# Lecture 17

## Contents:

1- C++ website for more libraries.

2- Python

3- Final exam 2018

المنهج خلص .

## 1- C++ website for more libraries.

هل كل ما هعمل sorting هقعد افكر ازاى وكده .. لا اغلب الناس بتستخدم على طول الlibraries اللى على النت.عشان كمان بتكون احسن ومتضبطة.

فهنشوف امثلة للlibraries دى

Containers:

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

### Containers

**Standard Containers**

A container is a holder object that stores a collection of other objects (its elements). They are implemented as class templates, which allows a great flexibility in the types supported as elements.

The container manages the storage space for its elements and provides member functions to access them, either directly or through iterators (reference objects with similar properties to pointers).

Containers replicate structures very commonly used in programming: dynamic arrays (vector), queues (queue), stacks (stack), heaps (priority_queue), linked lists (list), trees (set), associative arrays (map)...

iterators يعنى pointer بيتحرك.

تعالى نبص على الcontainers اللى عندنا وانواعها

**Container class templates**

**Sequence containers:**

| | |
|---|---|
| **array** C++11 | Array class (class template ) |
| **vector** | Vector (class template ) |
| **deque** | Double ended queue (class template ) |
| **forward_list** C++11 | Forward list (class template ) |
| **list** | List (class template ) |

نشوف مثلا الarray

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

## std::array <array>

```
template < class T, size_t N > class array;
```

**Array class**

Arrays are fixed-size sequence containers: they hold a specific number of elements ordered in a strict linear sequence.

Internally, an array does not keep any data other than the elements it contains (not even its size, which is a template parameter, fixed on compile time). It is as efficient in terms of storage size as an ordinary array declared with the language's bracket syntax ([]). This class merely adds a layer of member and global functions to it, so that arrays can be used as standard containers.

Unlike the other standard containers, arrays have a fixed size and do not manage the allocation of its elements through an allocator: they are an aggregate type encapsulating a fixed-size array of elements. Therefore, they cannot be expanded or contracted dynamically (see vector for a similar container that can be expanded).

طيب ايه الoptions اللى مديهانى؟

### *fx* Member functions

**Iterators**

| | |
|---|---|
| **begin** | Return iterator to beginning (public member function ) |
| **end** | Return iterator to end (public member function ) |
| **rbegin** | Return reverse iterator to reverse beginning (public member function ) |
| **rend** | Return reverse iterator to reverse end (public member function ) |
| **cbegin** | Return const_iterator to beginning (public member function ) |
| **cend** | Return const_iterator to end (public member function ) |
| **crbegin** | Return const_reverse_iterator to reverse beginning (public member function ) |
| **crend** | Return const_reverse_iterator to reverse end (public member function ) |

**Capacity**

| | |
|---|---|
| **size** | Return size (public member function ) |
| **max_size** | Return maximum size (public member function ) |
| **empty** | Test whether array is empty (public member function ) |

**Element access**

| | |
|---|---|
| **operator[]** | Access element (public member function ) |
| **at** | Access element (public member function ) |
| **front** | Access first element (public member function ) |
| **back** | Access last element (public member function ) |
| **data** | Get pointer to data (public member function ) |

ندخل نشوف beginمثلا

http://www.cplusplus.com/reference/array/array/begin/

## std::array::begin

```
      iterator begin() noexcept;
const_iterator begin() const noexcept;
```

**Return iterator to beginning**

Returns an iterator pointing to the first element in the array container.

انا ممكن أقول array واحط رقم الindex وممكن الطريقة اللى فوق دى

**Example**

```cpp
// array::begin example
#include <iostream>
#include <array>

int main ()
{
  std::array<int,5> myarray = { 2, 16, 77, 34, 50 };

  std::cout << "myarray contains:";
  for ( auto it = myarray.begin(); it != myarray.end(); ++it )
    std::cout << ' ' << *it;
  std::cout << '\n';

  return 0;
}
```

Output:

```
myarray contains: 2 16 77 34 50
```

وفضل الدكتور يبص على اللينكات الجاية دى .. وقال مش الهدف اعرف تفاصيل الحاجات دى .. بس الهدف انى اعرف انى ممكن الاقى حاجات تانية احسن على النت للdatastructure

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

هنا كان مكتوب ان الsize مش fixed يبقى الvector أصلا implemented بdynamic allocation .. هل انا عارف ازاى بالضبط ..لا ودى فكرة الencapsulation وهو حتى مش قايل تفاصيل الimplementation في الlink

بس ازاى dynamic وفى نفس الوقت مكتوب انى اقدر access اى element على طول؟

بان الpointer بيشاور على اول element وانا عارف ان جوة الarray في integers مثلا فهزود على ال pointerرقم عشان اروح للelement اللى انا عايزه.

طب انهى احسن دى وللا الlist ؟

على حسب الapp لو الsize هيتغير مرة وللا اتنين او انت عايز access سريع يبقى الvector احسن مثلا.

http://www.cplusplus.com/reference/vector/vector/begin/

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

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

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

طب افرض عندى صورة فيها object والobject ده عبارة عن pixels وعايز اعبر عنه فهعمل حاجة اسمها set فيها x,y,l مثلا .. يعنى الدكتور يقصد ان ممكن يبقى عندك حاجة تحتاج تعملها datastructure مناسب جديد.

## 2- Python

http://thomas-cokelaer.info/tutorials/python/data_structures.html

الpython بقى من اكتر اللغات انتشارا عشان فيه libraries كتير جدا

هي interpreted language يعنى مش لازم تcompile الأول قبل ما تrun

فال pythonكمان عنده حاجات مش موجودة في لغات تانية زى الdictionary

http://thomas-cokelaer.info/tutorials/python/dicts.html

هنا انت عايز تربط مجموعة keys بمجموعة values ..

### 4.3.1. Quick example

A dictionary is a sequence of items. Each item is a pair made of a key and a value. Dictionaries are not sorted. You can access to the list of keys or values independently.

```
>>> d = {'first':'string value', 'second':[1,2]}
>>> d.keys()
['first', 'second']
>>> d.values()
['string value', [1, 2]]
```

You can access to the value of a given key as follows:

```
>>> d['first']
'string value'
```

بس ال pythonمشكلته انه بطئ .. فانت بتشوف انت عايز سرعة وللا انت بتعمل processing لحاجات قليلة ومش مهتم بالسرعة ؟

طب افرض السرعة بالنسبة تمام في كود الpython ماعدا في حتة واحدة كانت بطيئة اوى ... فعايز تخليها c++

هتعمل ايه؟

هتعمل compile version من c++ library للحتة اللى عايزها اسرع وinterface الكلام ده مع code الpython.

## 3- Final exam 2018

**Failure is not falling down but refusing to get up.**