Практическая работа №6

«Дерево»

Цель: Изучить сд «дерево» и реализовать её на языке С++

Дерево – структура данных, представляющая собой древовидную структуру в виде набора связанных узлов.

Бинарное дерево — это конечное множество элементов, которое либо пусто, либо содержит элемент (корень), связанный с двумя различными бинарными деревьями, называемыми левым и правым поддеревьями. Каждый элемент бинарного дерева называется узлом. Связи между узлами дерева называются его ветвями.

Структура

```
struct node
{
   int key_value; //ключ – значение узла, типа int
   node* left;    //указатель на левого потомка
   node* right;   //указатель на правого потомка
};
```

Добавление узла в дерево

```
void btree::insert(int key)      //Функция, доступная для элементов, которые не являются членами класса
{                                          //Сначала проверит корневой элемент
  if(root!=NULL)                           //Если он не инициализирован,
    insert(key, root);              //то вызовется рекурсивная функция для добавления элемента.
  else                                     //Иначе Функция поместит новое значение:
  {
    root=new node;                         //инициализируется корневой элемент
    root->key_value=key;//поместится ключевое значение в соответствующую ячейку структуры.
    root->left=NULL;              //Инициализируется указатель на левый
    root->right=NULL;                      //и правый элемент как NULL
  }
}
```

					АиСД.09.03.02.050000 ПР		
Изм.	*Лист*	*№ докум.*	*Подпись*	*Дата*			
Разраб.		*Ермошина В.А*			Практическая работа №6 Дерево	*Лит.*	*Лист* 2 / *Листов*
Провер.		*Берёза А.Н.*					
Реценз						*ИСОиП (филиал) ДГТУ в г.Шахты ИСТ-Tb21*	
Н. Контр.							
Утверд.							

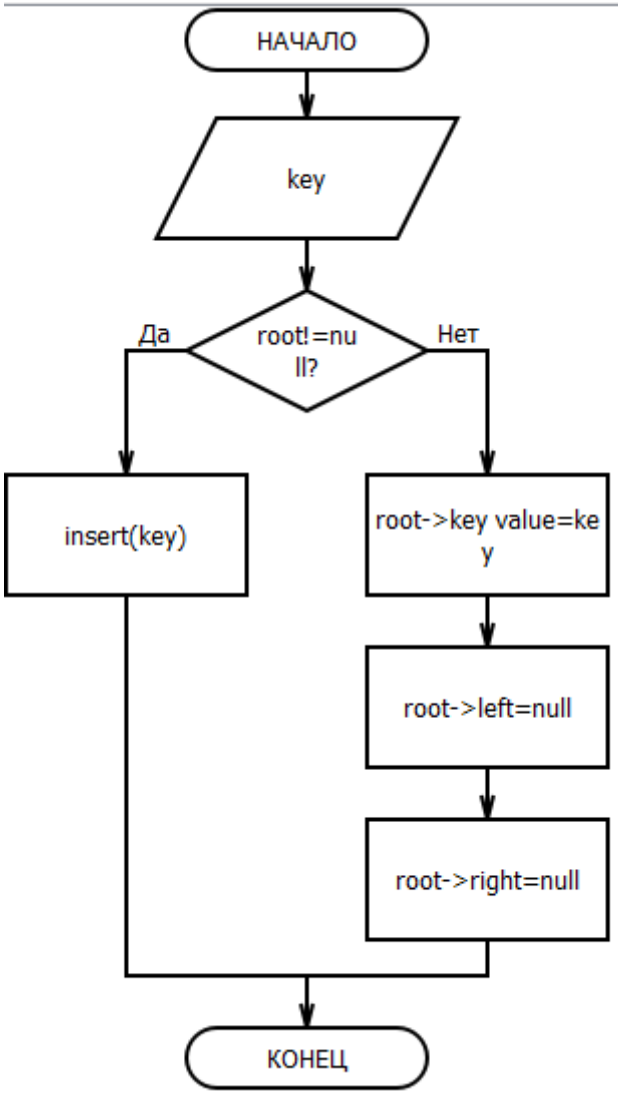

Рисунок 1. -Диаграмма для функции добавления без рекурсии

```
//Рекурсивная функция для вставки элемента.
void btree::insert(int key, node* leaf)//(Новое ключевое значение, указатель на текущий узел)
{
if (key < leaf->key_value)  //Если новое ключевое значение меньше чем ключевое значение в узле
   {
      if (leaf->left != NULL)//И если левый указатель узла инициализирован
         insert(key, leaf->left);//Функция вызывает саму себя, для левого узла потомка
else       {
 leaf->left = new node;//Функция создаст и поместит новый элемент на место левого потомка.
         leaf->left->key_value = key;//Внесёт новое ключевое значение в элемент.
         leaf->left->left = NULL;    //Установит левый дочерний указатель
         leaf->left->right = NULL;   //и правый дочерний указатель в NULL.
      }
   }
else if (key >= leaf->key_value)//Иначе Если новое ключевое значение не меньше ключевого значения в узле
   {
      if (leaf->right != NULL)//И если правый указатель инициализирован,
```

```
        insert(key, leaf->right);//То функция вызывает саму себя, для правого потомка
else//Иначе(если правый потомок не //инициализирован)
      {
     leaf->right = new node;//Функция создаст и поместит новый элемент на //место правого потомка.
        leaf->right->key_value = key;//Внесёт новое ключевое значение в элемент.
        leaf->right->left = NULL;  //Установит левый дочерний указатель
        leaf->right->right = NULL;      //и правый дочерний указатель в NULL.
      }
   }
}
```

Поиск

```
node* btree::search(int key, node* leaf)//(Ключевое значение, //указатель ну узел)
{
if (leaf != NULL)//Если узел //инициализирован
   {
  if (key == leaf->key_value)//И ключевое значение //узла совпадает с //искомым ключевым значением.
return leaf;//Возвращается указатель на найденный элемент.
if (key < leaf->key_value)//Иначе, если искомое ключевое //значение меньше //ключевого значения узла
return search(key, leaf->left);//Вызывается эта же функция //для левого потомка.
      else//Иначе
   return search(key, leaf->right);//Вызывается эта же функция //для правого потомка.
   }
   else return NULL;//Иначе возвращается NULL.
}
```

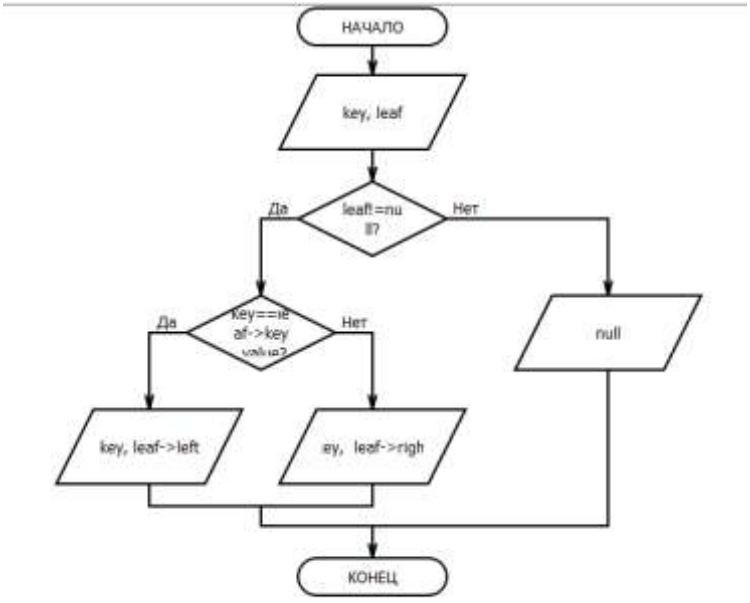

Рисунок 2. -Диаграмма функции поиска

Удаление узла

```
void btree::destroy_tree(node *leaf)
{
    if(leaf!=NULL)//Если узел дерева существует
```

```
    {//Функция вызовет сама себя
       destroy_tree(leaf->left);//сначала для левого потомка,
       destroy_tree(leaf->right);  //после для правого потомка.
       delete leaf;//Если потомков нет, она удалит
//Узел, полученный в качестве параметра //функции
void btree::insert(int key)  //Функция, доступная для элементов, которые не  являются членами класса
{//Сначала проверит корневой элемент
 if(root!=NULL)//Если он не инициализирован,
  insert(key, root);//то вызовется рекурсивная функция для добавления элемента.
 else      //Иначе Функция поместит новое значение:
 {
  root=new node;      //инициализируется корневой элемент
  root->key_value=key //поместится ключевое значение в соответствующую ячейку структуры.
  root->left=NULL;//Инициализируется указатель на левый
  root->right=NULL;//и правый элемент как NULL
 }
}
```

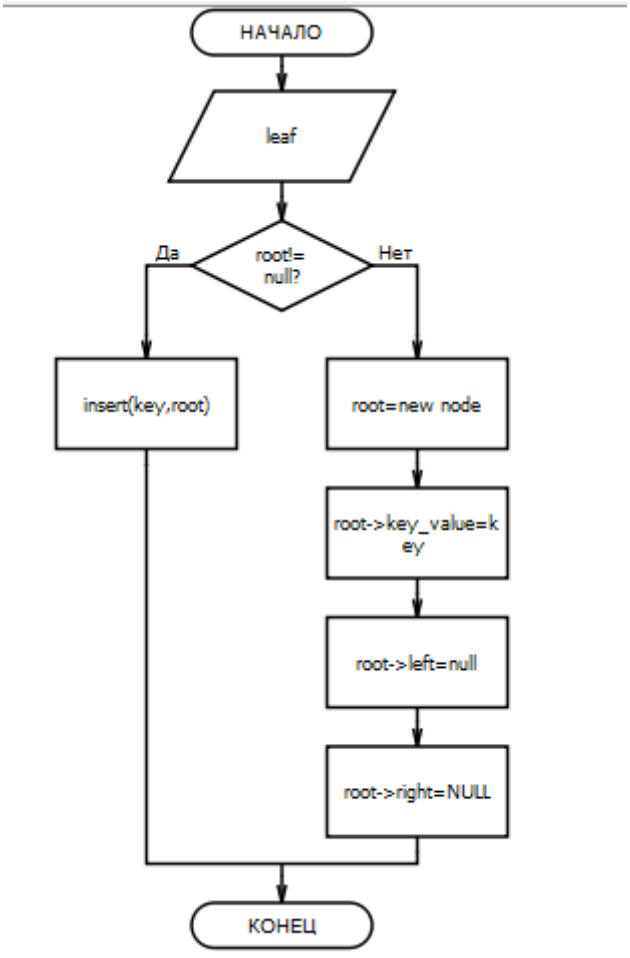

Рисунок 3. - Диаграмма для функции удаления

Вывод: В данной практической работе изучена сд «дерево» и реализован на языке C++