Algoritmalar ve Programlama II – BLM 102

Hafta 14: STL (Standard Template Library)

Fenerbahçe Üniversitesi

Öğretim Elemanları

Öğretim Üyesi: Dr. Vecdi Emre Levent

Ofis: 311

Email: emre.levent@fbu.edu.tr

Asistan: Arş. Gör. Uğur Özbalkan

Ofis: 311

Email: ugur.ozbalkan@fbu.edu.tr

Ders Planı

- STL Nedir?
- Algoritmalar
- Container’lar
- Iterator’lar

STL (Standard Template Library)

- STL, içerisinde bulunan hazır sınıf ve fonksiyonları ile, algoritma geliştirme sürecinde kolaylık sağlayabilen bir kütüphanedir.

- Geliştirme zamanının düşürürler.

STL (Standard Template Library)

3 temel bileşeni vardır.

- Algoritmalar
- Contanier’lar
- Iterator’lar

STL (Standard Template Library)

3 temel bileşeni vardır.

- Algoritmalar: Dizinin tersini almak veya sıralama gibi çeşitli algoritmarı barındırır.

STL (Standard Template Library)

3 temel bileşeni vardır.

- Container'lar: Diziler, vektörler, linked-list gibi veri yapılarını hazır olarak ifade etmek için container'lar sunar.

STL (Standard Template Library)

3 temel bileşeni vardır.

- Iterator'lar: Algoritmalar ile container'lar arasındaki bağı kurarlar.

STL (Standard Template Library)

Container’lar: Veri gruplarını, bir arada tutmak için farklı yaklaşımlar sunar.

Karmaşık veri yapılarını gerçeklemeyi kolaylaştırır.

Örn. linked-list, ağaçlar vb…

STL (Standard Template Library)

En sık kullanılan container'lar

- Pair
- Tuple
- Array
- Vektörler
- Queue
- Stack
- Priority Queue
- List
- Set

STL (Standard Template Library)

Pair:

İki farklı veri tipine sahip değişkenin bir arada grup olarak kullanılmasını sağlamaktadır.

STL (Standard Template Library)

```
#include <iostream>

using namespace std;

int main()
{
pair<int, int> pair1;
pair<int, string> pair2;
pair<int, int> pair3;

pair1 = make_pair(1, 2);
pair2 = make_pair(1, "deneme");
pair1 = make_pair(5, 23);

cout << pair1.first << endl;
cout << pair2.second << endl;

if (pair1 == pair3)
cout << "Pair'ler esit" << endl;
else
cout << "Pair'ler esit degil" << endl;

return 0;
}
```

Çıktı:
Pair'ler esit degil

STL (Standard Template Library)

Pair'de kullanılabilen operatörler ve fonksiyonlar:

= : Bir pair'i diğer bir pair'e atar
swap : İki pair'in değişkenlerini yer değiştirir
make_pair : Pair oluşturmak için kullanılır
== , != , > , < , <= , >= : Karşılaştırma operatörleri

STL (Standard Template Library)

Tuple:

Aynı pair’de olduğu gibi değişkenleri gruplamaktadır. Tuple 3 farklı değişkeni gruplayabilmektedir.

STL (Standard Template Library)

```
#include <iostream>
#include <tuple>
using namespace std;

int main()
{
tuple<int, int, int> tuple1;
tuple<int, string, string> tuple2;

tuple1 = make_tuple(1, 2, 3);
tuple2 = make_tuple(1, "deneme", "test");

int id;
string first_name, last_name;
tie(id, first_name, last_name) = tuple2;

cout << id << " " << first_name << " " <<
last_name;

return 0;
}
```

Çıktı:
1 deneme test

STL (Standard Template Library)

Tuple'de kullanılabilen operatörler ve fonksiyonlar:

=	: Bir tuple'i diğer bir tuple'e atar
swap	: İki tuple'in değişkenlerini yer değiştirir
make_tuple	: Tuple oluşturmak için kullanılır
== , != , > , < , <= , >=	: Karşılaştırma operatörleri
Tie	: Değişkenleri, tuple'e bağlar

STL (Standard Template Library)

Array:

Aynı türden olan obje grubudur. Bir array klasik tanımlama ile (Örn. int x [50]) tanımlanabilir. Ancak STL ile tanımlandığında, STL'in sunduğu çeşitli fonksiyonlar kullanılabilmektedir.

STL (Standard Template Library)

Array:

```
#include <array>
#include <iostream>
using namespace std;

int main()
{
array<int, 8> a = { 1,2,3,4,5,6,7,8 };
array<int, 8> b = { 8,7,6,5,4,3,2,1 };

a.swap(b);

cout << "a : ";
for (int i = 0; i < 8; i++) {
cout << a[i] << " ";
}
cout << endl;

cout << "b : ";
for (int i = 0; i < 8; i++) {
cout << b[i] << " ";
}
return 0;
}
```

Çıktı:
a : 8 7 6 5 4 3 2 1
b : 1 2 3 4 5 6 7 8

STL (Standard Template Library)

Array'de kullanılabilen operatörler ve fonksiyonlar:

at(int) : istenilen index'teki elemana erişim, sınır dışına çıktığında, exception throw edecektir.

[int] : istenilen index'teki elemana erişim

front() : ilk elemanı döndürür

back() : Son elemanı döndürür

fill(int) : Tüm diziyi verilen argüman ile doldurur

swap : İki dizinin elemanlarını değiştirir

size : Dizinin boyutunu döndürür

STL (Standard Template Library)

Vektör:

Dizinin başlangıçta boyutunun sabit olarak belirtilmesi nedeniyle, dinamik olarak boyutu arttırılamamaktadır. Vektörler, aynı diziler gibi verileri saklarlar (Bellekte ardışık olarak) ancak çalışma zamanında boyutları arttırılabilir.

STL (Standard Template Library)

```
#include <iostream>
#include <vector>

using namespace std;

int main()
{
vector<int>  v;
v.push_back(1);
v.push_back(2);
v.push_back(4);

for (int i = 0; i < v.size(); i++)
{
        cout << v[i] << " ";
}

return 0;
}
```

Çıktı:
1 2 4

STL (Standard Template Library)

İterator Örneği

```
#include <iostream>
#include <vector>

using namespace std;

int main()
{
vector<int>  v;
v.push_back(1);
v.push_back(2);
v.push_back(4);
for (vector<int>::iterator i = v.begin(); i != v.end(); i++)
{
        cout << *i << " ";
}

}
```

Çıktı:
1 2 4

STL (Standard Template Library)

Vector’de kullanılabilen operatörler ve fonksiyonlar:

- push_back() : Vektörün sonuna eleman ekler

STL (Standard Template Library)

V başlangıç:


```
vector<int>::iterator i = v.begin();

v.insert(i, 70);
```

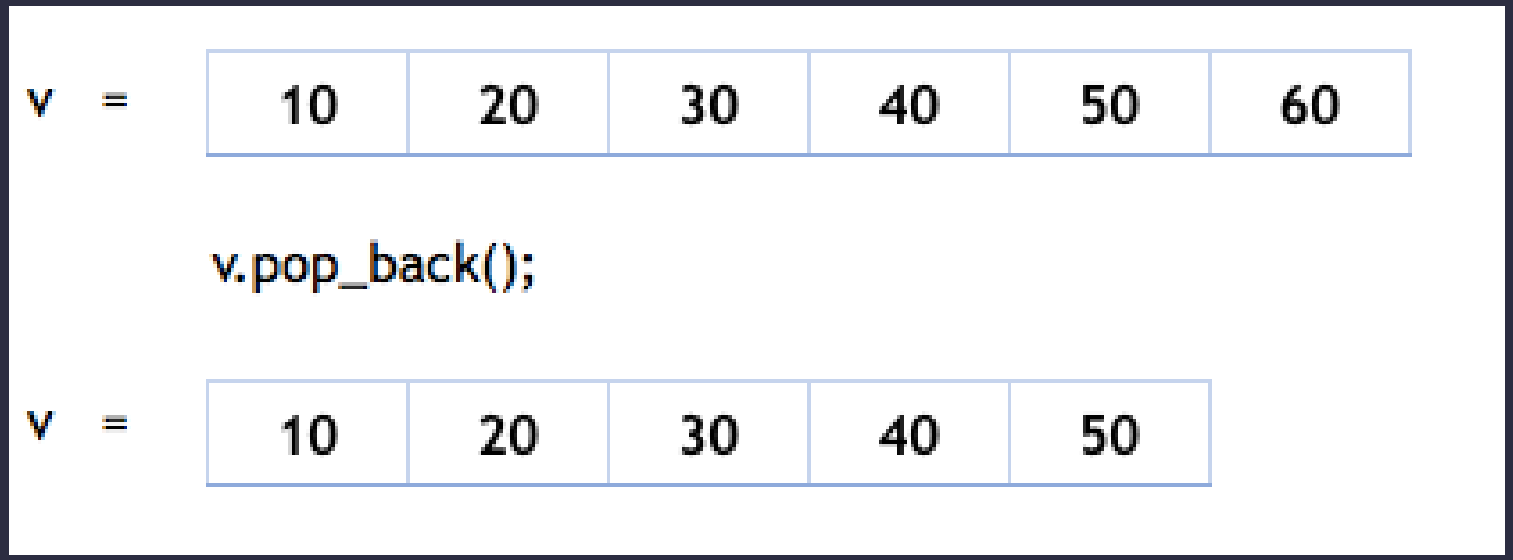
STL (Standard Template Library)

70	10	20	30	40	50	60

```
vector<int>::iterator i = v.begin();

v.insert(i, 2, 100);
```

100	100	70	10	20	30	40	50	60

STL (Standard Template Library)



STL (Standard Template Library)

```
#include <iostream>
#include <vector>

using namespace std;

int main()
{
vector<int>v{ 10,20,30,40 };
vector<int>::iterator it = v.begin();

v.erase(it);

v.erase(v.begin(), v.end() - 2);

for (it = v.begin(); it != v.end(); it++)
{
cout << *it << " ";
}
}
```

Çıktı:
30 40

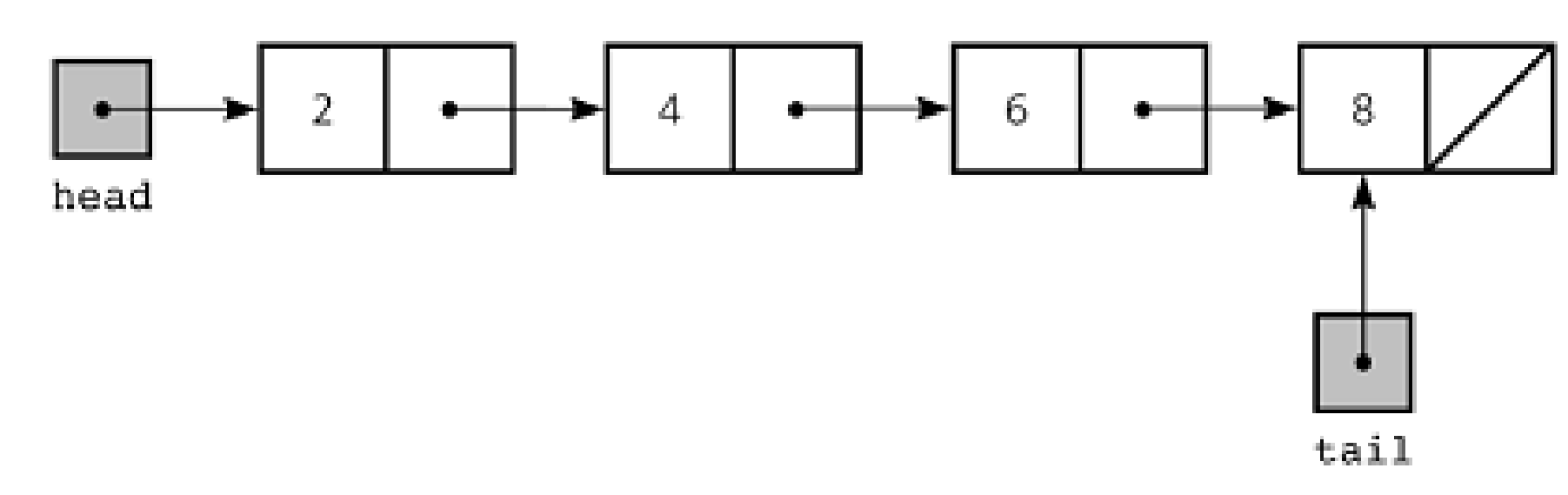
STL (Standard Template Library)

List:

Array ve vektörler elemanlarını bellekte ardışık olarak tutarlar. Dolayısıyla elemanlarının arasına yeni bir eleman eklenmek istendiğinde tüm elemanların başka bir RAM bölgesine kaydırılması gerekmektedir. Bu durum hesaplama gücü açısından maliyelidir. Listeler (linked-list) bu sorunu aşmak için kullanılmaktadır. Linked list'lerde veriler ardışık olarak tutulmaz. Bir eleman diğerinin adresini taşır.

STL (Standard Template Library)

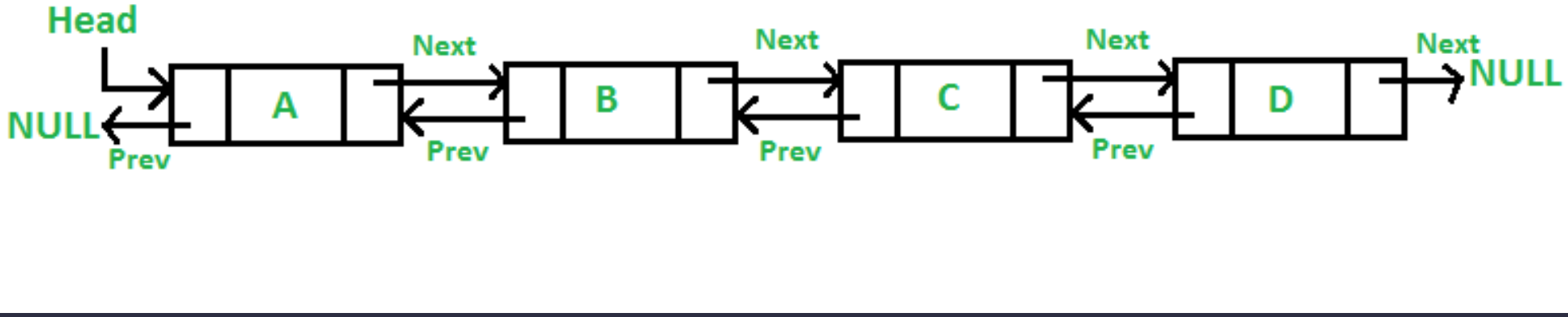
List:



Tek Yönlü Liste

STL (Standard Template Library)

List:



Çift Yönlü Liste

STL (Standard Template Library)

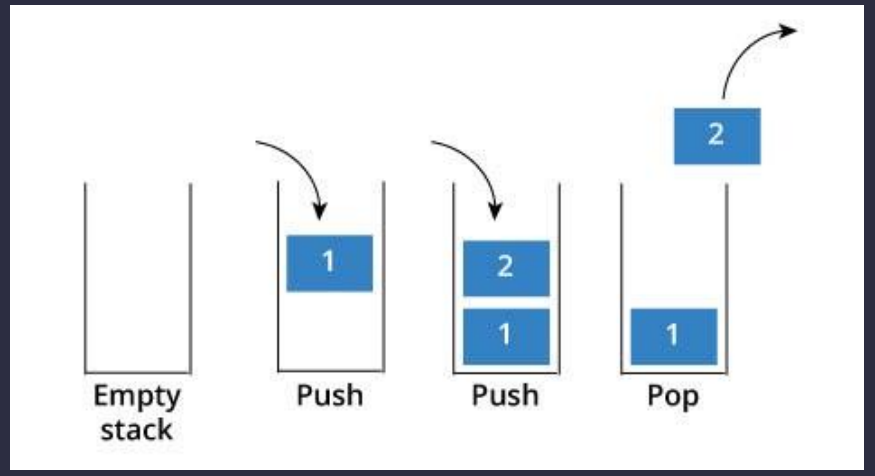
```
#include <iostream>
#include <list>

using namespace std;

int main()
{
list<int> l{ 1,2,3,4,5 };

l.push_back(6);
l.push_back(7);

for (list<int>::iterator i = l.begin(); i !=
l.end(); ++i)
cout << *i << " " ;

cout << "\n";

l.push_front(8);
l.push_front(9);

for (list<int>::iterator i = l.begin(); i !=
l.end(); ++i)
cout << *i << " ";

}
```

Çıktı:
1 2 3 4 5 6 7
9 8 1 2 3 4 5 6 7

STL (Standard Template Library)

Stack:

İçerisine veri eklendiğinde, her zaman eklenen son veri okunabilir. LIFO (Last In First Out) mantığı ile çalışır.



STL (Standard Template Library)

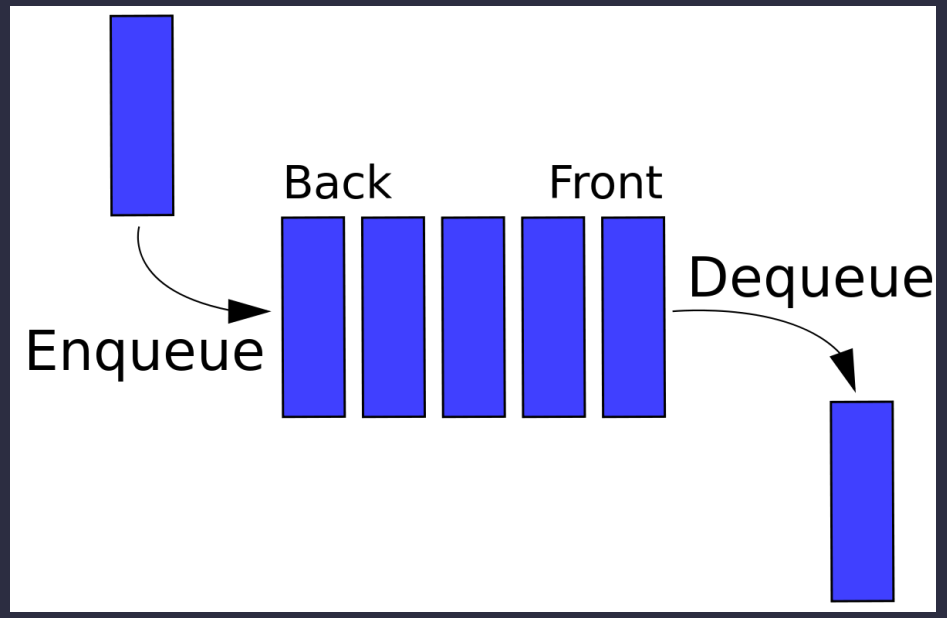
```
#include <iostream>
#include <stack>

using namespace std;

int main()
{
stack<int> s;

s.push(3);
s.push(4);
s.push(5);
cout << s.top() << endl;

cout << s.size() << endl;

s.pop();

cout << s.top();
}
```

Çıktı:
5
3
4

STL (Standard Template Library)

Queue:

FIFO (First In First Out) mantığı ile çalışır.



STL (Standard Template Library)

```
#include <iostream>
#include <queue>

using namespace std;

int main()
{
queue <int> q;

q.push(2);
q.push(3);
q.push(5);

cout << q.front() << endl;

q.pop();

cout << q.front();
}
```

Çıktı:
2
3

STL (Standard Template Library)

Priority Queue:

Queue gibidir, tek farklı en öndeki okunacak eleman daima queue'nin içerisindeki en büyük elemandır.

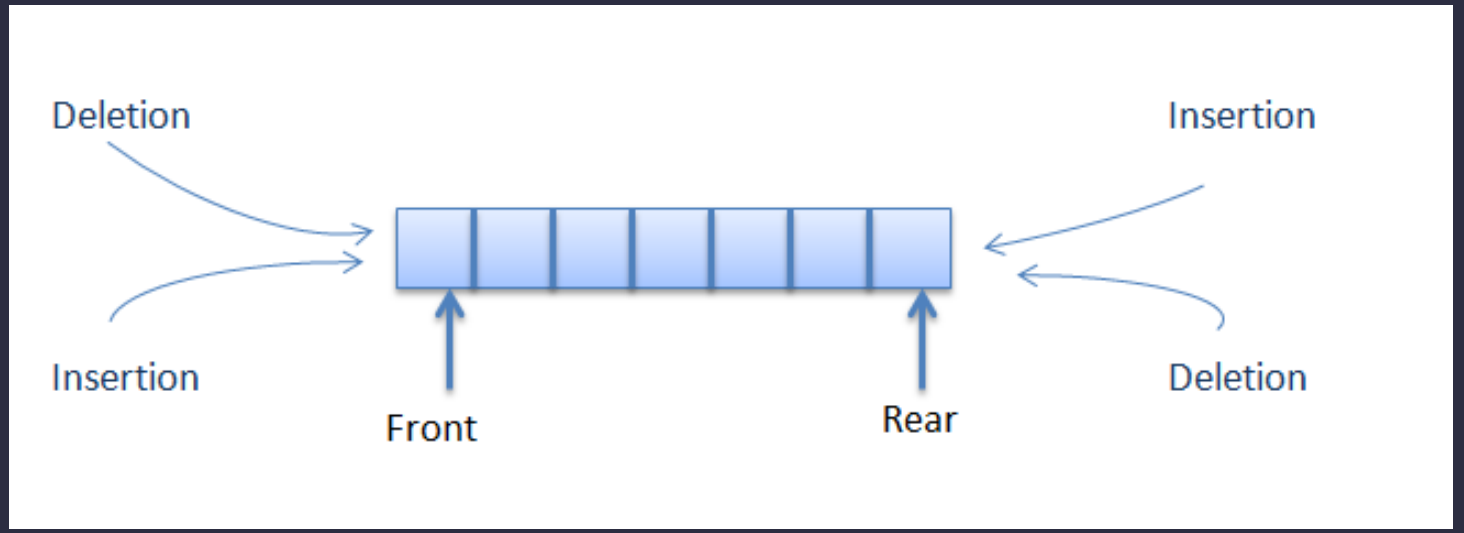
STL (Standard Template Library)

```
#include <iostream>>
#include <queue>

using namespace std;

int main()
{
priority_queue<int> pq1;

pq1.push(60);
pq1.push(30);
pq1.push(40);
pq1.push(90);

cout << pq1.top() << endl;

pq1.pop();

cout << pq1.top() << endl;

return 0;

}
```

Çıktı:
90
60

STL (Standard Template Library)

Double Ended Queue (Dequeue):
Queue'nin iki ucu bulunan versiyonudur.



STL (Standard Template Library)

```
#include <iostream>
#include <deque>
#include <vector>

using namespace std;

int main()
{

deque<int> dq = { 1,5,8,9,3 };

dq.push_back(10);
cout << dq.front() << " " << dq.back() << endl;

dq.push_front(20);

cout << dq.front() << " " << dq.back() << endl;

dq.pop_back();

cout << dq.front() << " " << dq.back() << endl;

}
```

Çıktı:
1 10
20 10
20 3

STL (Standard Template Library)

Sort:

Sıralama algoritmasıdır, konteyner'daki elemanları sıralamak için kullanılır.

STL (Standard Template Library)

```
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;

int main()
{
vector<int> v1;

v1.push_back(8);
v1.push_back(4);
v1.push_back(5);
v1.push_back(1);

cout << is_sorted(v1.begin(), v1.end()) <<
endl;

sort(v1.begin(), v1.end());

cout << is_sorted(v1.begin(), v1.end()) <<
endl;

for (vector<int>::iterator i = v1.begin(); i !=
v1.end(); i++)
{
cout << *i << " ";
}

}
```

Çıktı:

```
0
1
1 4 5 8
```

STL (Standard Template Library)

Binary search:

Arama algoritmasıdır, veri grubu içerisinde belirli bir elemanın var olup olmadığı kontrolü için kullanılabilir.

STL (Standard Template Library)

```
#include <vector>
#include<iostream>
#include <algorithm>
using namespace std;

int main()
{
vector<int> arr = { 10, 15, 20, 25, 30, 35 };

if (binary_search(arr.begin(), arr.end(), 15))
cout << "15 dizide vardir";
else
cout << "15 dizide yoktur";

cout << endl;
}
```

Çıktı:

15 dizide vardir

STL (Standard Template Library)

Min-Max:

STL’de Veri grubu içerisinde minimum ve maksimum bulma algoritması bulunmaktadır.

STL (Standard Template Library)

```
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;

int main()
{
vector<int> a = { 1, 45, 54, 71, 76, 12 };

cout << "Vector: ";
for (int i = 0; i < a.size(); i++)
cout << a[i] << " ";
cout << endl;
cout << "\nMin Element = "
<< *min_element(a.begin(), a.end());

cout << "\nMax Element = "
<< *max_element(a.begin(), a.end());

return 0;
}
```

Çıktı:
Vector: 1 45 54 71 76 12

Min Element = 1
Max Element = 76