ЛЕКЦИЯ №3

ЦИКЛИЧЕСКИЙ СПИСОК
ДЕРЕВЬЯ

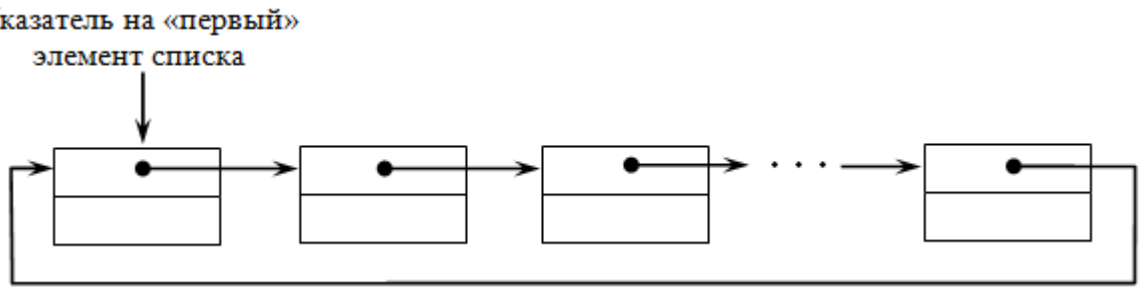
ЦИКЛИЧЕСКИЙ СПИСОК

Циклический (кольцевой) список – это структура данных, представляющая собой последовательность элементов, последний элемент которой содержит указатель на первый элемент списка, а первый (в случае двунаправленного списка) – на последний.

Основная особенность такой организации состоит в том, что в этом списке нет элементов, содержащих пустые указатели, и, следовательно, нельзя выделить крайние элементы.

ЦИКЛИЧЕСКИЙ СПИСОК ОДНОНАПР.

Циклические списки, так же как и линейные, бывают однонаправленными и двунаправленными.



Циклический однонаправленный список похож на линейный однонаправленный список, но его последний элемент содержит указатель, связывающий его с первым элементом

ЦИКЛИЧЕСКИЙ СПИСОК ОДНОНАПР.

Для полного обхода такого списка достаточно иметь указатель на произвольный элемент, а не на первый, как в линейном однонаправленном списке. Понятие "первого" элемента здесь достаточно условно и не всегда требуется. Хотя иногда бывает полезно выделить некоторый элемент как "первый" путем установки на него специального указателя. Это требуется, например, для предотвращения "зацикливания" при просмотре списка.

ЦИКЛИЧЕСКИЙ СПИСОК ОДНОНАПР.

Основные операции, осуществляемые с циклическим однонаправленным списком:

1. создание списка;
2. печать (просмотр) списка;
3. вставка элемента в список;
4. удаление элемента из списка;
5. поиск элемента в списке;
6. проверка пустоты списка;
7. удаление списка.

ЦИКЛИЧЕСКИЙ СПИСОК ОДНОНАПР.

Для описания алгоритмов этих основных операций будем использовать те же объявления, что и для линейного однонаправленного списка.

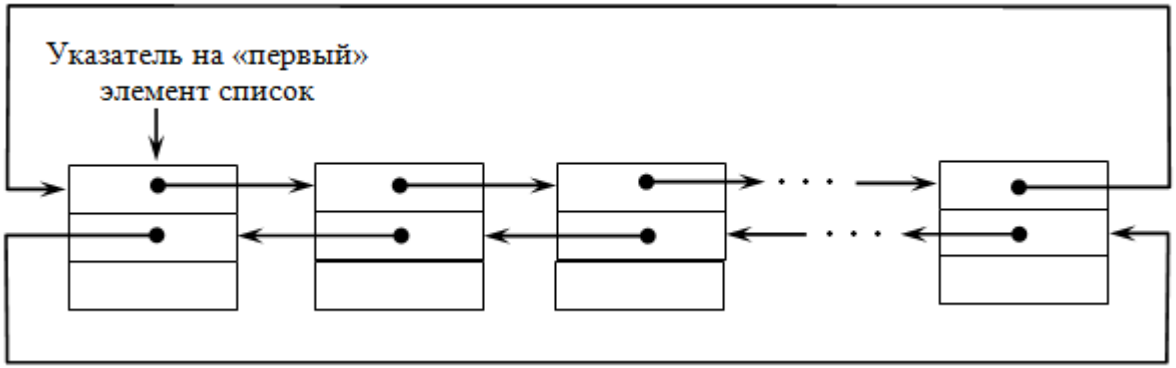
ЦИКЛИЧЕСКИЙ СПИСОК ОДНОНАПР.

Для описания алгоритмов этих основных операций используют те же объявления, что и для линейного однонаправленного списка.

Подробно останавливаться на этих объявлениях и действиях с циклическим списком (в рамках лекций) мы не будем.

ЦИКЛИЧЕСКИЙ СПИСОК ДВУНАПР.

Циклический двунаправленный список похож на линейный двунаправленный список, но его любой элемент имеет два указателя, один из которых указывает на следующий элемент в списке, а второй указывает на предыдущий элемент.



ЦИКЛИЧЕСКИЙ СПИСОК ДВУНАПР.

Основные операции, осуществляемые с циклическим двунаправленным списком:

1. создание списка;
2. печать (просмотр) списка;
3. вставка элемента в список;
4. удаление элемента из списка;
5. поиск элемента в списке
6. проверка пустоты списка;
7. удаление списка.

ЦИКЛИЧЕСКИЙ СПИСОК ДВУНАПР.

Для описания алгоритмов этих основных операций будем использовать те же объявления, что и для линейного двунаправленного списка (см. лекцию №2).

ДЕРЕВЬЯ

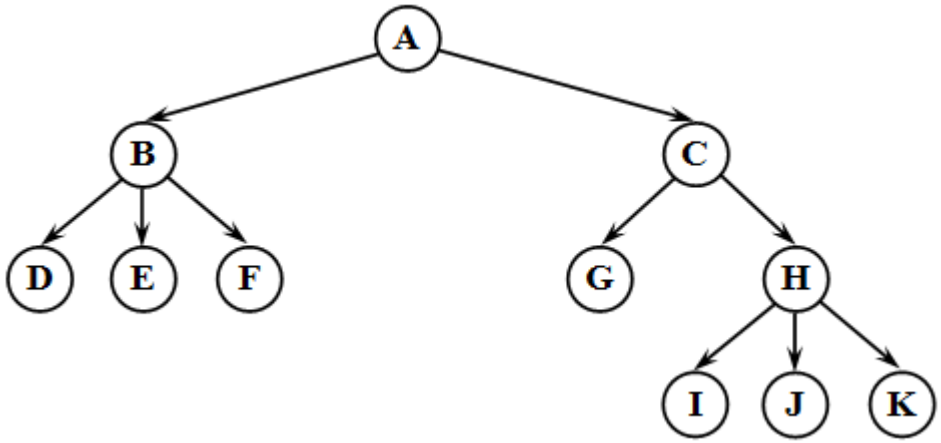
Дерево – это структура данных, представляющая собой совокупность элементов и отношений, образующих иерархическую структуру этих элементов. Каждый элемент дерева называется вершиной (узлом) дерева. Вершины дерева соединены направленными дугами, которые называют ветвями дерева. Начальный узел дерева называют корнем дерева, ему соответствует нулевой уровень. Листьями дерева называют вершины, в которые входит одна ветвь и не выходит ни одной ветви.

ДЕРЕВЬЯ

Каждое дерево обладает следующими свойствами:

существует узел, в который не входит ни одной дуги (корень);

в каждую вершину, кроме корня, входит одна дуга.



ДЕРЕВЬЯ

Все вершины, в которые входят ветви, исходящие из одной общей вершины, называются потомками, а сама вершина – предком. Для каждого предка может быть выделено несколько. Уровень потомка на единицу превосходит уровень его предка. Корень дерева не имеет предка, а листья дерева не имеют потомков.

ДЕРЕВЬЯ

Высота (глубина) дерева определяется количеством уровней, на которых располагаются его вершины. Высота пустого дерева равна нулю, высота дерева из одного корня – единице. На первом уровне дерева может быть только одна вершина – корень дерева, на втором – потомки корня дерева, на третьем – потомки потомков корня дерева и т.д.

ДЕРЕВЬЯ

Поддерево – часть древообразной структуры данных, которая может быть представлена в виде отдельного дерева.

Степенью вершины в дереве называется количество дуг, которое из нее выходит. Степень дерева равна максимальной степени вершины, входящей в дерево. При этом листьями в дереве являются вершины, имеющие степень нуль. По величине степени дерева различают два типа деревьев:

- **двоичные – степень дерева не более двух;**
- **сильноветвящиеся – степень дерева произвольная.**

ДЕРЕВЬЯ

Упорядоченное дерево – это дерево, у которого ветви, исходящие из каждой вершины, упорядочены по определенному критерию.

Деревья являются рекурсивными структурами, так как каждое поддерево также является деревом. Таким образом, дерево можно определить как рекурсивную структуру, в которой каждый элемент является:

- **либо пустой структурой;**
- **либо элементом, с которым связано конечное число поддеревьев.**

ДЕРЕВЬЯ

Действия с рекурсивными структурами удобнее всего описываются с помощью рекурсивных алгоритмов.

Списочное представление деревьев основано на элементах, соответствующих вершинам дерева. Каждый элемент имеет поле данных и два поля указателей: указатель на начало списка потомков вершины и указатель на следующий элемент в списке потомков текущего уровня. При таком способе представления дерева обязательно следует сохранять указатель на вершину, являющуюся корнем дерева.

ДЕРЕВЬЯ

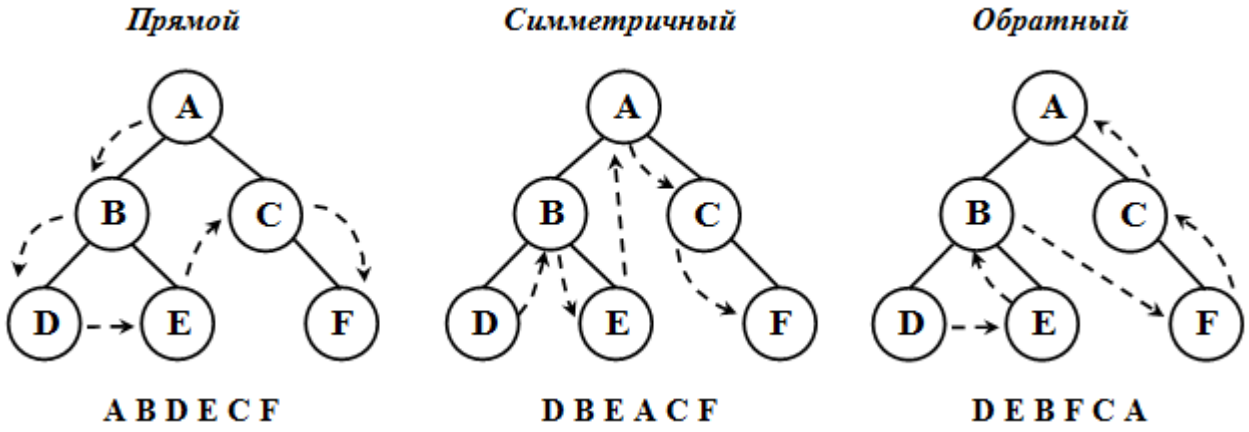
Для того, чтобы выполнить определенную операцию над всеми вершинами дерева необходимо все его вершины просмотреть. Такая задача называется обходом дерева.

Обход дерева – это упорядоченная последовательность вершин дерева, в которой каждая вершина встречается только один раз.

При обходе все вершины дерева должны посещаться в определенном порядке. Существует несколько способов обхода всех вершин дерева. Выделим три наиболее часто используемых способа обхода дерева

ДЕРЕВЬЯ

1) прямой; 2) симметричный; 3) обратный.



Существует большое многообразие древовидных структур данных. Выделим самые распространенные из них: бинарные (двоичные) деревья, красно-черные деревья, B-деревья, АВЛ-деревья, матричные деревья, смешанные деревья и т.д.

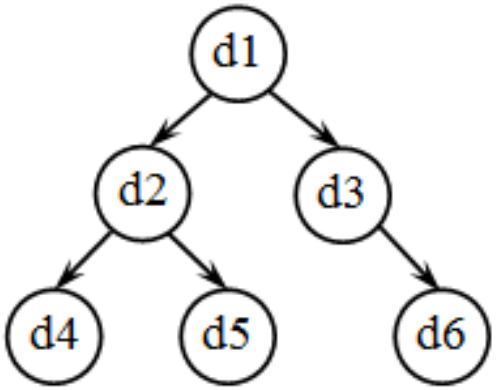
БИНАРНЫЕ ДЕРЕВЬЯ

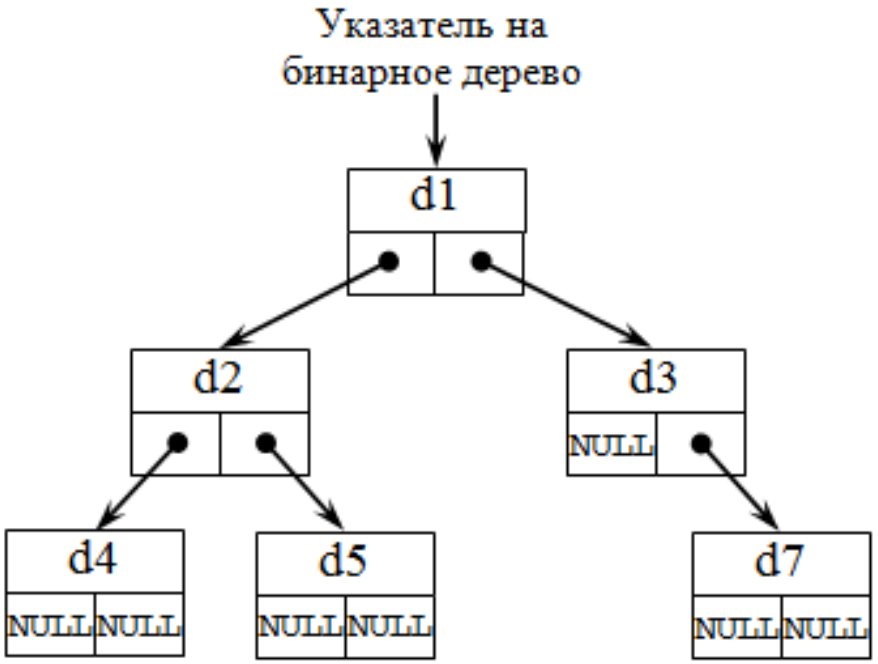
Бинарные деревья являются деревьями со степенью не более двух.

Бинарное (двоичное) дерево – это динамическая структура данных, представляющее собой дерево, в котором каждая вершина имеет не более двух потомков.

Таким образом, бинарное дерево состоит из элементов, каждый из которых содержит информационное поле и не более двух ссылок на различные бинарные поддеревья. На каждый элемент дерева имеется ровно одна ссылка.

БИНАРНЫЕ ДЕРЕВЬЯ





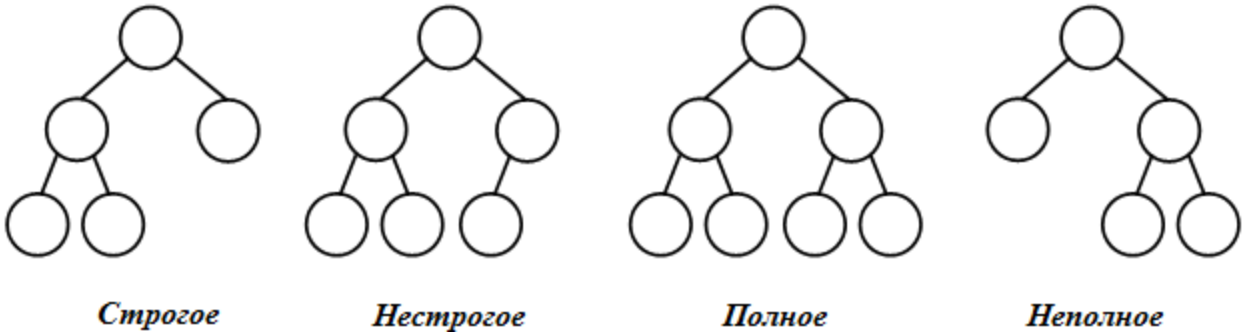
БИНАРНЫЕ ДЕРЕВЬЯ

Каждая вершина бинарного дерева является структурой, состоящей из четырех видов полей. Содержимым этих полей будут соответственно:

- информационное поле (ключ вершины);
- служебное поле (их может быть несколько или ни одного);
- указатель на левое поддерево;
- указатель на правое поддерево.

БИНАРНЫЕ ДЕРЕВЬЯ

По степени вершин бинарные деревья делятся на:



строгие – вершины дерева имеют степень ноль (у листьев) или два (у узлов);

нестрогие – вершины дерева имеют степень ноль (у листьев), один или два (у узлов).

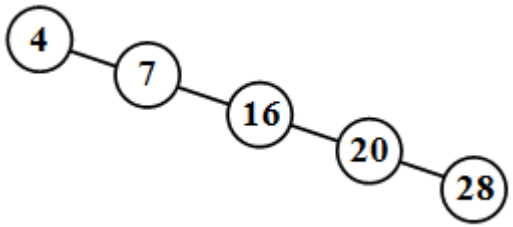
БИНАРНЫЕ ДЕРЕВЬЯ

В общем случае у бинарного дерева на k - м уровне может быть до 2^{k-1} вершин. Бинарное дерево называется полным, если оно содержит только полностью заполненные уровни. В противном случае оно является неполным.

Дерево называется сбалансированным, если длины всех путей от корня к внешним вершинам равны между собой. Дерево называется почти сбалансированным, если длины всевозможных путей от корня к внешним вершинам отличаются не более, чем на единицу.

БИНАРНЫЕ ДЕРЕВЬЯ

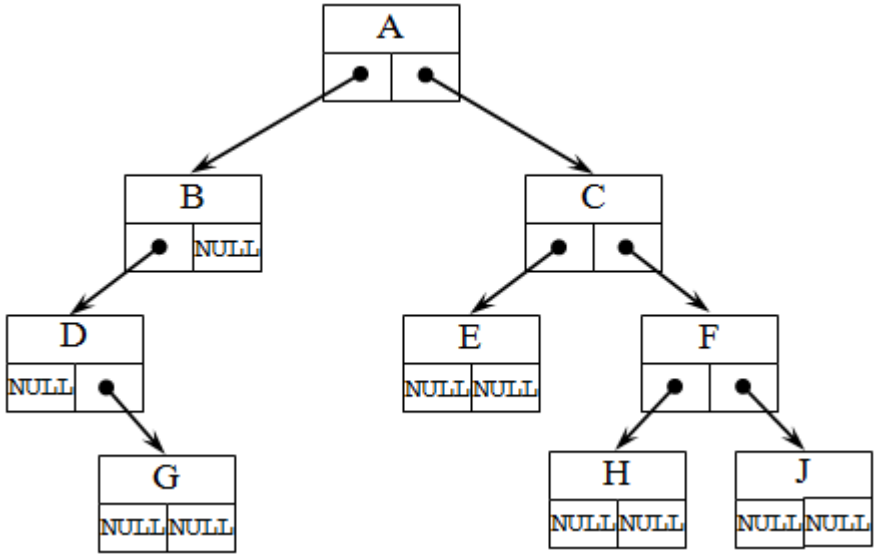
Бинарное дерево может представлять собой пустое множество. Бинарное дерево может выродиться в список:



Список – частный случай бинарного дерева.

БИНАРНЫЕ ДЕРЕВЬЯ

Структура дерева отражается во входном потоке данных так: каждой вводимой пустой связи соответствует условный символ, например, '*' (звездочка). При этом сначала описываются левые потомки, затем, правые. Для структуры бинарного дерева, представленного на следующем, *входной поток* имеет вид: ABD*G*CE**FH**J****



БИНАРНЫЕ ДЕРЕВЬЯ

Бинарные деревья могут применяться для поиска данных в специально построенных деревьях (базы данных), сортировки данных, вычислений арифметических выражений, кодирования (метод Хаффмана) и т.д.

Описание бинарного дерева выглядит следующим образом:

```
struct имя_типа {
        информационное поле;
        [служебное поле;]
        адрес левого поддерева;
        адрес правого поддерева;
      };
```

где информационное поле – это поле любого ранее объявленного или стандартного типа;

адрес левого (правого) поддерева – это указатель на объект того же типа, что и определяемая структура, в него записывается адрес следующего элемента левого (правого) поддерева.

БИНАРНЫЕ ДЕРЕВЬЯ

Например:

```
struct point {
        int data;//информационное поле
        int count; //служебное поле
        point *left;//адрес левого поддерева
        point *right;//адрес правого поддерева
      };
```

БИНАРНЫЕ ДЕРЕВЬЯ

Основными операциями, осуществляемыми с бинарными деревьями, являются:

1. создание бинарного дерева;
2. печать бинарного дерева;
3. обход бинарного дерева;
4. вставка элемента в бинарное дерево;
5. удаление элемента из бинарного дерева;
6. проверка пустоты бинарного дерева;
7. удаление бинарного дерева.

БИНАРНЫЕ ДЕРЕВЬЯ

Для описания алгоритмов этих основных операций используется следующее объявление:

```
struct BinaryTree{
    int Data; //поле данных
    BinaryTree* Left; //указатель на левый потомок
    BinaryTree* Right; /указатель на правый потомок
};
. . . . . .
BinaryTree* BTree = NULL;
```

БИНАРНЫЕ ДЕРЕВЬЯ

Приведем функции перечисленных основных операций при работе с бинарным деревом.

```
//создание бинарного дерева
void Make_Binary_Tree(BinaryTree** Node, int n){
  BinaryTree** ptr;//вспомогательный указатель
  srand(time(NULL)*1000);
  while (n > 0) {
    ptr = Node;
    while (*ptr != NULL) {
      if ((double) rand()/RAND_MAX < 0.5)
        ptr = &((*ptr)->Left);
      else ptr = &((*ptr)->Right);
    }
    (*ptr) = new BinaryTree();
    cout << "Введите значение ";
    cin >> (*ptr)->Data;
    n--;
  }
}
```

БИНАРНЫЕ ДЕРЕВЬЯ

```
        //печать бинарного дерева
void Print_BinaryTree(BinaryTree* Node, int l){
  int i;
  if (Node != NULL) {
    Print_BinaryTree(Node->Right, l+1);
    for (i=0; i< l; i++) cout << "    ";
    printf ("%4ld", Node->Data);
    Print_BinaryTree(Node->Left, l+1);
  }
  else cout << endl;
}
```

БИНАРНЫЕ ДЕРЕВЬЯ

```
        //прямой обход бинарного дерева
void PreOrder_BinaryTree(BinaryTree* Node){
  if (Node != NULL) {
    printf ("%3ld",Node->Data);
    PreOrder_BinaryTree(Node->Left);
    PreOrder_BinaryTree(Node->Right);
  }
}
```

БИНАРНЫЕ ДЕРЕВЬЯ

```
        //обратный обход бинарного дерева
void PostOrder_BinaryTree(BinaryTree* Node){
  if (Node != NULL) {
    PostOrder_BinaryTree(Node->Left);
    PostOrder_BinaryTree(Node->Right);
    printf ("%3ld",Node->Data);
  }
}
```

БИНАРНЫЕ ДЕРЕВЬЯ

```
//симметричный обход бинарного дерева
void SymmetricOrder_BinaryTree(BinaryTree* Node){
  if (Node != NULL) {
    PostOrder_BinaryTree(Node->Left);
    printf ("%3ld",Node->Data);
    PostOrder_BinaryTree(Node->Right);
  }
}
```

БИНАРНЫЕ ДЕРЕВЬЯ

```
//вставка вершины в бинарное дерево
void Insert_Node_BinaryTree(BinaryTree** Node,int Data) {
  BinaryTree* New_Node = new BinaryTree;
  New_Node->Data = Data;
  New_Node->Left = NULL;
  New_Node->Right = NULL;
  BinaryTree** ptr = Node;//вспомогательный указатель
  srand(time(NULL)*1000);
```

//ВСТАВКА ВЕРШИНЫ В БИНАРНОЕ ДЕРЕВО

```
while (*ptr != NULL) {
    double q = (double) rand()/RAND_MAX;
    if ( q < 1/3.0) ptr = &((*ptr)->Left);
    else if ( q > 2/3.0) ptr = &((*ptr)->Right);
    else break;
  }
```

//ВСТАВКА ВЕРШИНЫ В БИНАРНОЕ ДЕРЕВО

```
 if (*ptr != NULL) {
    if ( (double) rand()/RAND_MAX < 0.5 )
      New_Node->Left = *ptr;
    else New_Node->Right = *ptr;
    *ptr = New_Node;
  }
  else{
    *ptr = New_Node;
  }
}
```

БИНАРНЫЕ ДЕРЕВЬЯ

```
//удаление вершины из бинарного дерева
void Delete_Node_BinaryTree(BinaryTree** Node,int Data){
  if ( (*Node) != NULL ){
    if ((*Node)->Data == Data){
      BinaryTree* ptr = (*Node);
      if ( (*Node)->Left == NULL && (*Node)->Right == NULL ) (*Node) = NULL;
      else if ((*Node)->Left == NULL) (*Node) = ptr->Right;
      else if ((*Node)->Right == NULL) (*Node) = ptr->Left;
```

//УДАЛЕНИЕ ВЕРШИНЫ ИЗ БИНАРНОГО ДЕРЕВА

```
else {
  (*Node) = ptr->Right;
  BinaryTree ** ptr1;
  ptr1 = Node;
  while (*ptr1 != NULL)
    ptr1 = &((*ptr1)->Left);
  (*ptr1) = ptr->Left;
}
```

//УДАЛЕНИЕ ВЕРШИНЫ ИЗ БИНАРНОГО ДЕРЕВА

```
delete(ptr);
      Delete_Node_BinaryTree(Node,Data);
    }
    else {
      Delete_Node_BinaryTree(&((*Node)->Left),Data);
      Delete_Node_BinaryTree(&((*Node)->Right),Data);
    }
  }
}
```

БИНАРНЫЕ ДЕРЕВЬЯ

```
        //проверка пустоты бинарного дерева
bool Empty_BinaryTree(BinaryTree* Node){
  return ( Node == NULL ? true : false );
}
```

БИНАРНЫЕ ДЕРЕВЬЯ

```
//освобождение памяти, выделенной под бинарное дерево
void Delete_BinaryTree(BinaryTree* Node){
  if (Node != NULL) {
    Delete_BinaryTree(Node->Left);
    Delete_BinaryTree(Node->Right);
    delete(Node);
  }
}
```

КЛЮЧЕВЫЕ ТЕРМИНЫ

Бинарное (двоичное) дерево – это дерево, в котором каждая вершина имеет не более двух потомков.

Вершина (узел) дерева – это каждый элемент дерева.

Ветви дерева – это направленные дуги, которыми соединены вершины дерева.

Высота (глубина) дерева – это количество уровней, на которых располагаются его вершины.

Дерево – это структура данных, представляющая собой совокупность элементов и отношений, образующих иерархическую структуру этих элементов.

КЛЮЧЕВЫЕ ТЕРМИНЫ

Корень дерева – это начальный узел дерева, ему соответствует нулевой уровень.

Листья дерева – это вершины, в которые входит одна ветвь и не выходит ни одной ветви.

Неполное бинарное дерево – это дерево, уровни которого заполнены не полностью.

Нестрогое бинарное дерево – это дерево, у которого вершины имеют степень ноль (у листьев), один или два (у узлов).

Обход дерева – это упорядоченная последовательность вершин дерева, в которой каждая вершина встречается только один раз.

КЛЮЧЕВЫЕ ТЕРМИНЫ

Поддерево – это часть древообразной структуры данных, которая может быть представлена в виде отдельного дерева.

Полное бинарное дерево – это дерево, которое содержит только полностью заполненные уровни.

Потомки – это все вершины, в которые входят ветви, исходящие из одной общей вершины.

Почти сбалансированное дерево – это дерево, у которого длины всевозможных путей от корня к внешним вершинам отличаются не более, чем на единицу.

КЛЮЧЕВЫЕ ТЕРМИНЫ

Предок – это вершина, из которой исходят ветви к вершинам следующего уровня.

Сбалансированное дерево – это дерево, у которого длины всех путей от корня к внешним вершинам равны между собой.

Степень вершины – это количество дуг, которое выходит из этой вершины.

Степень дерева – это максимальная степень вершин, входящих в дерево.

КЛЮЧЕВЫЕ ТЕРМИНЫ

Строгое бинарное дерево – это дерево, у которого вершины имеют степень ноль (у листьев) или два (у узлов).

Упорядоченное дерево – это дерево, у которого ветви, исходящие из каждой вершины, упорядочены по определенному критерию.

Уровень вершины – это количество дуг от корня дерева до вершины.

ТЕЗИСЫ ДЛЯ ЗАПОМИНАНИЯ

Деревья являются одними из наиболее широко распространенных структур данных в программировании, которые представляют собой иерархические структуры в виде набора связанных узлов.

Каждое дерево обладает следующими свойствами: существует узел, в который не входит ни одной дуги (корень); в каждую вершину, кроме корня, входит одна дуга.

С понятием дерева связаны такие понятия, как корень, ветвь, вершина, лист, предок, потомок, степень вершины и дерева, высота дерева.

ТЕЗИСЫ ДЛЯ ЗАПОМИНАНИЯ

Списочное представление деревьев основано на элементах, соответствующих вершинам дерева.

Дерево можно упорядочить по указанному ключу.

Просмотреть с целью поиска все вершины дерева можно с помощью различных способов обхода дерева.

Наиболее часто используемыми обходами являются прямой, симметричный, обратный.

ТЕЗИСЫ ДЛЯ ЗАПОМИНАНИЯ

В программировании при решении большого класса задач используются бинарные деревья.

Бинарные деревья по степени вершин делятся на строгие и нестрогие, по характеру заполнения узлов – на полные и неполные, по удалению вершин от корня – на сбалансированные и почти сбалансированные.

Основными операциями с бинарными деревьями являются: создание бинарного дерева; печать бинарного дерева; обход бинарного дерева; вставка элемента в бинарное дерево; удаление элемента из бинарного дерева; проверка пустоты бинарного дерева; удаление бинарного дерева.

ТЕЗИСЫ ДЛЯ ЗАПОМИНАНИЯ

Бинарные деревья могут применяться для поиска данных в специально построенных деревьях (базы данных), сортировки данных, вычислений арифметических выражений, кодирования.