

* Bit Dosyaları Varsayımları Sırası:

```

auto ifs = open_text_file("forster.txt");
char c;
map<char, int> cmap;

while (ifs.get(c)) {
    if (isalpha(c)) {
        ++cmap[toupper(c)];
    }
}

vector<pair<char, int>> cvec{ cmap.begin(), cmap.end() };
sort(cvec.begin(), cvec.end(),
    [](const auto& p1, const auto& p2) {
        return p1.second > p2.second;
});

for (const auto [ch, count] : cvec) {
    cout << ch << " " << count << "\n";
}

```

sayı - karakter sıralamasını map kullanı.

alfabetik karakter sayesini

sort identifier oluşturmak generalized lambda

```

E 38933
T 27295
O 23096
A 22934
I 20236
H 19827
S 19408
N 19375
R 18001
D 13329
L 13079
U 9526
C 8116
M 7686
W 7129
Y 7052
G 6252
F 5571
P 5063
B 4766
V 3084
K 2285
X 379
J 310
Q 255
Z 167

```

C:\nec\1.exe (process 24116) exited with code 0.
Press any key to close this window . . .

```

while (ifs.get(c)) {
    if (isalpha(c)) {
        ++cmap[toupper(c)];
    }
}

vector<pair<char, int>> cvec{ cmap.begin(), cmap.end() };

const auto& fpred = [](const auto& p1, const auto& p2) {
    return p1.second > p2.second;
};

sort(cvec.begin(), cvec.end(), fpred);

```

lambda ifadesi okunaklı hale getirildi.

Ortalıda from boyut da genişlendi.

* Getline Örneği:

```

int main()
{
    using namespace std;

    auto ifs = open_text_file("forster.txt");
    string sline;
    while (getline(ifs, sline)) {
        cout << sline << "\n";
        _getch();
    }
}

```

döngünün sonuna kadar, dosyadaki satırı okup, sonda yer.

```

int main()
{
    using namespace std;

    auto ifs = open_text_file("suleyman.txt");
    string sline;

    while (getline(ifs, sline, ',')) {
        cout << sline << "\n";
        // getch();
    }
}

```

getline'a 3. arguman olarak türkçe içindeki delimiter giriniz.

Buylez nesneleri virgülle kader olur. Noktalı virgüller ise devam eder.

* Dosyayı Satır Satır Çıkarınca Tüme:

```
int main()
{
    using namespace std;

    auto ifs = open_text_file("forster.txt");
    string sline;

    vector<string> linevec;

    while (getline(ifs, sline)) {
        linevec.push_back(sline); } → push_back, konusine organen direkt
        getline nesnenin value category'sine girece ya kopyalar
        be da mireler.
    std::cout << "linevec.size() = " << linevec.size() << "\n";
}
```

```
int main()
{
    using namespace std;

    auto ifs = open_text_file("forster.txt");
    string sline;

    vector<string> linevec;

    while (getline(ifs, sline)) {
        linevec.push_back(move(sline)); } → istekli kodun daha
        effcient olması için move kullanı
        - Move edilmiş nesnenin value'su birebir
        ama one yandan değer etanobilsir.

    char c;
    cout << "hangi karakterler baslayanlar: ";
    cin >> c;

    auto iter = partition(linevec.begin(), linevec.end(), [c](const string& s) {return s.front() == c; });

    for (const auto& sline : linevec) {
        std::cout << sline << '\n';
        _getch();
    }
}
```

* Dosyadaki Belirli bir Karakterin Sayısını Dondurme:

```
12
13 int main()
14 {
15     using namespace std;
16
17     auto ifs = open_text_file("forster.txt");
18     cout << "hangi karakter sayilsin: ";
19     char c;
20     cin >> c;
21
22     cout << count(istream_iterator<char>{ifs}, {}, c) << "\n";
23
24 }
```

* Giriş C++'da Exception Handling:

- Default olarak, exception handling yok! Stream nesnesi fail bitinde exception throw ederse!
- Bir exception olan fenomenin hangi ros::state'ine exception throw edegemi ögrenmeye çalışır? ✓

```
int main()
{
    using namespace std;

    cin.exceptions(ios::failbit);

    int x;

    cout << "bir sayı giriniz: ";
    try {
        cin >> x;
    }
    catch (const std::exception& ex) {
        std::cout << "exception caught: " << ex.what() << '\n';
    }
}
```

```
Microsoft Visual Studio Debug Console
bir sayı giriniz: ali
exception caught: ios_base::failbit set: iostream stream error
C:\nec\ a.exe (process 37196) exited with code 0.
Press any key to close this window . . .
```



```
double readsum(std::istream& is)
{
    auto old_exception = is.exceptions();           "no exception state"
    is.exceptions(ios::failbit | ios::badbit);      //!!! exception state
    double dval, sum{};

    try {
        while (is >> dval)
            sum += dval;
    }
    catch (...) {
        if (!is.eof()) {
            is.exceptions(old_exception);
            throw;
        }
    }
    is.exceptions(old_exception);                  Erste例外を復元する

    return sum;
}
```

*Stream nesnesinin "failbit" eger
stream içinde bir exception olursa
setelrinin /formatunin belki de
dogrulugunu dogru dogru.*

*Eğer例外を復元する
例外!*

* Bir Dosya Yedeklemek için Kullanılır: → rdbuf()

```
int main()
{
    auto ifs = open_text_file("nutility.cpp");

    cout << ifs.rdbuf() << "\n";
}
```

1.) rdbuf(), bir FILE* dandır.

2.) cout.cc overloadedinden biriostream* olur.

→ Buyle böyle direkt rdbuf() ile cout'a ulaşılır.

```
using namespace std;

int main()
{
    auto ifs = open_text_file("forster.txt"); 3

    ofstream ofs{ "ali.txt" }; ← Ayni yontemle forster.txt,
    ofs << ifs.rdbuf(); ali.txt'ye kopyalandı.
}
```

* Tie Fonksiyonu:

→ Eger bir input nesnesi, output nesnesine tie edilirse, input buffer'a giren okuma, output buffer'ın sonuna okun flushlar. (cin → cout).

```
int main()
{
    cout << "&cout = " << &cout << '\n';
    std::cout << "cin.tie() = " << cin.tie() << "\n";
}
```

→ cout nesnesinin adresi ile, cin.tie()'in return ettiği adres aynı!

* tie fonksiyonu arguman olarak sadece ostream* olur. → in → out'a bağlanır / out → in'e BAĞLANAMAZ!

→ Diğer buffer flush yöntemleri:

1. buffer dolduğunda
2. normal terminate
3. unitbuf modunda olabilir
4. flush

* Random Kütüphanesi:

• C++'de, random bit üreten model ile / tarihi bilgisi uniform distribute eden model oyn

• Pseudo Number Generation: • Deterministik

- Hangi input'u verse, o seed value'a göre output olur
benzeri belli.

• True Random Number Gen: • I/O rotation yok

- maaşlıktır yoktur.

Predefined random number generators

Several specific popular algorithms are predefined.

Defined in header <random>

Type	Definition
minstd_rand0(C++11)	<pre>std::linear_congruential_engine<std::uint_fast32_t, 16807, 0, 2147483647></pre> Discovered in 1969 by Lewis, Goodman and Miller, adopted as "Minimal standard" in 1988 by Park and Miller
minstd_rand(C++11)	<pre>std::linear_congruential_engine<std::uint_fast32_t, 48271, 0, 2147483647></pre> Newer "Minimum standard", recommended by Park, Miller, and Stockmeyer in 1993
mt19937(C++11)	<pre>std::mersenne_twister_engine<std::uint_fast32_t, 32, 624, 397, 31, 0x9908b0df, 11, 0xffffffff, 7, 0xd2c5680, 15, 0xfc60000, 18, 1812433253></pre> 32-bit Mersenne Twister by Matsumoto and Nishimura, 1998
mt19937_64(C++11)	<pre>std::mersenne_twister_engine<std::uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9, 29, 0x5555555555555555, 17, 0x71d67ffeda60000, 37, 0xffff7eee00000000, 43, 6364136223846793005></pre> 64-bit Mersenne Twister by Matsumoto and Nishimura, 2000
ranlux24_base(C++11)	<pre>std::subtract_with_carry_engine<std::uint_fast32_t, 24, 10, 24></pre>
ranlux48_base(C++11)	<pre>std::subtract_with_carry_engine<std::uint_fast64_t, 48, 5, 12></pre>
ranlux24(C++11)	<pre>std::discard_block_engine<std::ranlux24_base, 223, 23></pre> 24-bit RANLUX generator by Martin Lüscher and Fred James, 1994
ranlux48(C++11)	<pre>std::discard_block_engine<std::ranlux48_base, 389, 11></pre> 48-bit RANLUX generator by Martin Lüscher and Fred James, 1994
knuth_b(C++11)	<pre>std::shuffle_order_engine<std::minstd_rand0, 256></pre>
default_random_engine(C++11)	implementation-defined

• mt19937 5000 byte lik bir blok. Construction maliyeti / belirtir yoks ve yes. Fakat sys adresiyorum

```
int main()
{
    using namespace std;

    mt19937 eng;

    cout << mt19937::default_seed << "\n";
    cout << mt19937_64::default_seed << "\n";
```

seed degeri oyn

```
int main()
{
    using namespace std;

    mt19937 eng1; → default seed
    mt19937 eng2{ 87435u}; → seed as constructor argument
```

```
int main()
{
    using namespace std;

    mt19937 eng;

    eng.seed(78435u);
```

seed deklarasyon ile de seed degeri
risip, farklı değerler elde ettiğimiz

```
using namespace std;  
  
mt19937 eng;  
  
cout << mt19937::min() << "\n"; → generate edilenin min deger  
cout << mt19937::max() << "\n"; | → generate edilenin max deger
```

```
int main()  
{  
    using namespace std;  
  
    mt19937 eng;  
  
    auto ofs = create_text_file("random.txt");  
    for (int i = 0; i < 10'000; ++i) {  
        ofs << eng() << '\n';  
    }  
}  
  
rng nesnesinin fonksiyonu olası  
Operatörün ne random sayı generate ettiğini.  
}
```

1000 adet rastgele değer
oçretti.
- Eger bir kırılma olsalırsa, yine aynı değerler
elde edilir.

* Chrono ile bitişte kalanma:

```
#include <iostream>  
#include <fstream>  
#include "nutility.h"  
#include <bitset>  
#include <conio.h>  
#include <sstream>  
#include <iostream>  
#include <chrono>  
  
int main()  
{  
    using namespace std;  
  
    mt19937 eng(chrono::high_resolution_clock::now().time_since_epoch().count());  
  
    for (int i = 0; i < 10; ++i) {  
        cout << eng() << '\n';  
    }  
}
```

program her çalıştığında
farklı değer üretir

*True Random Number Generator:

```
#include <iostream>
#include <fstream>
#include "nutility.h"
#include <bitset>
#include <conio.h>
#include <sstream>
#include <iostream>
#include <chrono>

int main()
{
    using namespace std;

    random_device eng;

    for (int i = 0; i < 10; ++i) {
        cout << eng() << '\n';
    }
}
```

random_device
actında.
oluruz.

Ürettiği değerler deterministik değil

mt19937 nesnesinin constructor'un
parametresi random_device{}() gecersel
her seferde farklı random.seed ile çalışırız.

```
int main()
{
    using namespace std;

    mt19937 eng{ random_device{}() };

    for (int i = 0; i < 10; ++i) {
        cout << eng() << '\n';
    }
}
```

*Random Device Entropy:

```
int main()
{
    using namespace std;

    cout << random_device{}.entropy() << '\n';
}
```

entropy değer ne kadar fazla ise gerçek rastgelelik
beklentisi daha fazladır.