41012 Programming for Mechatronic Systems

Class: Week 5

Introduction

- On Thursday 13th April
  - Assignment 1 Feedback
  - Assignment 2 Task Description

TODAY

- Templates – Generic Programming
  - Template Functions
- Standard Template Library
  - Containers
  - Parallels to Pointers / Arrays

Template Functions

Template declarations preceded by the template keyword and a set of type identifiers

```
template <typename T>
T maxof (T a, T b) {
        return a>b ? a : b)
}
```

Template Classes / Significantly More Advanced

Standard Template Library (STL)

- Provides containers and supporting data types:
  - Vector, Lists, Queues, Iterators

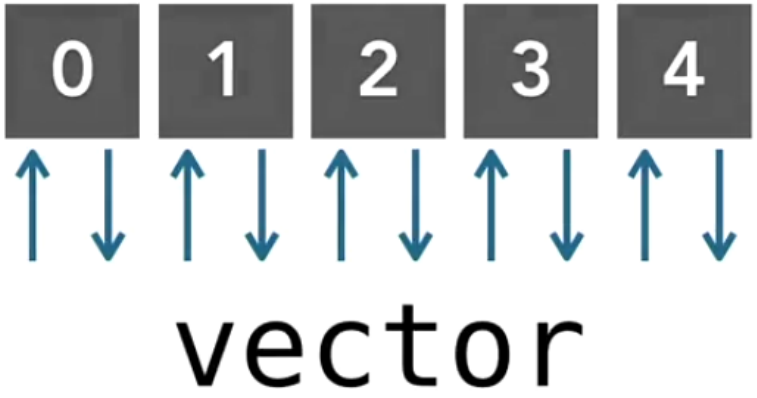
- STL Completes the dynamic memory handling
- Allows to be used as a container for any data type (Class / Structure)
- Iterator similar to Pointer

http://www.cplusplus.com/reference/stl/

Vector

- Provides random access to elements
  - Can change in size (dynamically allocated)
  - http://www.cplusplus.com/reference/vector/vector/

STL

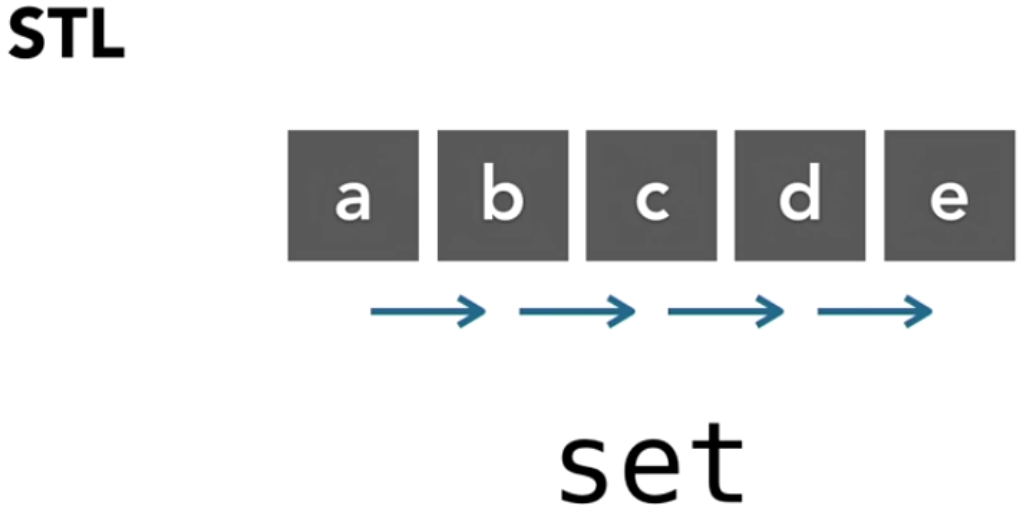


List

- Double Linked List – Optimised for Access
  - Constant time insert and erase operations
  - http://www.cplusplus.com/reference/list/list/

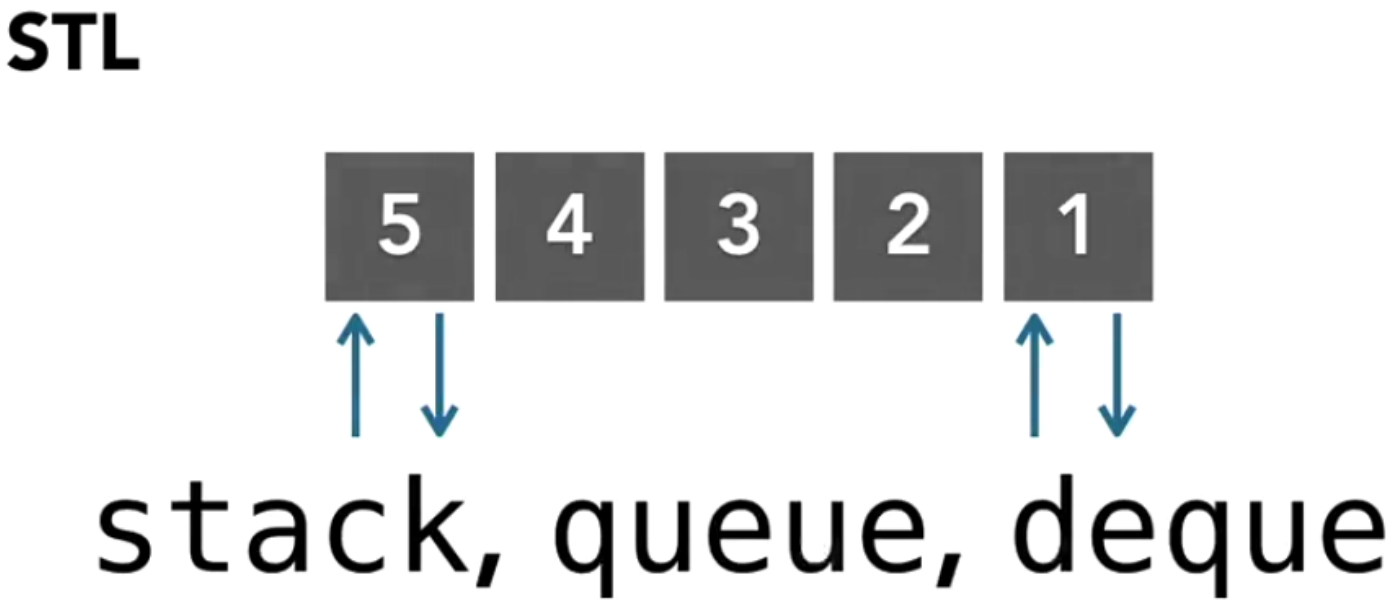
Set

- Ordered Sequence based on value
  - store unique elements following a specific order.
  - http://www.cplusplus.com/reference/set/set/



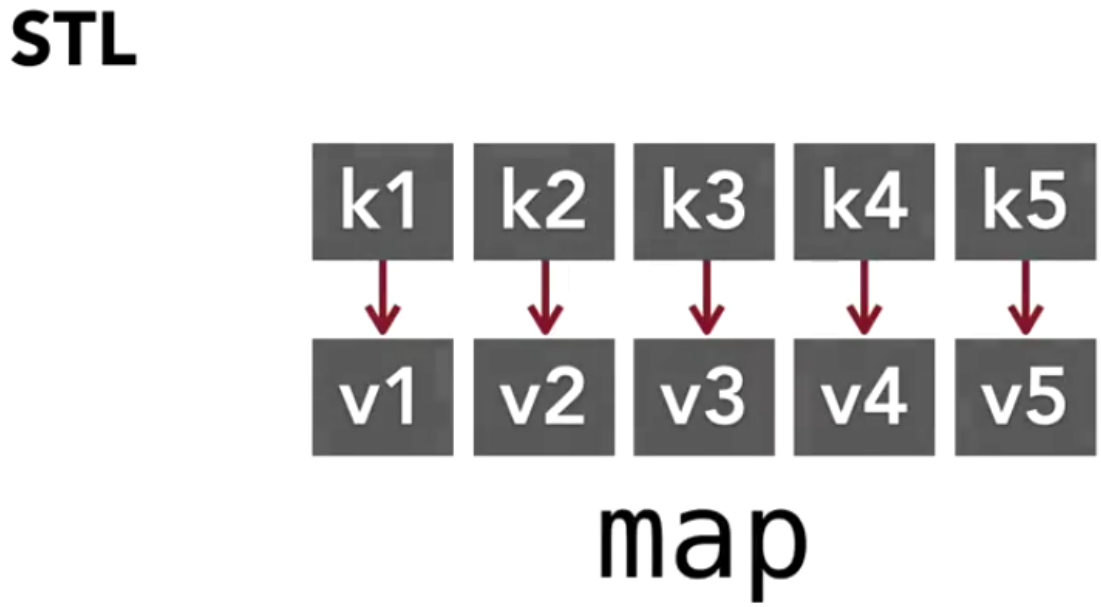
Stack / Queue / Deque

- LIFO / FIFO Access



Map

- Keys used to access associated values
  - http://www.cplusplus.com/reference/map/map/



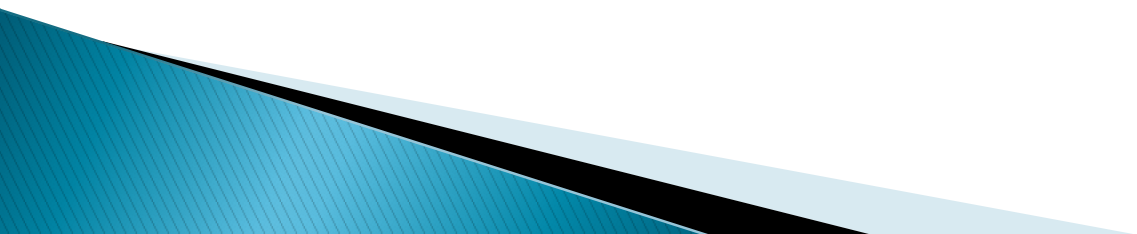
Iterators (example of vector)

```
vi1.size()
vi1.front()
vi1.back()
vi1[5]  // vi1.at(5)

vi1.insert(vi1.begin() + 5, 42);
vi1.erase(vi1.begin() + 5);
vi1.push_back(47);

vector<int>::iterator itbegin = vi1.begin();
vector<int>::iterator itend = vi1.end();
for (auto it = itbegin; it < itend; ++it) {
        cout << *it << ' ';
}   cout << endl;
```

STL selection

- Sorting Enforced
- Access (random / ordered)
- Keys -> values
- Size?

- Optimal … yes
- Interchangeable … not always, though yes, such is life of templates

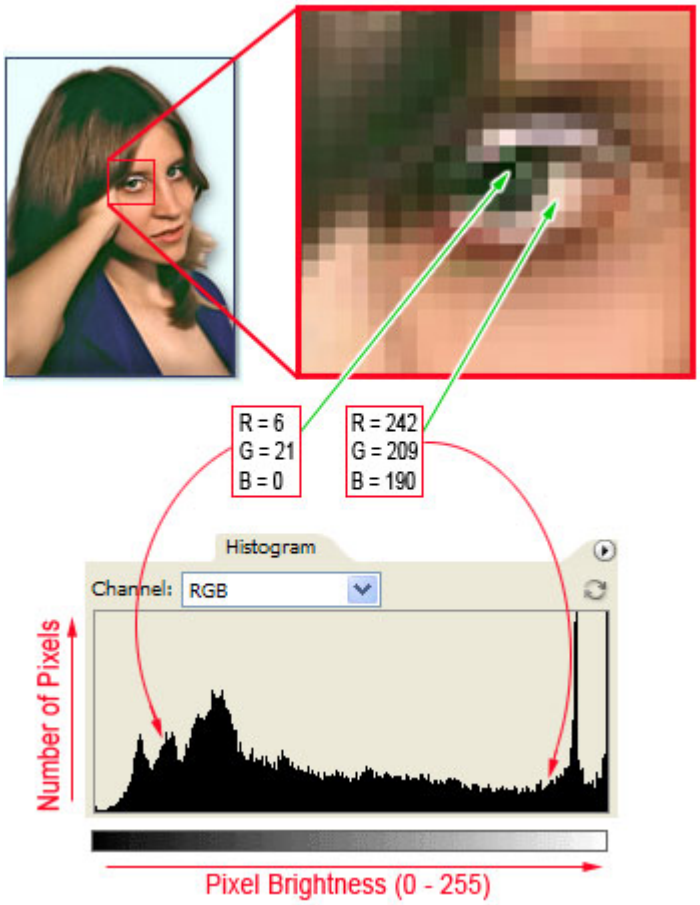


Discussions : STL Selection

1. Data arrives in a stream 30Hz
2. Processing time of data, data dependent (20-40Hz)
3. We always need to process all data (no data dropped) and oldest first

Discussions : STL Selection

1. Take each RGB pixel and convert to HSV space, select H value
2. Perform histogram H values
3. Allow user to select a bin
4. Recover all pixel locations that belong to that bin



http://www.pixartweb.com/tech-t7

Discussions : STL Selection

1. Take each pixel and compute optic flow (needs access to neighbour values)
2. Create histogram of values (of optic flow)
3. Recover all pixel locations that belong to the bin

https://www.youtube.com/watch?v=2xs0fcmgKC0

Discussions : STL Selection

- Two continuous streams of data
  - Sensor 1 (Data + Timestamp) 20Hz
  - Sensor 2 (Data + Timestamp) 100Hz
- Find the closest match for Sensor Data
- Which sensor data would you select to search through?