**STL(standard template library)**

هي عبارة عن مجموعة من المكتبات اللي بتحتوي مجموعة من الكلاسات الجاهزة ودوال للتحكم بها

**STL has four components:**

1-Containers.

2-Algorithms.

3-Iterators.

1. **Containers(it is a data structure):**

تستخدم لتخزين مجموعة من العناصر من نفس النوع في مكان واحد

**There are four types of containers:**

1. sequance container:

Provide a data structures which can be accessed in a sequential manner.

(vector ,List ,Deque ,Arrays ,forward\_list )

1. Container adaptors:

Provide a different interface for sequential containers.

(queue ,priority\_queue, stack)

1. Associative containers:

Implemented sorted data structures that can be quickly searched.

(set ,multiset ,map ,multimap)

1. Unordered associative containers:

Implement unsorted data structure that can be quickly searched.

(unordered\_set ,unordered\_multiset ,unordered\_map ,unordered\_multimap)

1. **Algorithms:**

لإجراء العديد من العمليات علي محتواياتهاcontainersهي مجموعة من الدوال اللتي تستخدم مع ال

**There are two types of containers:**

1-Non-manipulating algorithms:

Sort, reverse, \*max\_element, \*min\_element, accumulate, count, find, binary\_search, lower\_bound, upper\_bound

2-Some mnipulating algorithms:

arr.erase, next\_permutation, prev\_permutation, distance

1. **Iterators:**

) مثل تسلسل من الأرقام أو الحروفcontainersتستخدم للإشارة إلي عناوين في الذاكرة خاصة بالحاويات(

للتقليل من التعقيد ووقت تنفيذ البرنامج.

begin():is used to return the beginning position of the container.

end():is used to return the after end position of the container.

advance(itr, p):is used to increment the iterator position till the specified number mentioned in its arguments.

next():returns the new iterator that the iterator would point after advancing the positions mentioned in its

arguments.

prev():returns the new iterator that the iterator would point after decrementing the positions mentioned in its arguments.

Inserter():is used to insert the elements at any position in the container. It accepts 2 arguments, the container and iterator to position where the elements have to be inserted.

Vectors

vectorوهي عبارة عن نوع من المصفوفات لتخزين عدة قيم والتعامل معها بعنوان ال

|  |  |  |
| --- | --- | --- |
| #include <vector> // vectorالمكتبة الخاصة بتعريف ال | | |
| تعريف  vectorال | Method(1) | vector<int> vector3(5, 12); //where 5:is the size,12:is the value  Meaning: vector<int> vector3 = {12, 12, 12, 12, 12}; |
| Method(2) | Vector<int> name = {val\_1,val\_2,val\_3} |
| Method(3) | Vector<int> name {val\_1,val\_2,val\_3} |

**Example(1):**

#include <iostream>

#include <vector>

using namespace std;

int main(){

// initializer list method(1)

vector<int> vector1 = {1, 2, 3, 4, 5};

// initialization method(2)

vector<int> vector2{6, 7, 8, 9, 10};

// initialization method(3)

vector<int> vector3(5, 12); //which equal vector<int> vector3 = {12, 12, 12, 12, 12};

cout << "vector1 = ";

// looping through vector method\_1

for (const int& i : vector1){

cout << i << " ";

}

cout << "\nvector2 = ";

// looping through vector method\_2

for (int i : vector2){

cout << i << " ";

}

// looping through vector method\_3

cout << "\nvector3 = ";

vector<int>::iterator iter; // declare iterator

for (iter=vector3.begin();iter<vector3.end(); iter++) {

cout << \*iter << " "; }

return 0; }

**1-Add elements to a vector:**

|  |  |
| --- | --- |
| VectorName.push\_back(value) | vectorبتضيف عنصر في نهاية ال |

**Example:**

#include <iostream>

#include <vector>

using namespace std;

int main(){

cout << "Initializing a Vector: ";

vector<int> num {1, 2, 3, 4, 5};

for (const int& i : num) {

cout << i << " ";

}

// add the integers 6 and 7 to the vector

num.push\_back(6);

num.push\_back(7);

cout << "\n After updating the Vector: ";

for (const int& i : num) {

cout << i << " ";

}

return 0; }

**2-Access Elements of a Vector:**

|  |  |
| --- | --- |
| VectorName.at(index) | بتاعه indexبترجع العنصر من خلال ال |

**Example:**

#include <iostream>

#include <vector>

using namespace std;

int main(){

vector<int> num {1, 2, 3, 4, 5};

num.push\_back(200);

cout << "Element at Index 0: " << num.at(0) << endl;

cout << "Element at Index 2: " << num.at(2) << endl;

cout << "Element at Index 4: " << num.at(4);

cout << num[4]; // print the fifth element

return 0; }

**3-Change Vector Element:**

|  |  |
| --- | --- |
| VectorName.at(index)=value; | في مكان معين vectorلتعديل قيمة في ال |

**Example:**

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<int> num {1, 2, 3, 4, 5};

cout << "Initial Vector: ";

for (const int& i : num) {

cout << i << " "; }

// change elements at indexes 1 and 4

num.at(1) = 9; // بتغير قيمة العنصر الثاني إلي 9

num.at(4) = 7; // بتغير قيمة العنصر الخامس إلي 7

cout << "\nUpdated Vector: ";

for (const int& i : num) {

cout << i << " ";

}

return 0; }

1. **Change Vector Element:**

|  |  |
| --- | --- |
| VectorName.erase(VectorName.begin()+index) | من البدايةindexبتحذف العنصر اللي رقمه |
| VectorName.erase(VectorName.end()-index) | من النهاية indexبتحذف العنصر اللي رقمه |

**Example:**

#include <iostream>

#include <vector>

using namespace std;

int main(){

cout << "Initial Vector: ";

vector<int> num {1, 2, 3, 4, 5, 6, 7, 8, 9};

for (const int& i : num) {

cout << i << " "; }

num.erase(num.begin()+3); // بتحذف العنصر الرابع من البداية

num.erase(num.end()-2); // بتحذف العنصر الثاني من النهاية

cout << "\nUpdated Vector: ";

for (const int& i : num) {

cout << i << " "; }

return 0; }

1. **add vector element:**

|  |  |
| --- | --- |
| Insert(iterator+index, value) | vector في الindexبتضيف قيمة في أي مكان |

**Example(1):**

#include <bits/stdc++.h>

using namespace std;

int main(){

// initialising the vector

vector<int> vec = { 10, 20, 30, 40 };

// inserts 3 at front

auto it = vec.insert(vec.begin(), 3);

// inserts 2 at front

vec.insert(it, 2);

int i = 2;

// inserts 7 at i-th index

it = vec.insert(vec.begin() + i, 7);

cout << "The vector elements are: ";

for (auto it = vec.begin(); it != vec.end(); ++it)

cout << \*it << " ";

return 0;

}

**Example(2):**

#include <bits/stdc++.h>

using namespace std;

int main(){

// initialising the vector

vector<int> vec = { 10, 20, 30, 40 };

// inserts 3 one time at front

auto it = vec.insert(vec.begin(), 1, 3);

// inserts 4 two times at front

vec.insert(it, 2, 4);

cout << "The vector elements are: ";

for (auto it = vec.begin(); it != vec.end(); ++it)

cout << \*it << " ";

return 0;

}

1. **Some functions:**

|  |  |
| --- | --- |
| size() | vectorبترجع عدد العناصر في ال |
| capacity(( | vectorبترجع عدد العناصر في ال |
| clear() | vectorبتمسح كل العناصر في ال |
| front() | بترجع قيمة العنصر الأول |
| back() | بترجع قيمة العنصر الأخير |
| empty() | فاضي ولا لأ vectorبتختبر هل ال |
| resize(n) | فقط وتحذف الباقي n هوvectorبتخلي عدد عناصر ال |
| assign(int size,int value) | size وعدد العناصر هو valueبتحط لجميع العناصر القيمة |

**Example:assign():**

#include<iostream>

#include <bits/stdc++.h>

using namespace std;

int main(){

vector<int> v;

v.assign(7, 100);

cout << "Size of first: "

<< int(v.size()) << '\n';

cout << "Elements are\n";

for (int i = 0; i < v.size(); i++)

cout << v[i] << endl;

return 0; }

**Example:**

#include <iostream>

#include <vector>

using namespace std;

int main(){

vector<int> v;

// Initialize v with an initialization list

v.assign({ 1, 2, 3 });

cout << "The list is:" << endl;

for (auto i = v.begin(); i != v.end(); i++){

// Printing 1 2 3 as output

cout << \*i << " "; }

return 0; }

**Example:**

#include <iostream>

#include <vector>

using namespace std;

int main(){

vector<int> myvector = {1,2,3,4,5,6,7};

// print the size of the vector

cout << "the size of the vector is " << myvector.size() << endl;

// print the size of the vector

cout << "the size of the vector is " << myvector.capacity() << endl;

// print the first and ssecond element of the array

cout << "the first element of the vector is " << myvector.front() << endl;

cout << "the last element of the vector is " << myvector.back() << endl;

myvector.resize(5); // reduce the number of elements to 5

cout << "the elements after reduction is: ";

for(int i:myvector){

cout << i << " "; }

cout << "\n";

// vector becomes empty

myvector.clear();

// test if the vector is empty or not return 1:(true) or 0(false)

cout << "return 1 if the vector is empty >> " << myvector.empty() << endl;

// Printing the vector

for (auto it = myvector.begin(); it != myvector.end(); ++it)

cout << ' ' << \*it;

return 0; }