

## ТАБЛИЦЫ ДЛЯ РАБОТЫ С ПОЛИНОМАМИ

Запись в таблице (строка) состоит из ключа и значения. В общем случае можно создать структуру:

```
template <typename TKey, typename TValue>
struct TTableRecord {
    TKey key;
    TValue value;
};
```

Для нашей постановки: TTableRecord<std::string, Polinom>.

Список таблиц к реализации:

Неупорядоченная таблица на массиве	<p>Массив с запасом динамической памяти. Есть фактический размер памяти и реальный размер таблицы (сколько ячеек массива заполнено).</p> <pre>template &lt;typename TKey, typename TValue&gt; class TUnorderedTable {     TTableRecord&lt;typename TKey, typename TValue&gt;* data;     size_t size;     size_t count; public:     TUnorderedTable(): size(100), count(0) { /* ... */};     /* ... */ };</pre>
Неупорядоченная таблица на списке	<p>В отличие от таблицы на массиве, не нужно реализовывать выделение и перевыделение памяти. Данные хранятся в динамически расширяемом списке.</p> <pre>template &lt;typename TKey, typename TValue&gt; class TUnorderedTable {     TList&lt;TTableRecord&lt;typename TKey, typename TValue&gt;&gt; data; public:     TUnorderedTable() = default;     /* ... */ };</pre>
Упорядоченная таблица на массиве	<p>Массив с запасом динамической памяти. Есть фактический размер памяти и реальный размер таблицы (сколько ячеек массива заполнено).</p> <p>Порядок следования элементов в таблице определяется упорядоченностью ключей.</p>
Таблица на авл-дереве	<p>Мы изучали на занятиях бинарные деревья (дерево поиска и дерево арифметического выражения). Они могли превращаться в вырожденные из-за добавления всех элементов в одну длинную ветвь.</p> <p>Решение проблемы -реализация сбалансированного дерева. Дерево называется идеально сбалансированным, если для каждой его вершины количество вершин в левом и правом поддеревьях отличается не более, чем на единицу.</p>

	<p>Дерево поиска является АВЛ-деревом, если для каждой его вершины высота левого и правого поддеревьев отличается не более, чем на единицу.</p> <p>Служебное поле: показатель баланса – разность высот правого и левого поддеревьев.</p>
Хеш-таблица	<p>В основе работы с данными лежит принцип вычислимости адресов элементов в памяти согласно какому-то алгоритму (хеш-функция).</p> <p>Функция, отображающая значение ключа в индекс строки, называется функцией хеширования (расстановки, перемешивания).</p> <p>Таблицы, в которых данные размещены на основе хеш-функции, называются хеш-таблицами (таблицами с вычислимыми адресами).</p> <p>Например, простейший вариант использования хеш-функции <math>h(k) = k \bmod M</math>, где <math>M</math> размер таблицы.</p>

вставка

Неупорядоченная таблица на массиве

1. Вызвать поиск. Если уже есть – выдать исключение.

2. Если ключ не занят, добавить в конец таблицы новую запись. Если таблица переполнена – увеличить её реальный размер (перевыделить память).

Неупорядоченная таблица на списке

1. Вызвать поиск. Если уже есть – выдать исключение.

2. Если ключ не занят, добавить в конец таблицы новую запись (метод вставки в конец списка).

Упорядоченная таблица на массиве

1. Проверить, есть ли в таблице данные с таким ключом. Если уже есть – выдать исключение.

2. Если ключ не занят, то добавить по найденной в процессе поиска таблицы новый элемент:

- если таблица переполнена – увеличить её реальный размер (перевыделить память),

- вставить новый элемент на найденную ранее позицию, сохранить элемент, который был в этом месте,

- произвести перепакровку.

Пример: вставляем {name3, ...}

{name1,...}

{pol3, ...}

{pol5, ...}

{pol6, ...}

{qqq1, ...}

0

1

2

3

4

5

6

7

{name1, ...}

{pol3, ...}

{pol5, ...}

{pol6, ...}

{qqq1, ...}

0

1

2

3

4

5

6

7

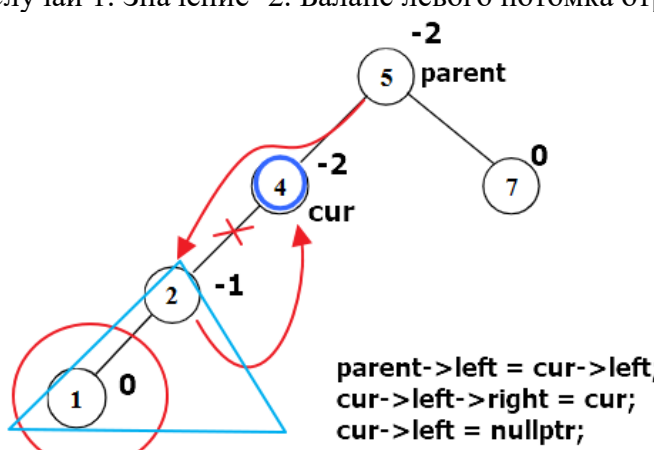
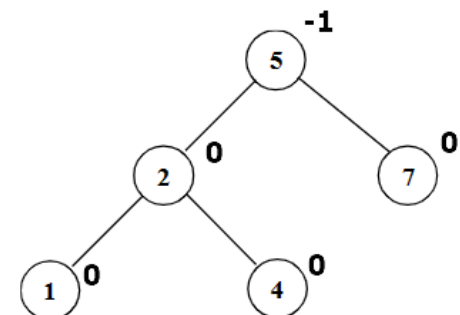
{name1, ...}

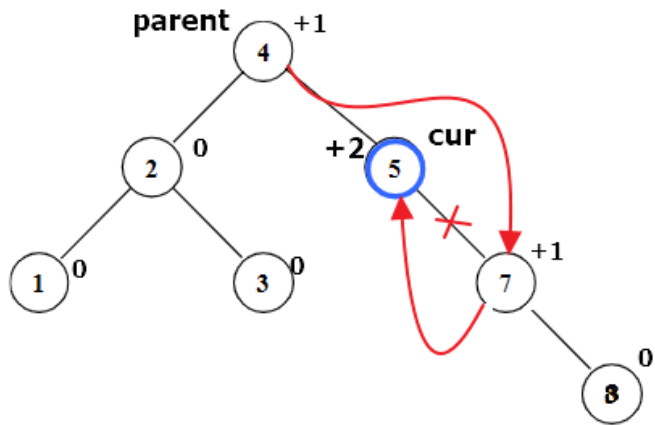
{pol3, ...}

{pol5, ...}

{pol6, ...}

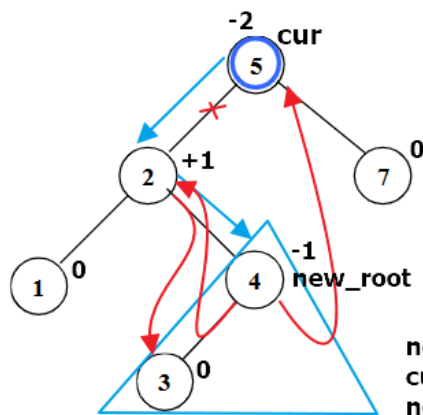
{qqq1, ...}

	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>{name1, ...}</td><td>{name3, ...}</td><td>{pol5, ...}</td><td>{pol6, ...}</td><td>{qqq1, ...}</td><td></td><td></td><td></td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr></table> <p>Save: {pol3, ...}</p> <table><tr><td>{name1, ...}</td><td>{name3, ...}</td><td>{pol5, ...}</td><td>{pol5, ...}</td><td>{pol6, ...}</td><td>{qqq1, ...}</td><td></td><td></td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr></table> <table><tr><td>{name1, ...}</td><td>{name3, ...}</td><td>{pol3, ...}</td><td>{pol5, ...}</td><td>{pol6, ...}</td><td>{qqq1, ...}</td><td></td><td></td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr></table>	0	1	2	3	4	5	6	7	{name1, ...}	{name3, ...}	{pol5, ...}	{pol6, ...}	{qqq1, ...}				0	1	2	3	4	5	6	7	{name1, ...}	{name3, ...}	{pol5, ...}	{pol5, ...}	{pol6, ...}	{qqq1, ...}			0	1	2	3	4	5	6	7	{name1, ...}	{name3, ...}	{pol3, ...}	{pol5, ...}	{pol6, ...}	{qqq1, ...}			0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7																																																		
{name1, ...}	{name3, ...}	{pol5, ...}	{pol6, ...}	{qqq1, ...}																																																					
0	1	2	3	4	5	6	7																																																		
{name1, ...}	{name3, ...}	{pol5, ...}	{pol5, ...}	{pol6, ...}	{qqq1, ...}																																																				
0	1	2	3	4	5	6	7																																																		
{name1, ...}	{name3, ...}	{pol3, ...}	{pol5, ...}	{pol6, ...}	{qqq1, ...}																																																				
0	1	2	3	4	5	6	7																																																		
Таблица на авл-дереве	<p>1. Осуществить классический поиск в бинарном дереве поиска с запоминанием пройденного пути.</p> <p>2. Если элемент найден, то – исключение. Иначе – добавить новый элемент на найденную позицию. Пройти обратно по полученному пути, пересчитывая баланс вершины.</p> <p>При обнаружении +2 или -2 – перебалансировать дерево.</p> <p><b>ОДНОКРАТНЫЕ ПОДЪЁМЫ</b></p> <p>Случай 1. Значение -2. Баланс левого потомка отрицательный.</p> <div></div> <div></div> <p>Случай 2. Значение +2. Баланс правого потомка положительный.</p>																																																								

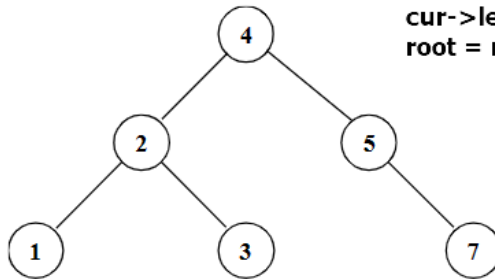


### ДВУКРАТНЫЕ ПОДЪЁМЫ

Случай 3. Значение -2. Баланс левого потомка положительный.



```
new_root = cur->left->right;
cur->left->right = new_root->left;
new_root->left = cur->left;
new_root->right = cur;
cur->left = nullptr;
root = new_root;
```



Случай 4. Значение +2. Баланс правого потомка отрицательный.

	<div><div><div><div><div>parent</div><div>4</div><div>+1</div></div><div><div>2</div><div>0</div></div><div><div>1</div><div>0</div></div><div><div>3</div><div>0</div></div><div><div>5</div><div>+2</div></div><div>cur</div><div><div>7</div><div>-1</div></div><div><div>6</div><div>0</div></div><div>new_root</div></div><div><div><div>4</div><div>2</div><div>1</div><div>3</div><div>6</div><div>5</div><div>7</div></div><div><pre>new_root = cur-&gt;right-&gt;left; cur-&gt;right-&gt;left = new_root-&gt;right; new_root-&gt;right = cur-&gt;right; new_root-&gt;left = cur; cur-&gt;right = nullptr; parent-&gt;right = new_root;</pre></div></div></div></div>																
Хеш-таблица с открытым перемешиванием	<div><div><p>Вставляем элемент</p><p>key = "pol"; k = 'p'+ 'o'+ 'l'; // = 112+111+108=331</p><p>h(k) = 11 – адрес ячейки для вставки.</p><p>key = "ned"; k = 'n'+ 'e'+ 'd'; // = 110+101+100=311</p><p>h(k) = 11 - пример коллизии при вставке, позиция уже занята.</p><p>Решение: повторное перемешивание с шагом перемешивания, удовлетворяющим <math>1 \leq h &lt; M</math>, <math>\text{НОД}(h, M) = 1</math>:</p><p><math>hh(k) = (h(k) + h) \bmod M</math></p><p>Значение шага перемешивания может быть любым, но обязательно взаимно простым с величиной, определяющей размер таблицы, чтобы обеспечить просмотр всех ее позиций.</p><p><math>hh(k) = (11 + 3) \bmod 20 = 14</math></p><table><tr><td>0</td><td></td></tr><tr><td>...</td><td></td></tr><tr><td>11</td><td>"pol"</td></tr><tr><td>12</td><td></td></tr><tr><td>13</td><td></td></tr><tr><td>14</td><td>"ned"</td></tr><tr><td>...</td><td></td></tr><tr><td>19</td><td></td></tr></table><p>Если таблица еще не переполнена:</p><div><div>1. Вычислить значение функции хеширования для заданного ключа.</div><div>2. Проверить, статус найденной позиции для вставки:<div><div>- если позиция <b>занята</b> и <b>ключ совпадает</b> с ее значением, выбросить исключение (дублирование), выход;</div><div>- если позиция <b>свободна</b>, вставить элемент по указанной позиции, изменить статус на занята, выход;</div><div>- если позиция <b>удалена</b>, вставить элемент по найденной позиции, изменить статус на занята, выход;</div></div></div><div>3. Вызвать повторное перемешивание hh(k). Вернуться к шагу 2.</div></div></div></div>	0		...		11	"pol"	12		13		14	"ned"	...		19	
0																	
...																	
11	"pol"																
12																	
13																	
14	"ned"																
...																	
19																	

	Хеш-таблица со списками (метод цепочек)	<p>Второй способ разрешения коллизий: метод цепочек.</p> <p>В методе цепочек хеш-таблица представляет собой массив из <math>M</math> списков. В каждом списке находятся записи с одинаковым значением хеш-функции.</p> <p>Переполнение таблицы в данном варианте невозможно.</p> <ol style="list-style-type: none"> <li>1. Вычислить значение функции хеширования для заданного ключа.</li> <li>2. Пройти по расположенному по данному адресу списку: <ul style="list-style-type: none"> <li>- если найден такой же ключ, выбросить исключение (дублирование);</li> <li>- иначе вставить в конец списка.</li> </ul> </li> </ol>
удаление	Неупорядоченная таблица на массиве	<ol style="list-style-type: none"> <li>1. Вызвать поиск.</li> <li>2. При нахождении заданного ключа: <ul style="list-style-type: none"> <li><i>1 способ:</i> сместить все данные вверх, уменьшить реальный размер таблицы;</li> <li><i>2 способ:</i> поставить на место удаляемой строки последнюю строку таблицы, уменьшить реальный размер таблицы.</li> </ul> </li> </ol>
	Неупорядоченная таблица на списке	<ol style="list-style-type: none"> <li>1. Вызвать поиск.</li> <li>2. При нахождении заданного ключа вызвать метод удаления элемента из списка по указанной позиции.</li> </ol>
	Упорядоченная таблица на массиве	<ol style="list-style-type: none"> <li>1. Вызвать поиск.</li> <li>2. При нахождении заданного ключа сместить все данные вверх, уменьшить реальный размер таблицы. Можно вызвать перепакровку, если она реализована подходящим образом.</li> </ol>
	Таблица на AVL-дереве	<ol style="list-style-type: none"> <li>1. Вызвать поиск.</li> <li>2. При нахождении заданного ключа, если это лист – удалить, иначе найти самую близкую по значению, переместить её на место удаляемой вершины, удалить в старом месте.</li> <li>3. Пересчитать баланс от удаленной вершины до корня. Отбалансировать при необходимости.</li> </ol>
	Хеш-таблица с открытым перемешиванием	<p>Каждой ячейке добавляем статус (значения: свободна, удалена, занята).</p> <ol style="list-style-type: none"> <li>1. Вызвать поиск.</li> <li>2. Если найдена, установить статус удалена.</li> </ol>
	Хеш-таблица со списками (метод цепочек)	<ol style="list-style-type: none"> <li>1. Вычислить значение функции хеширования для заданного ключа.</li> <li>2. Пройти по расположенному по данному адресу списку: <ul style="list-style-type: none"> <li>- если ключ найден, удалить элемент из списка;</li> <li>- иначе выбросить исключение.</li> </ul> </li> </ol>
поиск	Неупорядоченная таблица на массиве	<ol style="list-style-type: none"> <li>1. Идем по массиву, сравнивая ключи.</li> <li>2. При нахождении заданного ключа вернуть указатель на найденные данные.</li> </ol>
	Неупорядоченная таблица на списке	<ol style="list-style-type: none"> <li>1. Идем по списку, сравнивая ключи.</li> <li>2. При нахождении заданного ключа вернуть указатель на найденные данные.</li> </ol>
	Упорядоченная таблица на массиве	Реализовать бинарный поиск по ключу.

	Таблица наavl-дереве	Поиск в бинарном дереве поиска.
	Хеш-таблица с открытым перемешиванием	<ol style="list-style-type: none"> <li>1. Вычислить значение функции хеширования для заданного ключа.</li> <li>2. Проверить, статус найденной позиции для вставки: <ul style="list-style-type: none"> <li>- если позиция свободна, такого элемента нет, выход;</li> <li>- если позиция удалена или занята и ключ не совпадает со значением, дальше;</li> <li>- иначе – найден ответ, выход.</li> </ul> </li> <li>3. Вызвать повторное перемешивание <math>hh(k)</math>. Вернуться к шагу 2.</li> </ol>
	Хеш-таблица со списками (метод цепочек)	<ol style="list-style-type: none"> <li>1. Вычислить значение функции хеширования для заданного ключа.</li> <li>2. Пройти по расположенному по данному адресу списку: <ul style="list-style-type: none"> <li>- если ключ найден – вернуть по указателю или ссылке найденный элемент;</li> <li>- иначе элемента нет.</li> </ul> </li> </ol>