Государственный Университет Молдовы

Факультет Математики и Информатики

Кафедра Информатики

**Лабораторная работа №1**

по курсу «Алгоритмы и Структуры Данных»

Методы доступа к элементам массива.

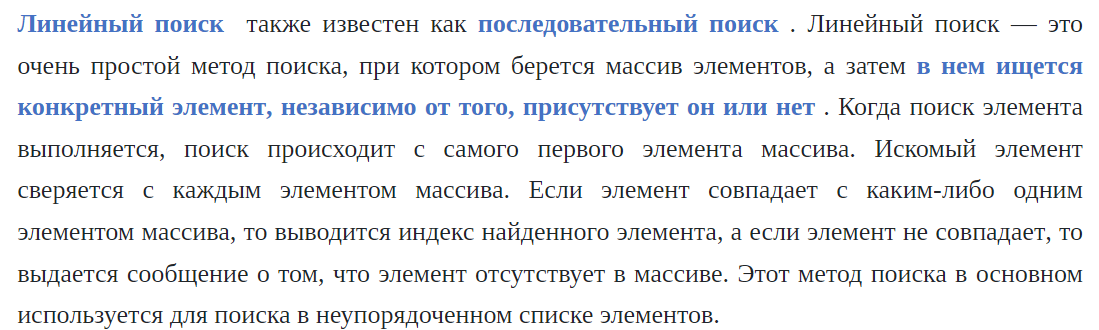
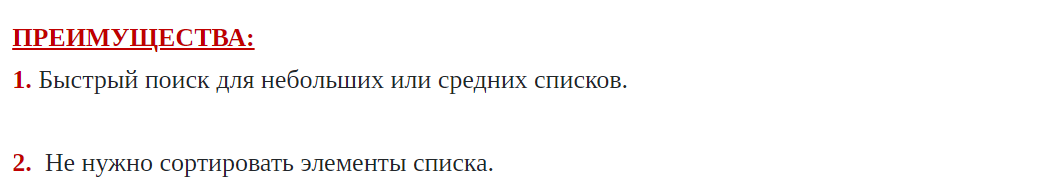
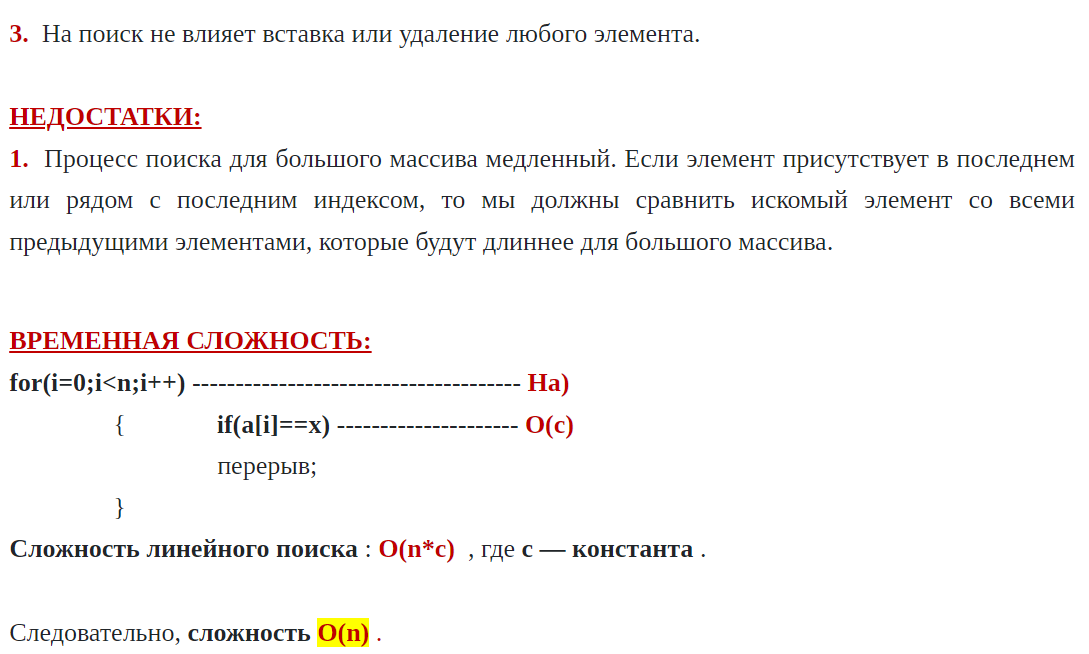
Выполнил студент группы I2102:

**Король Владимир**

Проверил преподаватель:

**Кроитор Михаил**

**Линейный поиск**

cout << "-------------------------POSLEDOVATELINII POISK--------------------------" << endl;

bool flag = false;

int var1;

int price\_value;

cout << "Enter the price value " << endl;

cin >> price\_value;

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

if (notebooks[i].price == price\_value) {

flag = true;

var1 = notebooks[i].id;

}

}

if (flag) {

cout << "Notebook number " << var1 << " has the price " << price\_value << endl;

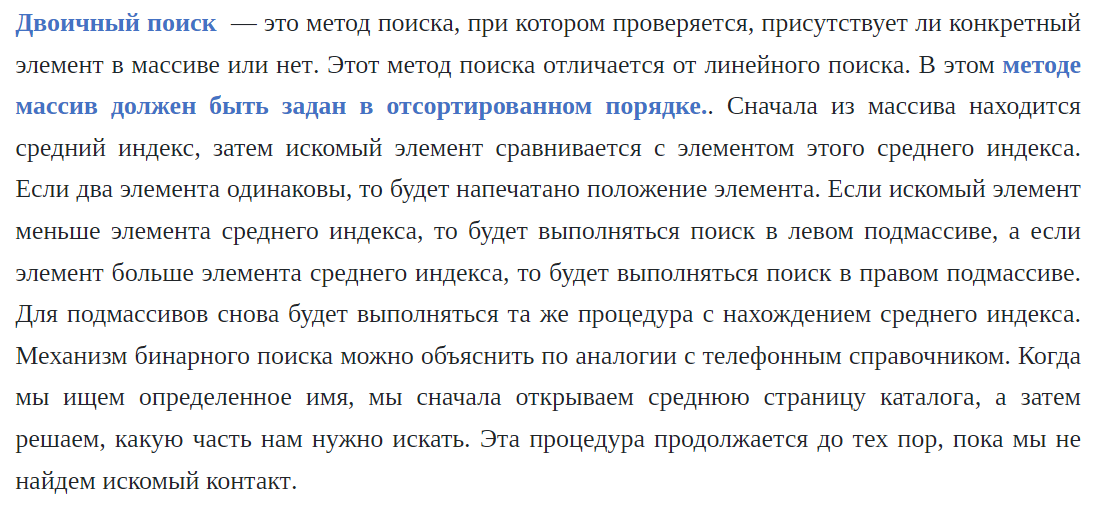
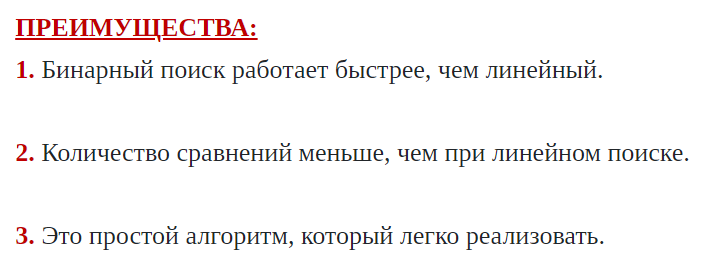
}

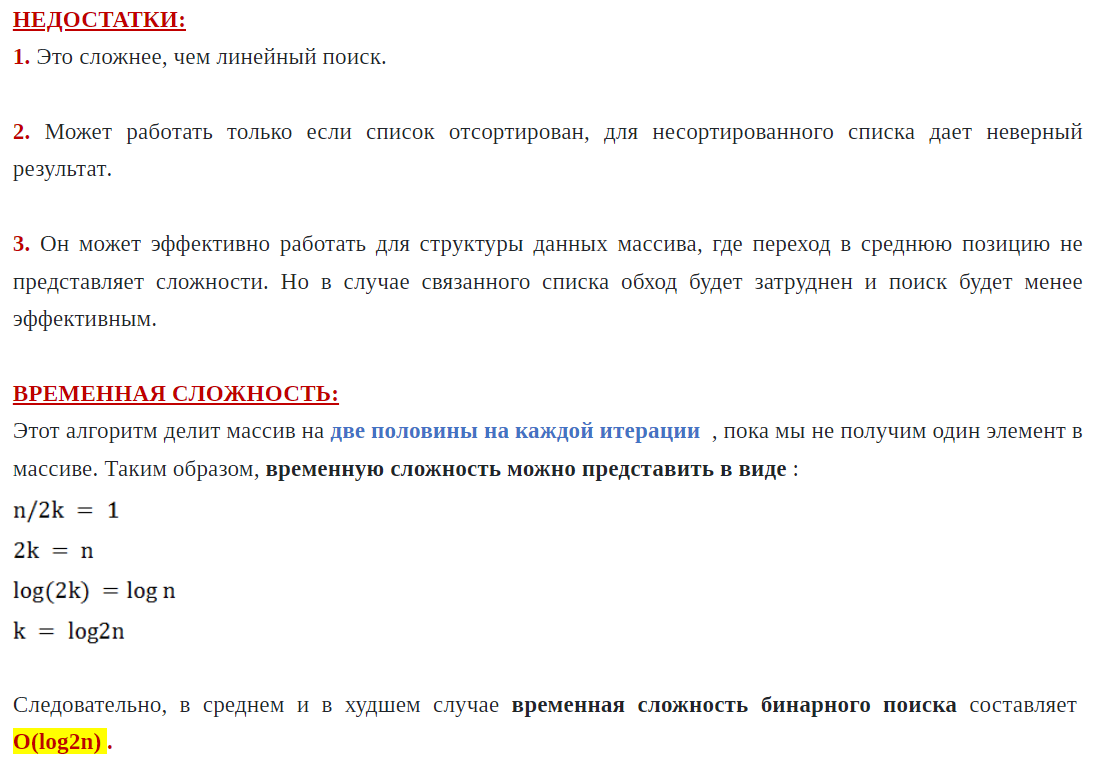
else {

cout << "There are no computers with this price" << endl;

}

**Бинарный поиск**



cout << "---------------------------BINARNII POISK----------------------------------------" << endl;

int price\_value;

int search = -1;

int left = 0;

int right = 50;

cout << "Enter the price value " << endl;

cin >> price\_value;

while (left <= right) {

int mid = (left + right) / 2;

if (price\_value == notebooks[mid].price) {

search = mid;

break;

}

if (price\_value < notebooks[mid].price)

right = mid - 1;

else

left = mid + 1;

}

if (search == -1) {

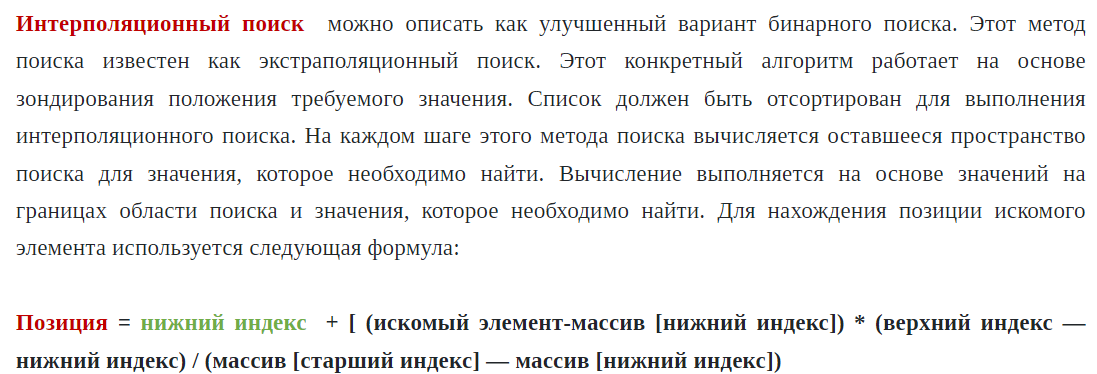
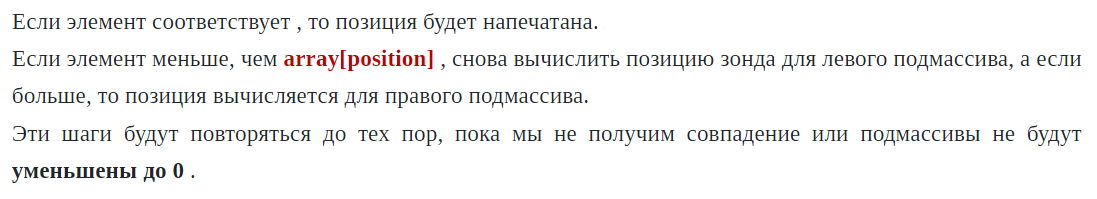
cout << "There are no computers with this price" << endl;

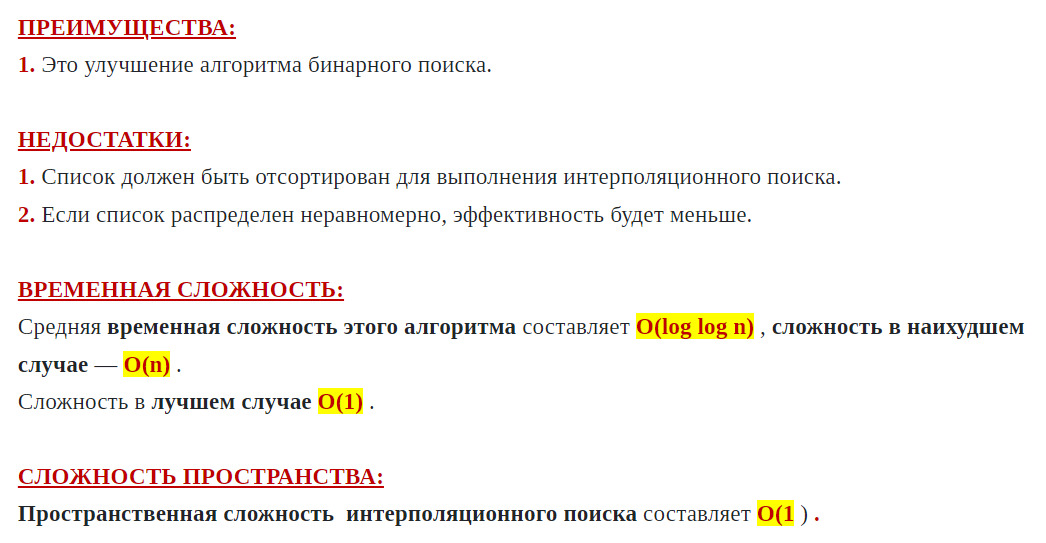
}

else

cout << "Notebook number " << search << " has the price " << price\_value << endl;

**Метод поиска интерполяцией**



cout << "---------------------POISK INTERPOLEATIEII-----------------------" << endl;

int x = 0;

int a = 0;

int b = 50;

bool flag;

int price\_value;

cout << "Enter the price value " << endl;

cin >> price\_value;

for (flag = false; (notebooks[a].price < price\_value) && (notebooks[b].price > price\_value) && !flag; ) {

x = a + ((price\_value - notebooks[a].price) \* (b - a)) / (notebooks[b].price - notebooks[a].price);

if (notebooks[x].price < price\_value)

a = x + 1;

else if (notebooks[x].price > price\_value)

b = x - 1;

else

flag = true;

}

if (notebooks[a].price == price\_value)

cout << price\_value << " founded in computer" << a << endl;

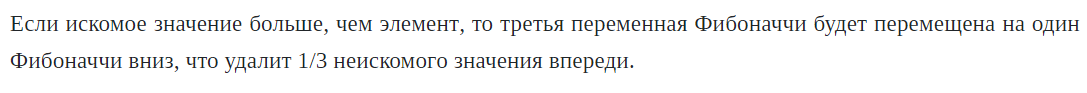
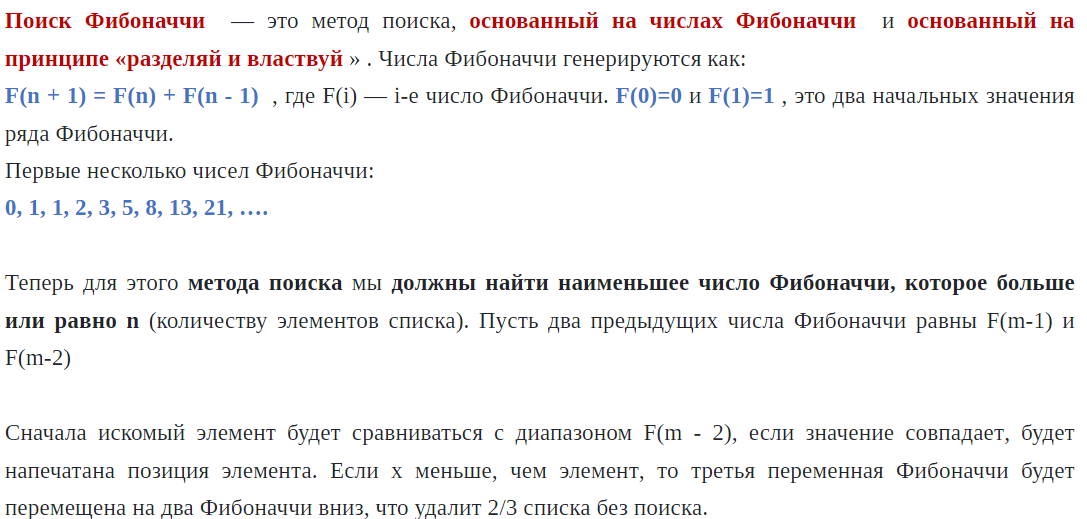
else if (notebooks[b].price == price\_value)

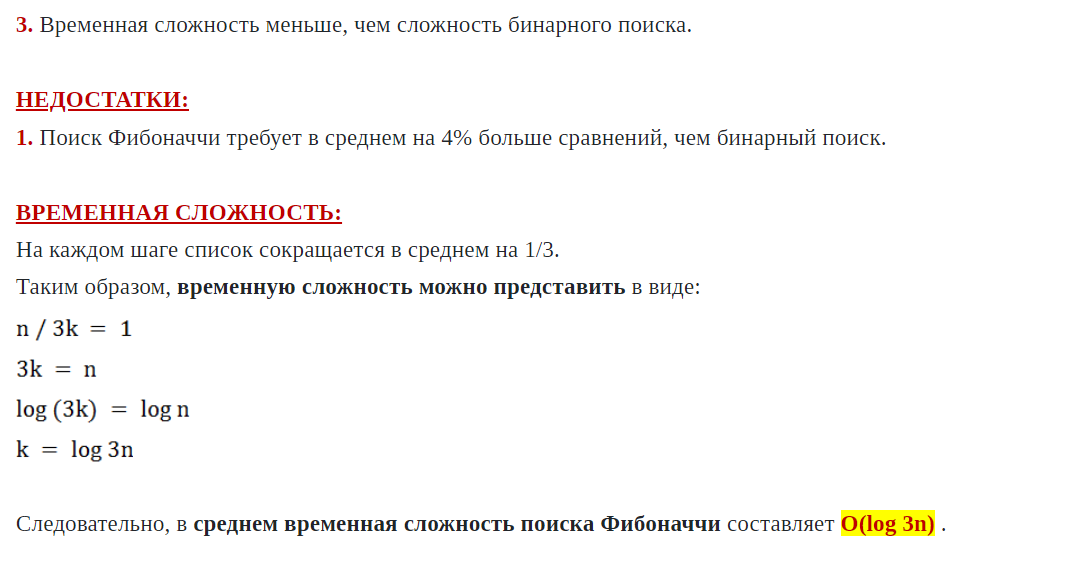
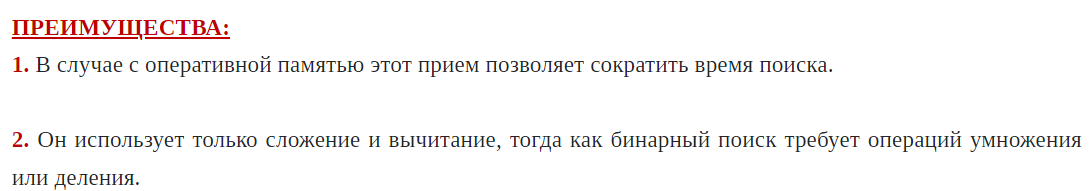
cout << price\_value << " founded in computer " << b << endl;

else

cout << "No elements with this price" << endl;

**Метод Фибоначи**





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

int i;

int a = 0;

int b = 1;

int c = a + b;

int n = 49;

int price\_value;

cout << "Enter the price value " << endl;

cin >> price\_value;

while (c < n) {

a = b;

b = c;

c = a + b;

}

int offset = -1;

while (c > 1) {

i = min(offset + a, n - 1);

if (notebooks[i].price < price\_value) {

c = b;

b = a;

a = c - b;

offset = i;

}

else if (notebooks[i].price > price\_value) {

c = a;

b = b - a;

a = c - b;

}

else {

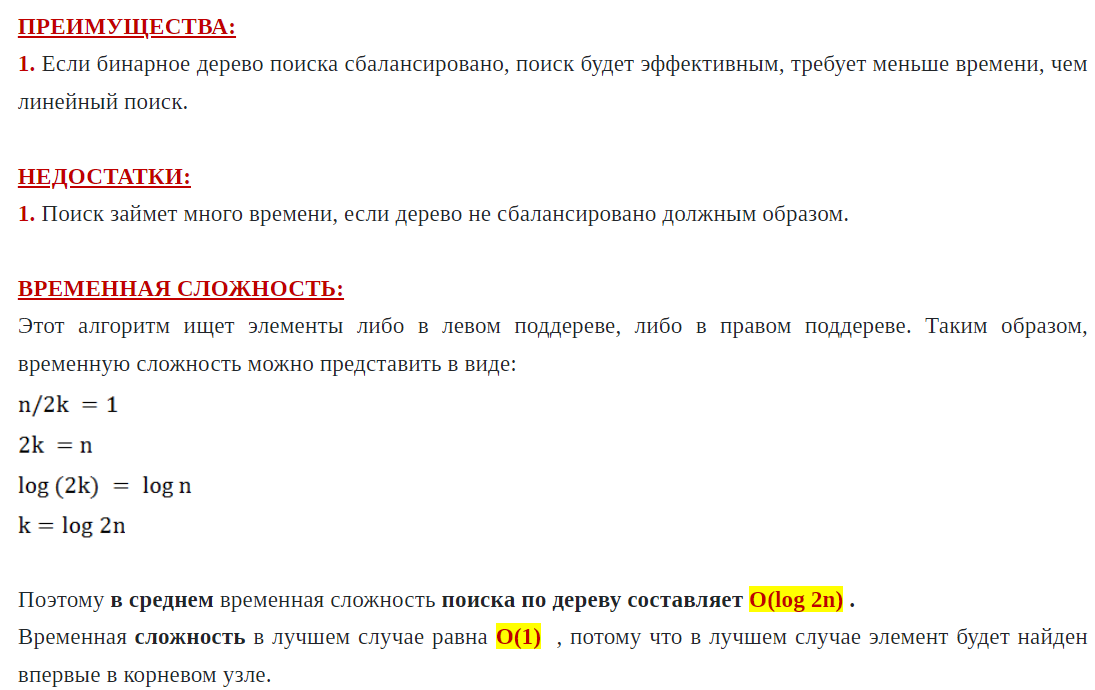
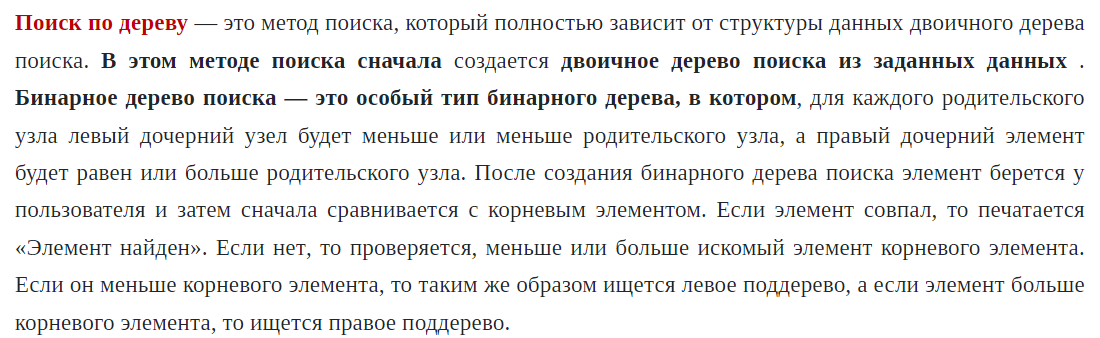
cout << "price " << price\_value << " is found at index " << i;

break;

}

}

**Поиск по дереву**



int result\_3;

cout << "\n\nTreeSearch" << endl;

cout << "Enter element search -> ";

BinTree\* Tree = NULL;

for (int i = 0; i < 50; i++) {

newBinTree(notebooks[i].price, &Tree);

}

int key;

cout << "Enter value for search -> ";

cin >> key;

int result\_4 = Search(Tree, key);

if (result\_4 == NULL)

cout << "Element is not found";

else

cout << "Your element is->" << endl;

for (int i = 0; i < 50; i++) {

if (result\_4 == notebooks[i].price) {

notebooks[i].display();

}

}

struct BinTree {

int value; //содержит значение

BinTree\* left;//адрес левого поддерева

BinTree\* right;//адрес правого поддерева

};

void newBinTree(int val, BinTree\*\* Tree) {

if ((\*Tree) == NULL)

{

(\*Tree) = new BinTree; //Выделить память

(\*Tree)->value = val; //Поместить в выделенное место аргумент

(\*Tree)->left = (\*Tree)->right = NULL;

}

if (val > (\*Tree)->value) newBinTree(val, &(\*Tree)->right);//Если аргумент больше чем текущий элемент, поместить его вправо

else if (val < (\*Tree)->value) newBinTree(val, &(\*Tree)->left);//Иначе поместить его влево

}

int Search(BinTree\* Tree, int key) {

if (Tree == NULL) return NULL;

if (Tree->value == key) return Tree->value;

if (key < Tree->value) return Search(Tree->left, key);

else

return Search(Tree->right, key);

}