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;
};
```

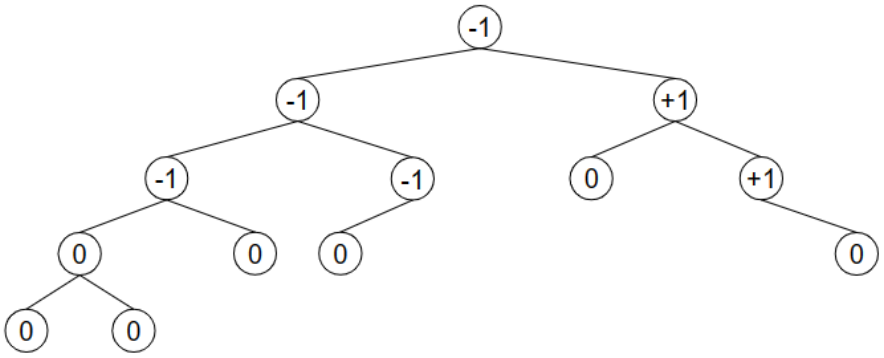
Для нашей постановки: `TTableRecord<std::string, Polinom>`.

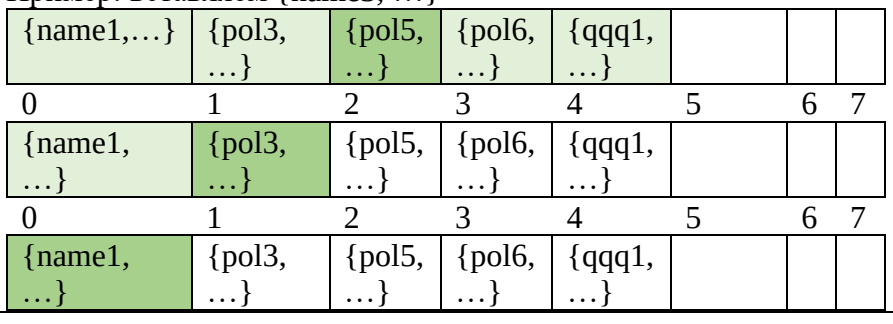
Список таблиц к реализации:

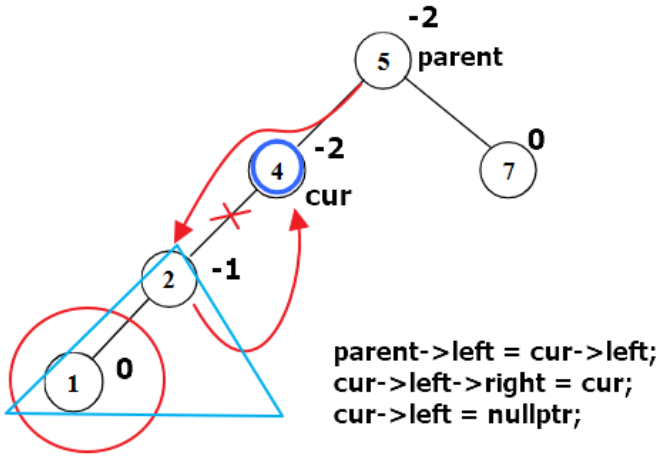
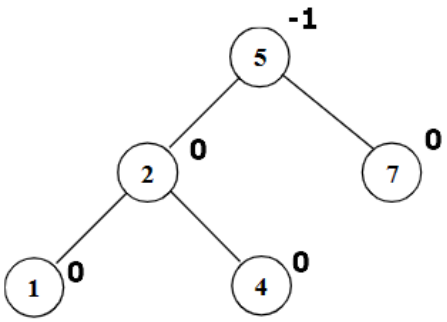
Неупорядоченная таблица на массиве	Массив с запасом динамической памяти. Есть фактический размер памяти и реальный размер таблицы (сколько ячеек массива заполнено).
	`template <typename TKey, typename TValue>`
	`class TUnorderedTable {`
	`  TTableRecord<typename TKey, typename TValue>* data;`
	`  size_t size;`
	`  size_t count;`
	`public:`
	`  TUnorderedTable(): size(100), count(0) { /* … */};`
	`  /* … */`
	`};`

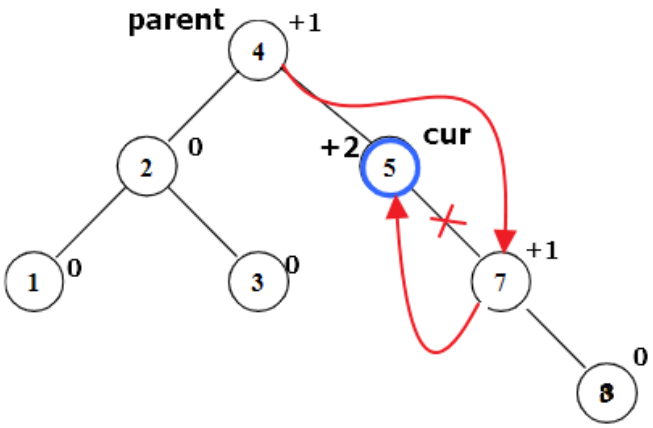
Неупорядоченная таблица на списке	В отличие от таблицы на массиве, не нужно реализовывать выделение и перевыделение памяти. Данные хранятся в динамически расширяемом списке.
	`template <typename TKey, typename TValue>`
	`class TUnorderedTable {`
	`  TList<TTableRecord<typename TKey, typename TValue>> data;`
	`public:`
	`  TUnorderedTable() = default;`
	`  /* … */`
	`};`

Упорядоченная таблица на массиве	Массив с запасом динамической памяти. Есть фактический размер памяти и реальный размер таблицы (сколько ячеек массива заполнено). Порядок следования элементов в таблице определяется упорядоченностью ключей.
Таблица на авл-дереве	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. 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.

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

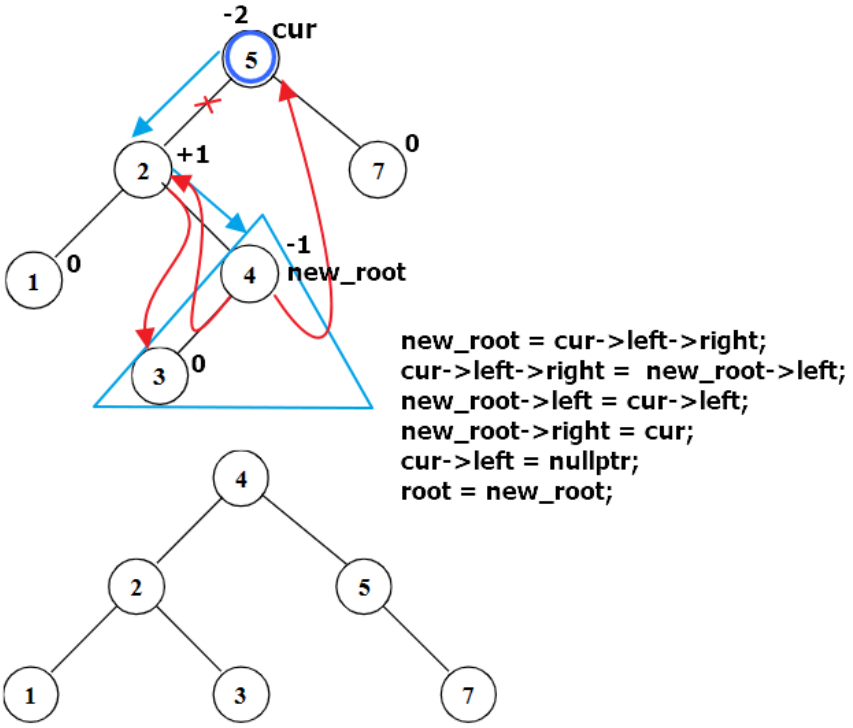
вставка	Неупорядоченная таблица на массиве	1. Вызвать поиск. Если уже есть – выдать исключение. 2. Если ключ не занят, добавить в конец таблицы новую запись. Если таблица переполнена – увеличить её реальный размер (перевыделить память).
	Неупорядоченная таблица на списке	1. Вызвать поиск. Если уже есть – 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 перепаковку. Пример: вставляем {name3, …} 

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

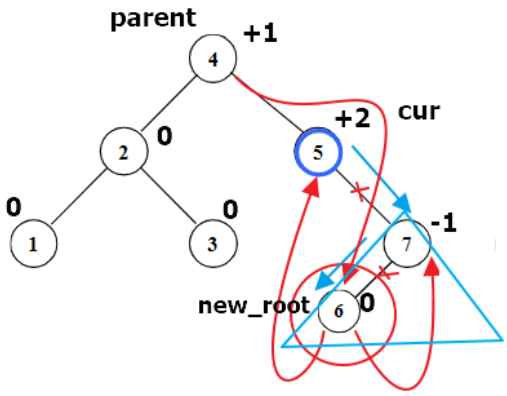


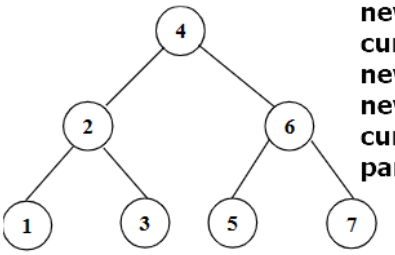
ДВУКРАТНЫЕ ПОДЪЁМЫ

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



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

<table>
<tr>
<td></td>
<td></td>
<td>




<pre>new_root = cur->right->left;
cur->right->left = new_root->right;
new_root->right = cur->right;
new_root->left= cur;
cur->right = nullptr;
parent->right = new_root;</pre>
</td>
</tr>
<tr>
<td></td>
<td>Хеш-таблица с открытым перемешиванием</td>
<td>
Вставляем элемент
key = “pol”; k = ‘p’+’o’+’l’; // = 112+111+108=331
h(k) = 11 – адрес ячейки для вставки.
key = “ned”; k = ‘n’+’e’+’d’; // = 110+101+100=311
h(k) = 11 - пример коллизии при вставке, позиция уже занята.
Решение: повторное перемешивание с шагом перемешивания, удовлетворяющим $1 \le h < M$, НОД(h, M) = 1:
$hh(k) = (h(k) + h)\ mod\ M$
Значение шага перемешивания может быть любым, но обязательно взаимно простым с величиной, определяющей размер таблицы, чтобы обеспечить просмотр всех ее позиций.
hh(k) = (11 + 3) mod 20 =14
<table>
<tr><td>0</td><td></td></tr>
<tr><td colspan="2">…</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 colspan="2">…</td></tr>
<tr><td>19</td><td></td></tr>
</table>
Если таблица еще не переполнена:

Вычислить значение функции хеширования для заданного ключа.
Проверить, статус найденной позиции для вставки:
- если позиция занята и ключ совпадает с ее значением, выбросить исключение (дублирование), выход;
- если позиция свободна, вставить элемент по указанной позиции, изменить статус на занята, выход;
- если позиция удалена, вставить элемент по найденной позиции, изменить статус на занята, выход;
Вызвать повторное перемешивание hh(k). Вернуться к шагу 2.

</td>
</tr>
</table>

	Хеш-таблица со списками (метод цепочек)	Второй способ разрешения коллизий: метод цепочек. В методе цепочек хеш-таблица представляет собой массив из *M* списков. В каждом списке находятся записи с одинаковым значением хеш-функции. Переполнение таблицы в данном варианте невозможно. 1. Вычислить значение функции хеширования для заданного ключа. 2. Пройти по расположенному по данному адресу списку: - если найден такой же ключ, выбросить исключение (дублирование); - иначе вставить в конец списка.
удаление	Неупорядоченная таблица на массиве	1. Вызвать поиск. 2. При нахождении заданного ключа: *1 способ:* сместить все данные вверх, уменьшить реальный размер таблицы; *2 способ:* поставить на место удаляемой строки последнюю строку таблицы, уменьшить реальный размер таблицы.
	Неупорядоченная таблица на списке	1. Вызвать поиск. 2. При нахождении заданного ключа вызвать метод удаления элемента из списка по указанной позиции.
	Упорядоченная таблица на массиве	1. Вызвать поиск. 2. При нахождении заданного ключа сместить все данные вверх, уменьшить реальный размер таблицы. Можно вызвать перепаковку, если она реализована подходящим образом.
	Таблица на авл-дереве	1. Вызвать поиск. 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. 3. Пересчитать баланс от удаленной вершины до корня. Отбалансировать при необходимости.
	Хеш-таблица с открытым перемешиванием	Каждой ячейке добавляем статус (значения: свободна, удалена, занята). 1. Вызвать поиск. 2. Если найдена, установить статус удалена.
	Хеш-таблица со списками (метод цепочек)	1. Вычислить значение функции хеширования для заданного ключа. 2. Пройти по расположенному по данному адресу списку: - если ключ найден, удалить элемент из списка; - иначе выбросить исключение.
поиск	Неупорядоченная таблица на массиве	1. Идем по массиву, сравнивая ключи. 2. При нахождении заданного ключа вернуть указатель на найденные данные.
	Неупорядоченная таблица на списке	1. Идем по списку, сравнивая ключи. 2. При нахождении заданного ключа вернуть указатель на найденные данные.
	Упорядоченная таблица на массиве	Реализовать бинарный поиск по ключу.

	Таблица на авл-дереве	Поиск в бинарном дереве поиска.
	Хеш-таблица с открытым перемешиванием	1. Вычислить значение функции хеширования для заданного ключа. 2. Проверить, статус найденной позиции для вставки: - если позиция свободна, такого элемента нет, выход; - если позиция удалена или занята и ключ не совпадает со значением, дальше; - иначе – найден ответ, выход. 3. Вызвать повторное перемешивание hh(k). Вернуться к шагу 2.
	Хеш-таблица со списками (метод цепочек)	1. Вычислить значение функции хеширования для заданного ключа. 2. Пройти по расположенному по данному адресу списку: - если ключ найден – вернуть по указателю или ссылке найденный элемент; - иначе элемента нет.