МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСТИЕТ

Контрольная работа работа №1
по дисциплине «Информатика»
Деревья

Группа: **АВТ-115**
Студент: **Качурин А. Е.**
Преподаватель: **Романов Е.Л.**

НОВОСИБИРСК 2012

Текстпрограммы

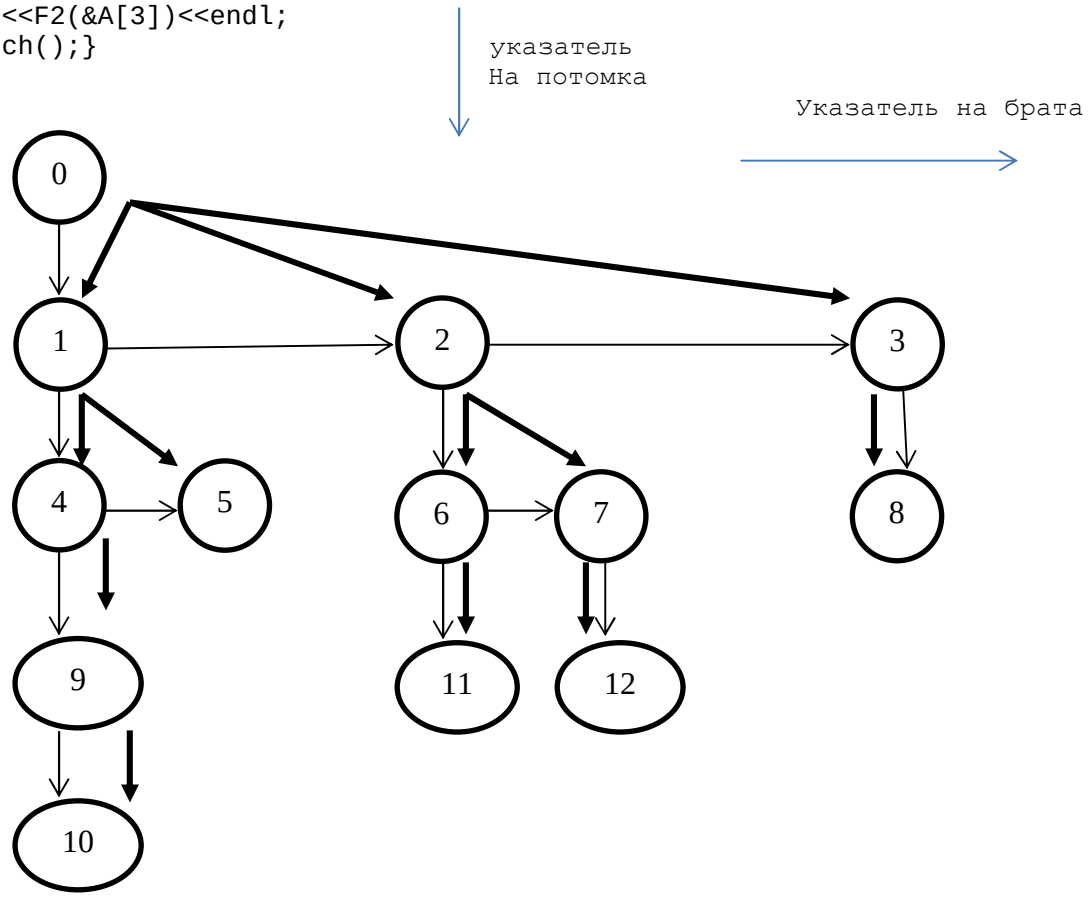
```
#include<iostream>
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
#include<conio.h>
usingnamespace std;
struct ltree{
      int val;
      ltree *child,*next;
      };

intF1(ltree *p){//подсчёт количества потомков
// во всех поколениях включая себя
      intn=1;//текущий элемент тоже учитывается
      for (ltree *q=p->child; q!=NULL; q=q->next)// перебираемпотомка
//иегобратьев
            n+=F1(q);//прибавляем количество элементов потомка или его брата
      //к формируемой сумме элементов потомка и его братьев
      return n;
      }

intF2(ltree *p){//суммачиселувсехпотомковвключаясвоё.
      intn=p->val;//единственное отличие -сумма чисел
      for (ltree *q=p->child; q!=NULL; q=q->next)
            n+=F2(q);
      return n;
      }

void main(){ltree A[13];
A[0].val=4;
A[0].child=&A[1];
A[0].next=NULL;
A[1].val=1;
A[1].child=&A[4];
A[1].next=&A[2];
A[2].val=5;
A[2].child=&A[6];
A[2].next=&A[3];
A[3].val=3;
A[3].child=&A[8];
A[3].next=NULL;
A[4].val=10;
A[4].child=&A[9];
A[4].next=&A[5];
A[5].val=3;
A[5].child=NULL;
A[5].next=NULL;
A[6].val=4;
A[6].child=&A[11];
A[6].next=&A[7];
A[7].val=7;
A[7].child=&A[12];
A[7].next=NULL;
A[8].val=5;
A[8].child=NULL;
A[8].next=NULL;
A[9].val=2;
A[9].child=&A[10];
A[9].next=NULL;
```

```
A[10].val=5;
A[10].child=NULL;
A[10].next=NULL;
A[11].val=9;
A[11].child=NULL;
A[11].next=NULL;
A[12].val=2;
A[12].child=NULL;
A[12].next=NULL;
cout<<F1(&A[0])<<endl;
cout<<F1(&A[2])<<endl;
cout<<F1(&A[4])<<endl<<endl;
cout<<F2(&A[0])<<endl;
cout<<F2(&A[1])<<endl;
cout<<F2(&A[3])<<endl;
_getch();}
```



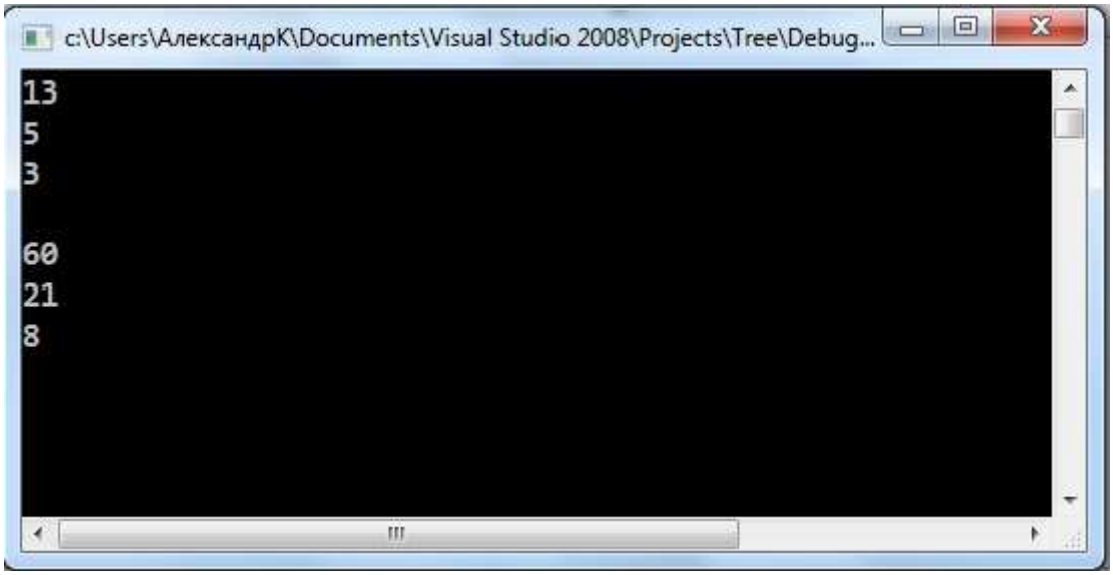
Пояснение

Каждый элемент дерева состоит из 3-х частей: число, указатель на потомка, указатель на брата. Строение дерева описано выше. Дерево, в данном случае состоит из 13 элементов.

Первая функция, начиная с текущего элемента, обращается к потомку и братьям потомка, суммируя воедино все их числа и вызывая себя для каждого элемент, если не наткнётся на пустой указатель, возвращая всю сумму элементов ветви, начиная с текущего элемента.

Вторая функция аналогична первой, но оперирует не количеством элементов, а суммой чисел во всей ветви.

Ниже приведены результаты троекратного вызова для первой и второй функции (для первой функции – с 0-го, 2-го и 4-го – количество элементов ветви; для второй функции – с 0-го, 1-го и 3-го – сумма элементов ветви).



Вывод: Вышеуказанные функции выводят (при текущих методах задания переменных) количество элементов ветви и сумму чисел элементов соответственно.