2023 12 13

std::istream_iterator

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

    istream_iterator<int> iter{cin}; // streamdeki öğeleri alıyoruz
    int val{};

    val = *iter;

    cout << "val = " << val << "\n";
    ++iter;
    val = *iter;
    ++iter;
    cout << "val =  " << val "\n";
}
```

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

    istringstream iss{"burak kose damla kubat furkan mert melike kaptan"};
         // begin                        end
    copy(istream_iterator<string>{iss}, istream_iterator<string>{},
        ostream_iterator<string>{cout, "\n"});


    istringstream iss1{"98 123 43 523 123 54 64 12 56 64 23"};
    auto sum = accumulate(istream_iterator<int>{iss1}, {}, 0);
    auto max = *max_element(istream_iterator<int>{iss1}, {}, 0);

}
```

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

    ostream os_hex{ cout.rdbuf() };
    ostream os_oct{ cout.rdbuf() };

    os_hex << hex << uppercase << showbase;
    os_oct << oct << showbase;

    Irand myrand{34523, 123412};

    for (int = 0; i < 10; ++i)
    {
        auto val = myrand();
        cout << val << "\n"; // decimal
        os_hex << val << "\n"; // hex
        os_oct << val << "\n"; // octal
    }
}
```

Dosya İşlemleri

```
/*
    std::ofstream   yazma işlemleri için
    std::ifstream   okuma işlemleri için
    std::fstream    yazma ve okuma işlemleri için

    #include <fstream>
*/
/*
    ifstream ctor ve open fonksiyonu parametre olarak:
        dosyanın ismi
        açış modulus
            ios::in okuma
            ios::out yazma
            ios::app    append / sona yazma
            ios::trunc  truncate
            ios::ate    at end
            ios::binary
*/
```

```
#include <fstream>
int main()
{
    using namespace std;

    ofstream ofs{"emre.txt", ios::trunc | ios::out}; // default ofstream
    ifstream ifs{"emre.txt", ios::in};
}
```

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

    ofstream ofs{"kutay.txt"};

    if (!ofs) // !ofs.good() ya da !ofs.fail()
    {
        cerr << "cannot open file\n";
        return 1;
    }

    std::cout << "file opened succesfully";
}
```

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

    ifstream ifs;
    boolalpha(cout);
    // is_open streamin state kontrol etmez stream nesnesiyle ilişkilenmiş açık
bir dosya var mı kontrol eder
    cout << "ifs.is_open() = " << ifs.is_open() << "\n"; // false
    ifs.open("main.cpp");
    cout << "ifs.is_open() = " << ifs.is_open() << "\n"; // true
    ifs.close();
    cout << "ifs.is_open() = " << ifs.is_open() << "\n"; // false

}
```

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

    ofstream ofs{ "kutay.txt" };

    if (!ofs)
    {
        cerr << "dosya olusturulamadi\n";
        return 1;
    }

    for (int i = 0; i < 100; ++i)
    {
        ofs << i << " " ;
    }

    // ofstream dtor dosyayı kapatır.
    // eğer stream good state ise dosyayı kapatıp başka bir dosyada işlem
yapabiliriz
}
```

```cpp
// dosyaya yazma işlemleri
#include <format> // cpp 20
int main()
{
    using namespace std;
    ofstream ofs { "emre.txt" };
    if (ofs.fail())
    {
        cerr << "dosya olusturulamadi\n";
    }
    Irand myrand {0, 300'000};
    ofs << left;
    for (int i = 0; i < 10'000; ++i)
    {
        ofs << format("{:<12} {:<16} {:<20} {}\n", myrand(), rname(), rfname(),
rtown());
    }
}
```

```cpp
// bir vektörü dosyaya yazma yöntemleri:
int main()
{
    using namespace std;

    vector<string> svec;
    rfill(svec, 10'000, rname);

    ofstream ofs { "emre.txt" };
    if (!ofs)
    {
        cerr << "dosya olusturulamadi\n";
    }
    // 1. yöntem
    for (const auto& name : svec)
        ofs << name << "\n";
    // 2. yöntem
    copy(svec.begin(), svec.end(), ostream_iterator<string>(ofs, "\n"));

    copy_if(svec.begin(), svec.end(), ostream_iterator<string>(ofs, "\n"),[](cont
string& s)
    {
        return s.length() == 5 && s.front() = 'a';
    });

    generate_n(ostream_iterator<string>{ofs , "\n"}, 1000, []
    {
        return rname() + ' '+ rfname();
    });
}
```

```cpp
// okuma işlemleri
int main()
{
    using namespace std;
    ifstream ifs{"primes.txt"};

    if (!ifs)
        cerr << "dosya acilamadi\n";
    int v;
    // (ifs >> word).operator bool
    while(ifs >> v) // akım fail olursa false döner ve while sonlanır
    {
        cout << v << " ";
    }
}
```

```cpp
void func(std::ifstream);
int main()
{
    ifstream ifs {"emre.txt"};

    func(ifs); // error copy ctor yok
    func(std::move(ifs)); // hata yok
}
```

```
/*
    ostream'deki sınıfları incompling type olarak kullancaksak
    #include<iosfwd> dahil etmek yeterli header'a iosfwd daha light bir include

*/
```

```
std::ifstream open_text_file(const std::string& filename)
{
    std::ifstream ifs {filename};
    if (!ifs)
    {
        throw std::runtime_error{ filename + " dosyayisi acilamiyor\n"};
    }
    return ifs;
}

std::ofstream create_text_file(const std::string& filename)
{
    std::ofstream ofs {filename};

    if (!ofs)
        throw std::runtime_error{filename + "dosyayisi olusturulamadi\n"};

    return ofs;
}

int main()
{
    try
    {
        auto ofs = create_text_file("emre.txt");
    }
    catch(const std::exception &ex) {
        std::cout << "exception caught: " << ex.what() << "\n";
    }

    try
    {
        auto ifs = open_text_file("emre.txt");
    }
    catch(const std::exception &ex) {
        std::cout << "exception caught: " << ex.what() << "\n";
    }
}
```

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

    int ival = 2345;
    string name{"emrebahtiyar"};

    auto filename = (ostringstream{} << name << "_"  << ival << ".txt").str();

    ofstream ofs{"out.txt"} << "bugun hava cok soguk";
}
```

```cpp
// byte byte dosya okuma
int main()
{
    ifstream ifs {"main.cpp"};

    if (!ifs)
        cerr << "dosya acilamadi\n";

    int c;
    while ((c = ifs.get()) != EOF)
    {
        cout << static_cast<char>(c); // cout.put((char)c);
    }
}
```

```cpp
// dosyayı bir defa da okuma
int main()
{
    auto ifs = open_text_file("emre.txt");
    cout << ifs.rdbuf();
}
```

```cpp
// getline
int main()
{
    auto ifs = open_text_file("emre.txt");
    string sline;
    getline(ifs , sline); // ifs döndürür

    cout << "[" << sline << "]";

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

```cpp
// dosyada veri okuyup, vector'e atıp, sıralama
int main()
{
    vector<string> linevec;
    linevec.reserve(10'000);

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

    while (getline(ifs, sline))
    {
        linevec.push_back(move(sline));
    }

    sort(linevec.begin(), linevec.end());
    copy(linevec.begin(), linevec.end(), ostream_iterator<string>{cout, "\n"});
}
```

```cpp
int main()
{
    auto ifs = open_text_file("kutay.txt");
    // 1
    vector<int> ivec({istream_iterator<int>{ifs}, {}};

    // 2
    vector<int> ivec1;
    ivec.reserve(100'000);
    ivec.assign(istream_iterator<int>{ifs}, {});
}
```

```cpp
// formatsız dosyaya yazma
int main()
{
    using namespace std;

    ofstream ofs{"primes.dat", ios::binary};

    int prime_count =0 ;
    int x = 2;

    while (prime_count < 1'000'000)
    {
        if (isprime(x))
        {
            ofs.write(reinterpret_cast<char*>(&x), sizeof(int)));
            ++prime_count;
        }
        ++x;
    }
}
```