**Цель работы:** изучение принципов построения некоторых типов таблиц и основных операций, выполняемых с ними.

**Постановка задачи:** разработать программу формирования древовидной таблицы с помощью массива. В программе предусмотреть следующие операции: добавление, удаление, поиск и получение информации об элементе.

**Описание переменных и методов**

В программе использовалось следующие переменные:

**private** TreeTableFields[] tree\_table – переменная, хранящая экзмепляр таблицы

**private** **int** root – указатель на следующую запись

**public** **boolean** write(**int** key, **int** data) – метод, реализующий добавление элементов

**public** **int** search(**int** key) – метод, реализующий поиск элементов

**public** **int**[] getInfo(**int** key) – метод, реализует возврат информации об элементе

**public** **boolean** delete(**int** key) – метод, реализующий удаление элементов

**Текст программы.**

**package** com.worm2fed.sp\_lab5;

**public** **class** TreeTable {

**private** TreeTableFields[] tree\_table;

**private** **int** root; // указатель на первую запись

TreeTable(**int** n) {

tree\_table = **new** TreeTableFields[n];

root = 0;

}

TreeTable() {

**this**(16);

}

**public** **boolean** write(**int** key, **int** data) {

**if** (root == 0)

tree\_table[root] = **new** TreeTableFields(key, data, -1, -1);

**else** **if** (root == tree\_table.length - 1)

**return** **false**;

**else** {

**if** (**this**.search(key) == key)

**return** **false**;

**else** {

**int** left = key - 1;

**int** right = key + 1;

**for** (**int** i = 0; i < tree\_table.length; i++) {

**if** (**this**.search(left) == -1) {

left--;

**if** (left == -1)

**break**;

**else**

**continue**;

} **else**

**break**;

}

**for** (**int** i = 0; i < tree\_table.length; i++) {

**if** (**this**.search(right) == -1) {

right++;

**if** (right == tree\_table.length)

**break**;

**else**

**continue**;

} **else**

**break**;

}

right = (tree\_table.length == right) ? -1 : right;

tree\_table[root] = **new** TreeTableFields(key, data, left, right);

**if** (left != -1)

tree\_table[search(left)].rlink = key;

**if** (right != -1)

tree\_table[search(right)].llink = key;

}

}

root++;

**return** **true**;

}

**public** **int** search(**int** key) {

**int** i;

**try** {

**for** (i = 0; i < root; i++) {

**if** (tree\_table[i].key != key)

**continue**;

**else**

**return** i;

}

} **catch** (NullPointerException e) {

**return** -1;

}

**return** -1;

}

**public** **int**[] getInfo(**int** key) {

**int** position = **this**.search(key);

**if** (position != -1) {

**int**[] result = { key, tree\_table[position].data,

tree\_table[position].llink, tree\_table[position].rlink };

**return** result;

} **else**

**return** **null**;

}

**public** **boolean** delete(**int** key) {

**int** position = **this**.search(key);

**if** (position != -1) {

**int** left\_pos = tree\_table[position].llink;

**int** right\_pos = tree\_table[position].rlink;

tree\_table[position] = tree\_table[root - 1];

**if** (left\_pos != -1) {

**int** right = key + 1;

**for** (**int** i = 0; i < tree\_table.length; i++) {

**if** (**this**.search(right) == -1) {

right++;

**if** (right == tree\_table.length)

**break**;

**else**

**continue**;

} **else**

**break**;

}

tree\_table[search(left\_pos)].rlink = right;

}

**if** (right\_pos != -1) {

**int** left = key - 1;

**for** (**int** i = 0; i < tree\_table.length; i++) {

**if** (**this**.search(left) == -1) {

left--;

**if** (left == -1)

**break**;

**else**

**continue**;

} **else**

**break**;

}

tree\_table[search(right\_pos)].llink = left;

}

root--;

**return** **true**;

} **else**

**return** **false**;

}

**class** TreeTableFields {

/\*

\* указатель на родителя (меньшее значения ключа) - llink указатель на

\* сына (большее значение ключа) - rlink данные - data

\*/

**public** **int** key;

**public** **int** data;

**public** **int** llink;

**public** **int** rlink;

TreeTableFields(**int** key, **int** data, **int** llink, **int** rlink) {

**this**.key = key;

**this**.data = data;

**this**.llink = llink;

**this**.rlink = rlink;

}

}

**public** **static** **void** main(String[] args) {

TreeTable tree\_table = **new** TreeTable();

tree\_table.write(8, 1);

tree\_table.write(7, 2);

tree\_table.write(3, 3);

System.***out***.println("До удаления:");

**int**[] info = tree\_table.getInfo(3);

System.***out***.println("Ключ: " + info[0]);

System.***out***.println("Данные: " + info[1]);

System.***out***.println("Эл-т слева: " + info[2]);

System.***out***.println("Эл-т справа: " + info[3]);

System.***out***.println("");

**int**[] info1 = tree\_table.getInfo(7);

System.***out***.println("Ключ: " + info1[0]);

System.***out***.println("Данные: " + info1[1]);

System.***out***.println("Эл-т слева: " + info1[2]);

System.***out***.println("Эл-т справа: " + info1[3]);

System.***out***.println("");

**int**[] info11 = tree\_table.getInfo(8);

System.***out***.println("Ключ: " + info11[0]);

System.***out***.println("Данные: " + info11[1]);

System.***out***.println("Эл-т слева: " + info11[2]);

System.***out***.println("Эл-т справа: " + info11[3]);

System.***out***.println("");

System.***out***.println("После удаления:");

tree\_table.delete(7);

**int**[] info111 = tree\_table.getInfo(3);

System.***out***.println("Ключ: " + info111[0]);

System.***out***.println("Данные: " + info111[1]);

System.***out***.println("Эл-т слева: " + info111[2]);

System.***out***.println("Эл-т справа: " + info111[3]);

System.***out***.println("");

**int**[] info1111 = tree\_table.getInfo(8);

System.***out***.println("Ключ: " + info1111[0]);

System.***out***.println("Данные: " + info1111[1]);

System.***out***.println("Эл-т слева: " + info1111[2]);

System.***out***.println("Эл-т справа: " + info1111[3]);

System.***out***.println("");

}

}

**Пример работы:**

До удаления:

Ключ: 3

Данные: 3

Эл-т слева: -1

Эл-т справа: 7

Ключ: 7

Данные: 2

Эл-т слева: 3

Эл-т справа: 8

Ключ: 8

Данные: 1

Эл-т слева: 7

Эл-т справа: -1

После удаления:

Ключ: 3

Данные: 3

Эл-т слева: -1

Эл-т справа: 8

Ключ: 8

Данные: 1

Эл-т слева: 3

Эл-т справа: -1

**Вывод:** Изучили принципы работы с базовыми табличными структурами