Data Structures in C++

Array

An array is a data structure which allows for storage of specific number of elements. This can be changed however, if this is a requirement then a different data structure should be chosen.

Arrays a quick and easy use, offering fast element access and easy to iterate over using a simple for loop.

Arrays however do not have a lot of built in functionality.

Vector

Vectors are very similar to arrays. Vectors are essentially dynamic arrays. Vectors should be used when an array is ideal but with frequent size manipulation required.

Due to this specific purpose, vectors offer a great deal more functionality than arrays. Most notably, vectors have both insert and remove. This will dynamically adjust the size of the vector as more or less space is required.

List

Lists are a data structure in which each element in the list is connected to the previous and next element only. This makes the addition and deletion of objects very fast with the use of pointers. To add an object in a list, you can squeeze it in. With an array, all other values need to be moved.

However, this means that retrieving elements is slower as you need to iterate through the list until you reach the element you want. This is a lot slower than using the element access in a vector or an array.

Lists offer even more functionality than arrays with the addition of reverse and sort built in.

MultiMap

Maps are a sorted data structure which feels similar to an array. Data is stored as a key-value pair. The value is then accessed through use of the key.

One advantage to a Multimap over a regular map is that the same key can be used more than once.

Maps do offer very fast retrieval of values through clever sorting techniques.

Maps offer similar functionality to vectors.

Conclusion

Based on this short analysis, I have chosen to implement a vector for my inventory. This is because it is easy to use and offers the amount of functionality I believe is required.