2023 12 11

formatlama

```
/*
formatlama nitelikleri 2 ye ayrılır:
    - on-off bayrakları
    - alan  bayrakları

on-off
    .setf(ios::boolalpha)
    .unsetf(ios::boolalpha)

    ios::showbase
    ios::uppercase
    ios::skipws
    ios:showpoint

alan bayrakları
    .setf(ios::hex, ios::basefield);
    .setf(ios::left, ios::adjustfield);
    .setf(ios::fixed, ios::floatfield);

*/
```

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

    std::cout << "bir tam sayi girin: ";
    int x{};

    cin.setf(ios::hex,  ios::basefield); // ab girdik
    cin >> x; // 171 çıktı

    cout << "x = " << x << "\n";

    int k{};
    int l{};
    int m{};

    cin >> hex >> k >> oct >>  l >> dec >> m;
}
```

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

    cin.unsetf(ios::dec);
    // hiçbir if'e girmez
    if (cin.flags() & ios::hex) { std::cout  "hex set\n";}
    if (cin.flags() & ios::dec) { std::cout  "dec set\n";}
    if (cin.flags() & ios::oct) { std::cout  "oct set\n";}

    int x{};
    // kendi karar verir hangisi olacağına
    cout << "bir sayi girin: ";
    cin >> x;
    cout << "x = " << x << "\n";
}
```

```cpp
// setbase
#include <iomanip> // parametreli manipülatör için bu lib kullanılır
int main()
{
    using namespace std;
    int x;

    cin >> setbase(0); >> x;
    cin >> hex >> x;
}
```

```cpp
int main()
{
    /* std::unitbuf() kullanmadık. Eğer fonksiyona gönderilen arguman bir
    namespace ilişkin türden ise fonksiyon o namespace içinde aranır. Buna
    ADL (argument dependent lookup)
    */
    unitbuf(std::cout);
    endl(std::cout);
    flush(std::cout);
}
```

```cpp
// C memory yazma
#include <cstdio>
int main()
{
    int x = 5;
    double dval = 4.5;

    char name[] = "melike";
    char buffer[256];

    sprintf(buffer, "x = %d dval = %f name = %s",x ,dval, name);

    int k, l, m;

    char buf[] = "213 456 790";
    sscanf(buf, "%d%d%d", &k, &l, &m);

    printf("k = %d  l = %d  m = %d);
}
```

```cpp
// C++ belleğe formatlı yazma
#include <sstream>
int main()
{
    // basic_ostringstream<char> = ostringstream<char> aynı anlamda türeş işim

    ostringstream os;
    int amount = 200'000;

    os << "benim " << amount << " dolarim var\n";

    auto s = os.str();
    reverse(s.begin(), s.end());
    cout << s  << "\n";

    os << " emre";
    auto s = os.str() // üzerine yazar
}
```

```cpp
int main()
{
    int day, mon, year;

    cout << "tarihi giriniz: ";
    cin >> day >> mon >> year;

    ostringstream oss;

    oss << setfill('0') << setw(2) << day << '_' << setw(2) << mon << '_' << year
<< ".txt";
    cout << "[" << oss.str() << "]"; //04_08_1987.txt
}
```

```cpp
// mülakat sorusu
```

```
class date
{
    public:
        date(int day, int mon, int year) : day_(day), mon_(mon), year_(year){}
        friend std::ostream& operator<<(std::ostream& os, const date& dt)
        {
            //40 karakterlik alana sadece 11 yazar  çıktı 11                -12-
2023mustafa
            //return os << dt.day_ << "-" << dt.mon_ << "-" << dt.year_;
            os << dt.day_ << "-" << dt.mon_ << "-" << dt.year_;
            // 11-12-2023mustafa yazar
            return os.str();

        }
    private:
        int day_, mon_, uerar_;
}

int main()
{
    date mydate{ 11, 12, 2023};
    cout << left << setw(40) << mydate << "mustafa\n";
}
```

```
// ostringstream formatlama
int main()
{
    using namespace std;

    ostringstream oss;

    oss << hex << uppercase << showbase;
    oss << 47802 << " " << 54807;
    cout << oss.str() << "\n"; // 0XBABA 0XD617
}
```

```
//istreamstream
using namespace std;
int main()
{
    cout << "toplanacak sayilari girin: ";

    string str;
    getline(cin, str);

    cout << "[" << str << "]";
    istreamstream iss {str};
    int ival;
    int sum{};
    while(iss >> ival){
        sum += ival;
        cout << ival << "\n";
    }
    cout << sum;
}
```

```cpp
int main()
{
    cout << "sayilar giriniz: ";
    string str;
    getline(cin, str);

    istreamstream iss{ str };
    int ival;

    vector<int> ivec;

    while(iss >> ival)
    {
        ivec.push_back(ival);
    }

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

    for (const auto val : ivec)
    {
        cout << val << "\n";
    }
}
```

```cpp
int main()
{
    cout << "bir cumle girin : ";
    string str;

    getline(cin, str);
    istringstream iss(str);

    string word;
    vector<string> svec;

    while (iss >> word)
    {
        svec.push_back(move(word));
    }

    mt19937 eng;

    for (int i = 0 ; i < 20; ++i)
    {
        shuffle(svec.begin(), sved.end(), eng);
        copy(svec.begin(), svec.end(), ostream_iterator<string>(cout, " "));
    }
}
```

```cpp
// condition state   --iostate
```

```
/*
    .good() akım(stream) sağlıklı durumdaysa true
    .eof() akım hata durumunda ve hatanın nedeni stream karakter olmadıysa true
döner
    .fail() akımda hata var ama kullanabiliriz
    .bad() // akımda hata var ve akımı kullanamayız
*/
int main()
{
    cout << boolalpha;
    cout << "bir tam sayi giriniz: ";
    int x{};

    cin >> x; // string verirsek fail ve bad true dönür int verirse good true
döner
    cout << "cin.good() : " << cin.good() << "\n"; // true
    cout << "cin.eof() : " << cin.eof() << "\n"; // false
    cout << "cin.fail() : " << cin.fail() << "\n"; // false
    cout << "cin.bad() : " << cin.bad() << "\n"; // false

}
```

```
// iostate
/*
    ios::goodbit
    ios::eofbit
    ios::failbit
    ios::badbit
*/
int main()
{
    cout << ios::goodbit; // 0 yazar
    cout << ios::eosbit; // 1 yazar
    cout << ios::failbit; // 2 yazar
    cout << ios::badbit; // 4 yazar
}
```

```cpp
void print_state(const ios& s)
{
    const auto stream_state = s.rdstate();
    if (stream_state == 0)
    {
        cout << "stream is ok not bit set\n";
    }

    if (stream_state & ios::failbit)
    {
        cout << "failbit set\n";
    }
    if (stream_state & ios::badbit)
    {
        cout << "badbit set\n";
    }

    if (stream_state & ios::eosbit)
    {
        cout << "eosbit set\n";
    }
}

int main()
{
    int x ;
    cin > x;
    print_state(cin); // 12 ok ,  ali --failbad set, ctrl z eosbit set
    // eğer state hata durumdaysa yazmaya devam etmez
    cin.clear();
    print_state(cin); // hata flaglari sonlandırır ve good hale getirir

    if (cin) // cin.good()
    {
        std::cout << "ok\n";
    }
    if (!cin)   // cin.fail()
    {
        std:cout << "not ok\n";
    }
}
```

```cpp
int main()
{
    cout << "bir tam sayi girin : ";

    int x{};

    while (!(cin >> x))
    {
        if (cin.eof())
        {
            std::cout << "ama hic giris yapmadiniz ki\n";
            cin.clear();
            cout << "tekrar deneyin\n";
        }
        else
        {
            cin.clear();
            string sline;

            getline(cin, sline);
            cout << qouted(sline) << "gecerli bir sayi degil... tekrar deneyin\n";
        }
    }
}
```

```cpp
// cin.exceptions()
int main()
{
    auto cstate = cin.exceptions(); // good state döner ios::good

    cin.exceptions(ios::eofbit); // eofbit exception throw eder
    cin.exceptions(ios::failbit | ios::badbit); // failbit ya da badbit exception
throw eder
}

// istream_iterator
using namespace std;
int main()
{
    cout << "sayilar giriniz: ";

    // toplama yapacak girilen değerlerin
    cout << accumulate(istream_iterator<int>{cin}, istream_iterator<int>{}, 0)
<<"\n";
    cout << max_element(istream_iterator<int>{cin}, istream_iterator<int>{}) <<
"\n";

    string str{"2.3 5.6 6.7 4.5 1.2 4.4 3.9"}
    istringstream iss(str);

    vector<double> dvec{istream_iterator<double>{iss},
istream_iterator<double>{}};

}
```