Бинарное дерево

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>

typedef int T;
typedef struct Item* Link_Item;
typedef Link_Item Tree;

struct Item { T Key; Link_Item Left; Link_Item Right; };
T Item_get_Key(const Link_Item a) { return a->Key; }
Link_Item Item_get_Left(const Link_Item a) {return a->Left; }
Link_Item Item_get_Right(const Link_Item a) {return a->Right; }
Link_Item Create_Item(T K, Link_Item L, Link_Item R) {
        Link_Item r = (Link_Item)calloc(1,sizeof(struct Item)); r->Key=K; r->Left=L; r->Right=R; return r; }
```

Печать ключей бинарного дерева

```
void Tree_print(const Tree a) {
int i; static int k=0;
   k++; if(Item_get_Right(a)) Tree_print(Item_get_Right(a)); k--;
   for(i=0;i<k;i++) putchar('\t'); printf("%d\n",Item_get_Key(a));
   k++; if(Item_get_Left(a)) Tree_print(Item_get_Left(a)); k--;
}
```

Компаратор ключей (для их сортировки)

```
int cmp(const void* a, const void* b) { return *(T*)a-*(T*)b; }
```

Построение дерева поиска

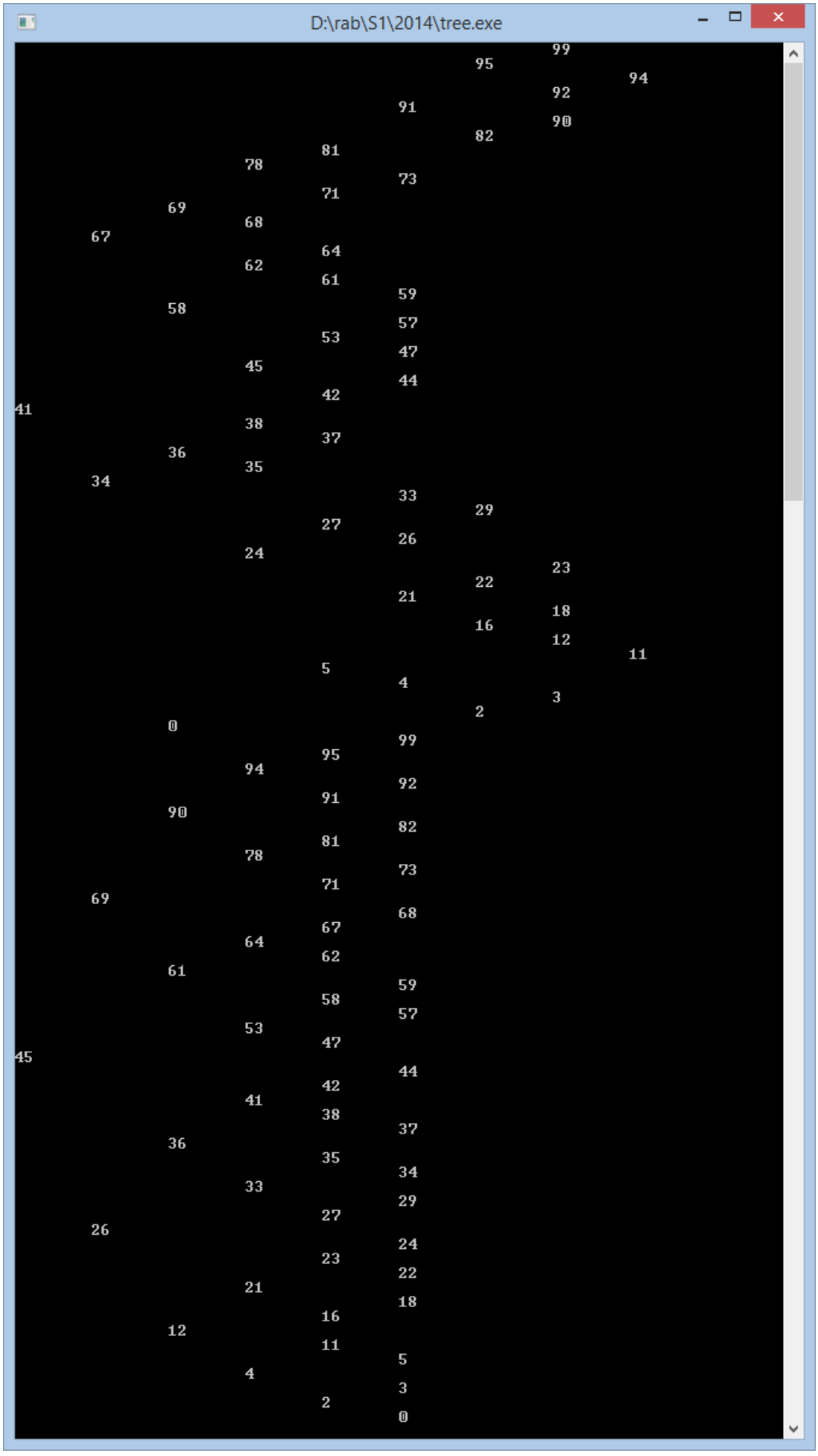
```
Tree Tree_Insert(Tree a, T K) {
   if(a==NULL) return a = Create_Item(K,NULL,NULL);
int r=Item_get_Key(a)-K;
   if(r<0) a->Right = Tree_Insert(Item_get_Right(a),K);
   else if(r>0) a->Left = Tree_Insert(Item_get_Left(a),K);
   return a; }
```

Построение идеально сбалансированного дерева

```
Tree Tree_Define(Tree a, T b[], int n) {
int nl=n/2, nr=n-nl-1;
    if(n) a = Tree_Insert(a,b[n/2]); else return a;
    a=Tree_Define(a,b,nl); a=Tree_Define(a,&b[n/2+1],nr);
    return a;
}
```

Пример построения дерева поиска и идеально сбалансированного дерева поиска

```
int main() {
Tree a1=NULL; Tree a2=NULL;
const int N=63;
int i,k,b[N];
   for(i=0;i<N;i++) { a1 = Tree_Insert(a1,k=rand()%100); b[i]=k; }
   qsort(b,N,sizeof(T),cmp);
   a2=Tree_Define(a2,b,N);
   Tree_print(a1);
   Tree_print(a2);
   return 0;
}
```


D:\rab\S1\2014\tree.exe
99
95
94
92
91
90
82
81
78
73
71
69
68
67
64
62
61
59
58
57
53
47
45
44
42
41
38
37
36
35
34
33
29
27
26
24
23
22
21
18
16
12
11
5
4
3
2
0
99
95
94
92
91
90
82
81
78
73
71
69
68
67
64
62
61
59
58
57
53
47
45
44
42
41
38
37
36
35
34
33
29
27
26
24
23
22
21
18
16
12
11
5
4
3
2
0