Вариант 1  
Найти дубликаты в списке чисел с использованием дерева поиска  
  
#include<iostream>;

using namespace std;

int k = 0; //Summa

struct Tree//создание структуры

{

char data; // дата

Tree\* left; //левый указатель

Tree\* right; // правый

};

Tree\* Btree = NULL; // дерева нет

Tree\* create(int n)

{

Tree\* Q; //создаем указатель на корень

if (n == 0) Q = NULL; //возвращаем пустое дерево

else

{

Q = new Tree; //выделяем память под узел

int nl = n / 2, nr = n - nl - 1;

cout << "Введите значение узла ";

cin >> Q->data; //заполняем информационное поле

Q->left = create(nl); //переходим к созданию левого поддерева

Q->right = create(nr); //переходим к созданию правого поддерева

}

return Q;

} // создание дерева и элементов управления

void print(Tree\* Q, int l) //укладка дерева с помощью отступов

{

int i;

if (Q != NULL)

{

print(Q->right, l + 1);

for (i = 0; i < l; i++) cout << " ";

printf("%41c", Q->data);

print(Q->left, l + 1);

}

else cout << endl;

} // это не трогать, так в примере

Tree\* dubl(Tree\* Q) // структура суммы

{

if (Q->left != NULL && Q->right != NULL) // пока элементы не будут никакие

{

if (Q->left != NULL || Q->right != NULL)

{

if (Q->left->data == Q->data) //если дата слева = q

{

cout << Q->left->data << endl;

}

if (Q->data == Q->right->data)

{

cout << Q->data << endl;

}

if (Q->left->data == Q->right->data)

{

cout << Q->left->data << endl;

}

dubl(Q->left); // идет к след значению

dubl(Q->right);

}

}

return Q;

}// конец дерева

int main()

{

setlocale(LC\_ALL, "Russian"); //подключение русского

Tree\* a = create(7); //создаем дерево

print(a, 0);//вывод на экран

cout << "Дубликаты"<<endl;

dubl(a); // считывает ее

cout << endl;

}

