

**curr = curr->head;**

**curr->tail = 0;**

**delete temp;**

**std::cout << "Элемент удалён" << '\n';**

**}**

**else {**

**temp = curr;**

**curr = curr->head;**

**curr->tail = temp->tail;**

**temp->tail->head = curr;**

**delete temp;**

**std::cout << "Элемент удалён" << '\n';**

**}**

**return begin;**

**}**

**list\* remove\_items(list\* begin, int item) {//удаление по значению**

**list\* curr = begin;**

**list\* temp;**

**while (curr->data == item) {**

**temp = curr;**

**curr = curr->tail;**

**remove(temp, 1);**

**begin = curr;**

**}**

**while (curr) {**

**if (curr->data == item) {**

**temp = curr;**

**curr = curr->tail;**

**remove(temp, 1);**

**}**

**else {**

**curr = curr->tail;**

**}**

**}**

**return begin;**

**}**

**int take(list\* begin, unsigned int posicion) {//возвращение значения по позиции**

**list\* curr = begin;**

**int size = 0;**

**while (curr) {**

**curr = curr->tail;**

**size++;**

**}**

**if (posicion > size || posicion == 0) {**

**std::cout << "Invalid posicion" << '\n';**

**return 0;**

**}**

**curr = begin;**

**for (int i = 1; i != posicion; i++) {**

**curr = curr->tail;**

**}**

**return curr->data;**

**}**

**list\* where\_item(list\* begin, int item) {//возвращение позиции по значению**

**list\* curr = begin;**

**int i = 0;**

**while (curr) {**

**i++;**

**if (curr->data == item) {**

**std::cout << i << " ";**

**}**

**curr = curr->tail;**

**}**

**std::cout << '\n';**

**return begin;**

**}**

**// Доп задание на удаление нечётных элементов ===============**

**int\* remove\_odd\_items(int\* arr) {**

**for (int i = 0; i != arr\_size; i++) {**

**if (arr[i] % 2 != 0) {**

**arr = remove(arr, i + 1);**

**i--;**

**}**

**}**

**return arr;**

**}**

**list\* remove\_odd\_items(list\* spisok) {**

**list\* curr = spisok;**

**list\* temp;**

**while (curr->data % 2 != 0) {**

**temp = curr;**

**curr = curr->tail;**

**remove(temp, 1);**

**spisok = curr;**

**}**

**while (curr) {**

**if (curr->data % 2 != 0) {**

**temp = curr;**

**curr = curr->tail;**

**remove(temp, 1);**

**}**

**else {**

**curr = curr->tail;**

**}**

**}**

**return spisok;**

**}**

**void chet\_pos(int\* arr, list\* spisok) {**

**std::chrono::system\_clock::time\_point start = std::chrono::system\_clock::now();**

**int chet = 0, nechet = 0;**

**for (int i = 0; i < arr\_size; i += 2) {**

**if (arr[i] % 2 == 0) {**

**chet++;**

**}**

**}**

**for (int i = 1; i < arr\_size; i += 2) {**

**if (arr[i] % 2 != 0) {**

**nechet++;**

**}**

**}**

**std::chrono::system\_clock::time\_point end = std::chrono::system\_clock::now();**

**std::chrono::duration <double> time = end - start;**

**std::cout << "Для массива" << '\n';**

**std::cout << "Чётных на четном месте: " << chet << '\n';**

**std::cout << "Нечётных на нечётном месте: " << nechet << '\n';**

**std::cout << "Время для массива: " << time.count() << '\n';**

**chet = 0;**

**nechet = 0;**

**start = std::chrono::system\_clock::now();**

**while (spisok) {**

**if (spisok->data % 2 == 0) {**

**chet++;**

**}**

**spisok = spisok->tail;**

**if (spisok) {**

**if (spisok->data % 2 != 0) {**

**nechet++;**

**}**

**spisok = spisok->tail;**

**}**

**}**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Для списка" << '\n';**

**std::cout << "Чётных на чётном месте: " << chet << '\n';**

**std::cout << "Нечётных на нечётном месте: " << nechet << '\n';**

**std::cout << "Время для списка: " << time.count() << '\n';**

**}**

**//===============================================================**

**int main() {**

**setlocale(0,"");**

**std::chrono::system\_clock::time\_point start;**

**std::chrono::system\_clock::time\_point end;**

**std::chrono::duration <double> time;**

**int choise = 1;**

**list\* spisok = 0,\* temp\_l;**

**int\* arr = 0;**

**unsigned int size = 0;**

**int num = 0;**

**while (choise != 0) {**

**std::cout << '\n' << "Выберите пункт:" << '\n';**

**std::cout << "1) Сосздать массив" << '\n';**

**std::cout << "2) Добавить элемент в массив" << '\n';**

**std::cout << "3) Удалить элемент из массива" << '\n';**

**std::cout << "4) Поиск элемента массива" << '\n' << '\n';**

**std::cout << "5) Создать двусвязный список" << '\n';**

**std::cout << "6) Добавить элемент в список" << '\n';**

**std::cout << "7) Удалить элемент списка" << '\n';**

**std::cout << "8) Поиск эелемента списка" << '\n' << '\n';**

**std::cout << "9) Вывод массива и списка" << '\n' << '\n';**

**std::cout << "10) Удалить нечётные числа (доп задание)" << '\n';**

**std::cout << "11) Проверить чётные и нечётные места" << '\n';**

**std::cout << "0) Выход" << '\n';**

**std::cin >> choise;**

**switch (choise) {**

**case 1://Создание масиива**

**choise = 0;**

**if (arr) {**

**delete[] arr;**

**arr = 0;**

**}**

**while (!choise) {**

**std::cout << "Выберите способ: " << '\n';**

**std::cout << "1) Массив N чисел от 0 до 100" << '\n';**

**std::cout << "2) Ввести элементы массива" << '\n';**

**std::cout << "3) Прочесть из файла" << '\n';**

**std::cin >> choise;**

**switch (choise) {**

**case 1:**

**std::cout << "Размер массива: ";**

**std::cin >> arr\_size;**

**start = std::chrono::system\_clock::now();**

**arr = create\_rand\_array();**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на создание массива: " << time.count() << '\n';**

**break;**

**case 2:**

**start = std::chrono::system\_clock::now();**

**arr = input\_array();**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на создание массива: " << time.count() << '\n';**

**break;**

**case 3:**

**start = std::chrono::system\_clock::now();**

**arr = read\_array();**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на создание массива: " << time.count() << '\n';**

**break;**

**default:**

**choise = 0;**

**std::cout << "Try again" << '\n';**

**break;**

**}**

**}**

**break;**

**case 2://Добавление в массив**

**std::cout << "Добавляемое число: ";**

**std::cin >> num;**

**start = std::chrono::system\_clock::now();**

**arr = add(arr, num);**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на добавление: " << time.count() << '\n';**

**break;**

**case 3://Удаление из массива**

**choise = 0;**

**while (!choise) {**

**std::cout << "1) по значению" << '\n';**

**std::cout << "2) по позиции" << '\n';**

**std::cin >> choise;**

**switch (choise) {**

**case 1:**

**std::cout << "Удаляемое число: ";**

**std::cin >> num;**

**start = std::chrono::system\_clock::now();**

**arr = remove\_items(arr, num);**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на удаление: " << time.count() << '\n';**

**break;**

**case 2:**

**std::cout << "Удаляемая позиция: ";**

**std::cin >> size;**

**start = std::chrono::system\_clock::now();**

**arr = remove(arr, size);**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на удаление: " << time.count() << '\n';**

**break;**

**default:**

**choise = 0;**

**std::cout << "Try again" << '\n';**

**break;**

**}**

**}**

**break;**

**case 4://возврат из массива**

**choise = 0;**

**while (!choise) {**

**std::cout << "1) по значению" << '\n';**

**std::cout << "2) по позиции" << '\n';**

**std::cin >> choise;**

**switch (choise) {**

**case 1:**

**std::cout << "Искомое число: ";**

**std::cin >> num;**

**start = std::chrono::system\_clock::now();**

**where\_item(arr, num);**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на поиск: " << time.count() << '\n';**

**break;**

**case 2:**

**std::cout << "Искомая позиция: ";**

**std::cin >> size;**

**start = std::chrono::system\_clock::now();**

**take(arr, size);**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на поиск: " << time.count() << '\n';**

**break;**

**default:**

**choise = 0;**

**std::cout << "try again" << '\n';**

**break;**

**}**

**}**

**break;**

**case 5://создание списка**

**choise = 0;**

**if (spisok) {**

**while (spisok) {**

**temp\_l = spisok;**

**spisok = spisok->tail;**

**delete temp\_l;**

**}**

**}**

**while (!choise) {**

**std::cout << "Выберите способ:" << '\n';**

**std::cout << "1) N рандомных чисел" << '\n';**

**std::cout << "2) Ввести числа" << '\n';**

**std::cout << "3) Прочитать из файла" << '\n';**

**std::cin >> choise;**

**switch (choise) {**

**case 1:**

**std::cout << "Размер списка: ";**

**std::cin >> size;**

**start = std::chrono::system\_clock::now();**

**spisok = create\_rand\_list(size);**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на создание списка: " << time.count() << '\n';**

**break;**

**case 2:**

**start = std::chrono::system\_clock::now();**

**spisok = input\_list();**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на создание списка: " << time.count() << '\n';**

**break;**

**case 3:**

**start = std::chrono::system\_clock::now();**

**spisok = read\_list();**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на создание списка: " << time.count() << '\n';**

**break;**

**default:**

**std::cout << "Try again" << '\n';**

**choise = 0;**

**break;**

**}**

**}**

**break;**

**case 6://добавление в список**

**std::cout << "Добавляемое число: ";**

**std::cin >> num;**

**start = std::chrono::system\_clock::now();**

**spisok = add(spisok, num);**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на добавление: " << time.count() << '\n';**

**break;**

**case 7://удаление из списка**

**choise = 0;**

**while (!choise) {**

**std::cout << "1) По значению" << '\n';**

**std::cout << "2) По позиции" << '\n';**

**std::cin >> choise;**

**switch (choise) {**

**case 1:**

**std::cout << "Удаляемое число: ";**

**std::cin >> num;**

**start = std::chrono::system\_clock::now();**

**spisok = remove\_items(spisok, num);**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на удаление: " << time.count() << '\n';**

**break;**

**case 2:**

**std::cout << "Удаляемая позиция: ";**

**std::cin >> size;**

**start = std::chrono::system\_clock::now();**

**spisok = remove(spisok, size);**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на удаление: " << time.count() << '\n';**

**break;**

**default:**

**choise = 0;**

**std::cout << "try again" << '\n';**

**break;**

**}**

**}**

**break;**

**case 8://возврат из списка**

**choise = 0;**

**while (!choise) {**

**std::cout << "1) По значению" << '\n';**

**std::cout << "2) По позиции" << '\n';**

**std::cin >> choise;**

**switch (choise) {**

**case 1:**

**std::cout << "Искомое число: ";**

**std::cin >> num;**

**start = std::chrono::system\_clock::now();**

**where\_item(spisok, num);**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на поиск: " << time.count() << '\n';**

**break;**

**case 2:**

**std::cout << "Искомая позиция: ";**

**std::cin >> size;**

**start = std::chrono::system\_clock::now();**

**std::cout << take(spisok, size) << '\n';**

**end = std::chrono::system\_clock::now();**

**time = end - start;**

**std::cout << "Времени на поиск: " << time.count() << '\n';**

**break;**

**default:**

**choise = 0;**

**std::cout << "try again" << '\n';**

**break;**

**}**

**}**

**break;**

**case 9://вывод списка и массива**

**std::cout << "Массив: ";**

**print\_array(arr);**

**std::cout << "Список: ";**

**print\_list(spisok);**

**break;**

**case 10://удалкние нечётных чисел**

**arr = remove\_odd\_items(arr);**

**spisok = remove\_odd\_items(spisok);**

**break;**

**case 11:**

**chet\_pos(arr, spisok);**

**break;**

**case 0:**

**return 0;**

**break;**

**default:**

**std::cout << "Try again";**

**break;**

**}**

**}**

**return 0;**

**}**