

Часть 1.

Отображение.

Я встречал этот тип данных, во время работы на perl, назывался ассоциативным массивом (associative array).

```
sub main
{
    my %hobbies = (
        'Roger' => 'hang gliding',
        'Penny' => 'diving',
        'Peter' => 'bus surfing',
        'Richard' => 'collects spores and fungi',
        'Clare' => 'competitive drinking',
        'Lisa' => 'pole vaulting',
    );
    $hobbies{'John'} = 'running';
    delete $hobbies{'Peter'};
    print "Richard's hobby: ", $hobbies{'Richard'}, "\n";
    use Data::Dumper;
    print Dumper(%hobbies);
}

main();
```

Красота.

И при работе с lua, что уже не так красиво:

<https://github.com/acmeism/RosettaCodeData/blob/948b86eafab0e034330a3b6c31617370c6cca2fc/Task/Associative-array-Creation/Lua/associative-array-creation.lua>

```
hash = {}
```

```
hash[ "key-1" ] = "val1"
```

```
hash[ "key-2" ] = 1
```

```
hash[ "key-3" ] = {}
```

Язык—>	Java	Python
название контейнера—>	Map	dict (dictionary)
Пример—>	<pre> Map<String, String> map = new HashMap<String, String>(); map.put("name", "demo"); map.put("fname", "fdemo"); // etc map.get("name"); // returns "demo" </pre>	<pre> dict([('sape', 4139), ('guido', 4127), ('jack', 4098)]) {'sape': 4139, 'jack': 4098, 'guido': 4127} или >>> tel = {'jack': 4098, 'sape': 4139} >>> tel['guido'] = 4127 >>> tel {'sape': 4139, 'guido': 4127, 'jack': 4098} >>> tel['jack'] 4098 >>> del tel['sape'] >>> tel['irv'] = 4127 >>> tel {'guido': 4127, 'irv': 4127, 'jack': 4098} >>> tel.keys() ['guido', 'irv', 'jack'] >>> 'guido' in tel True СТАТЬЯ С ПОНЯТНЫМ ЯЗЫКОМ https://habr.com/post/112421/ </pre>

ЯЗЫК—>	C++	Haskell
название контейнера—>	std::map	Data.Map
Пример—>	<pre> int main() { map <string,int> myFirstMap = {{ "Mother", 37 }, { "Father", 40 },//map явно инициализирована { "Brother", 15 }, { "Sister", 20 }}; ///ВЫВОД ЯВНО инициализированной map на экран for (auto it = myFirstMap.begin(); it != myFirstMap.end(); ++it) { cout << it->first << " : " << it->second << endl; } char c; map <char,int> mySecondMap; for (int i = 0,c = 'a'; i < 5; ++i,++c) { mySecondMap.insert (pair<char,int>(c,i)); } ///ВЫВОД не ЯВНО инициализированной map на экран for (auto it = mySecondMap.begin(); it ! = mySecondMap.end(); + +it) { cout << (*it).first << " : " << (*it).second << endl; } return 0; </pre>	<p>https://hackage.haskell.org/package/containers-0.5.8.1/docs/src/Data.Map.Base.html#line-3873</p> <pre> countryCurrency = fromList([("USA", "Dollar"), ("France", "Euro")]) </pre>

Дерево.

ЯЗЫК—>	Java	Python
название контейнера—>	класс	класс
Пример—>	<pre>public class TreeNode<T> implements Iterable<TreeNode<T>> { T data; TreeNode<T> parent; List<TreeNode<T>> children; public TreeNode(T data) { this.data = data; this.children = new LinkedList<TreeNode<T>>(); } public TreeNode<T> addChild(T child) { TreeNode<T> childNode = new TreeNode<T>(child); childNode.parent = this; this.children.add(childNo de); return childNode; } // other features ... }</pre>	<pre>class Tree: def __init__(self, left, right): self.left = left self.right = right >>> t = Tree(Tree("a", "b"), Tree("c", "d")) >>> t.right.left 'c'</pre> <p>и снова статья понятным языком: https://habr.com/post/112421/</p>

язык—>	C++	Haskell
название контейнера—>	класс	data
Пример—>	http://ci-plus-plus-snachala.ru/?p=89 ну, или https://github.com/yokkidack/BSTree	<pre> data Tree a = Empty Node a [Tree a] deriving (Show) instance Functor Tree where fmap f Empty = Empty fmap f (Node a ys) = Node (f a) (map (fmap f) ys) treeHeight :: Tree a -> Integer treeHeight Empty = 0 treeHeight (Node a b) = 1 + maximum (fmap treeHeight b) treeSum :: (Num a) => Tree a -> a treeSum (Node a b) = a + sum(fmap treeSum b) --treeFold :: (b -> (a, [b])) -> b -> Tree a treeFold f b = let (a, bs) = f b in Node a [treeFold f bs] </pre>

Список.

язык—>	Java	Python
название контейнера—>	<ul style="list-style-type: none"> • java.util.ArrayList • java.util.LinkedList • java.util.Vector • java.util.Stack 	list
Пример—>	http://tutorials.jenkov.com/java-collections/list.html	<pre> l = ['s', 'p', ['isok'], 2] </pre>

язык—>	C++	Haskell
название контейнера—>	std::list	Data.List
Пример—>	http://ru.cppreference.com/w/cpp/container/list	(++) :: [a] -> [a] -> [a]

ИСТОЧНИКИ:
http://www.cyberforum.ru/haskell/thread879831.html
http://ru.cppreference.com/w/cpp/container/list
http://tutorials.jenkov.com/java-collections/list.html
https://pythonworld.ru/typy-dannyx-v-python/spiski-list-funkcii-i-metody-spiskov.html
http://cppstudio.com/post/9535/
http://fkn.ktu10.com/?q=node/5877
https://www.cs.cmu.edu/Groups/AI/html/cltl/clm/node153.html
https://mail.haskell.org/pipermail/beginners/2012-August/010443.html
https://hackage.haskell.org/package/containers-0.5.8.1/docs/Data-Map-Strict.html
http://learnyouahaskell.com/making-our-own-types-and-typeclasses
https://hackage.haskell.org/package/containers-0.5.8.1/docs/src/Data.Map.Base.html#line-3873
https://docs.python.org/2/tutorial/datastructures.html
http://qaru.site/questions/16983/java-tree-data-structure
https://habr.com/post/112421/
http://ci-plus-plus-snachala.ru/?p=89
https://github.com/yokkidack/BSTree