Introduction to the C++ Language

Samuel Navarro

May 4, 2019

A vector is a linear sequence of contiguously allocated memory. There are linked lists. Hash tables and the maps: Which are collections of data that skips order which makes them much faster to look up in them than in other data structures.

1 Vector

In the case of vectors, we can print them with this:

The important thing here was that, in the *range-based* loop, we put the type of the element we are *iterating*. We could also use de keyword `auto`.

We want to make a function which accept a `vector` of `int` as its argument and return the sum of all the `int`s in the vector.

When we mark the elements as `const` you are declaring that you are going to just read the variable. You are not going to change it.

2 Containers

From Bjarne: The most useful is the vector because most of the standard algorithms work on it. The number 2 for most people is that they want to build a look up table (a map, a dictionary) and if that's for ordinary strings, you can probably could use a **hash table** they are called unordered maps and if they are anything else, you want a red black tree which is called a map.

Container data structures are fantastic for storing ordered data, and classes are useful for grouping related data and functions together, but neigher of these data structures is optimal for storing associated data.

Dictionary Example

A hash table (alternatively has map, or dictionary) is a data structure that uses *key/value* pairs to store data, and provides efficient lookup and insertion of the data.

```cpp
                #include <iostream>
                #include <vector>
                using std::vector;
                using std::cout;

                int main(){
                        vector<int> v{6, 7, 8};
                        for(int i : v){
                                cout << i << "\n";
                        }
                }
```

Listing 1: caption name

```cpp
        #include <iostream>
        #include <vector>
        using std::cout;
        using std::vector;

        int main() {
                // Add your code here.
                vector<vector<int>> b {{1, 2},
                                                        {3, 4},
                                                        {5, 6}};

                // Write your double loop here.
                for(vector i : b){
                        for(int j : i){
                                cout << j << "\n";
                        }
                }
        }
```

Listing 2: vector

```
        int AdditionFunction(vector<int> v){
                int sum = 0;
                for (int i : v){
                        sum += i;
                }
                return sum;
        }
```

Listing 3: Sum elements of vector