**Структуры и объединения СР**

**Цель лабораторной работы**: освоить структуры и объединения с помощью языка программирования C.

**Задача 1**

**Постановка задачи**: 

**Программный код:**

#include<stdlib.h>

#include<stdio.h>

typedef struct myStr

{

struct myStr \*f;

}myStr;

int main(){

myStr obj1, obj2, obj3, obj4;

obj1.f = &obj2;

obj2.f = &obj3;

obj3.f = &obj4;

obj4.f = &obj1;

system("pause");

return 0;

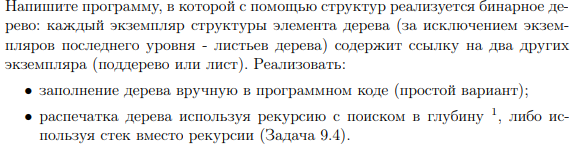
}

**Вывод консоли:**



**Задача 3**

**Постановка задачи**:



**Программный код:**

#include<stdlib.h>

#include<stdio.h>

typedef struct myTree

{

int value;

struct myTree\* leftH;

struct myTree\* rightH;

}myTree;

void create(myTree \*k, int lvl);

void delet(myTree \*k);

void print(myTree \*k);

int main(){

myTree\* k = malloc(sizeof(myTree));

create(k, 3);

print(k);

delet(k);

system("pause");

return 0;

}

void create(myTree \*k, int lvl)

{

myTree \*left, \*right;

k->value = lvl;

if (lvl)

{

left = malloc(sizeof(myTree));

right = malloc(sizeof(myTree));

k->leftH = left;

k->rightH = right;

lvl--;

creat(left, lvl);

creat(right, lvl);

}

else

{

k->leftH = NULL;

k->rightH = NULL;

}

}

void delet(myTree\* k)

{

if (k->leftH != NULL)

delet(k->leftH);

if (k->rightH != NULL)

delet(k->rightH);

free(k);

}

void print(myTree\* k)

{

printf("%d ", k->value);

if (k->leftH != NULL)

print(k->leftH);

if (k->rightH != NULL)

print(k->rightH);

}

**Вывод консоли:**

