2023 12 29

std::any

```cpp
// std::any
int main()
{
    using namespace std;
    any a(45);
    cout << a.type().name() << '\n';

    if (a.type() == typeid(int))
    {
        auto ai = any_cast<int>(a); // bad_any_cast throw eder eğer int değilse
    }

    int &r = any_cast<int&>(a);
    ++r;
    cout << any_cast<int>(a); // 46
}
```

```cpp
// reset()
int main()
{
    using namespace std;
    any a = "emrecan suster";
    cout << (a.has_value() ? "dolu" : "bos") << "\n"; // dolu
    a.reset();
    cout << (a.has_value() ? "dolu" : "bos") << "\n"; // bos
}
```

```cpp
void *operator new(std::size_t n)
{
    std::cout << "operator new called... n = " << n << "\n";
    return std::malloc(n);
}
void operator delete(void* vp)
{
    if (vp)
        std::free(vp);
}
struct Nec {
    unsigned char buffer[1024]{};
};
int main()
{
    using namespace std;
    // size derleyiciden derleyiciye değişir
    std::cout << "sizeof(any) = " << sizeof(any) << "\n"; // 40
    std::cout << "sizeof(type_info) = " << sizeof(type_info) << "\n"; // 12
    any ax = 12;
    ax = 4.5;
    ax = bitset<128>{};
    ax = Nec{}; // operator new çağrılır
}
```

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

    int arr[100]{};

    any a = arr;

    std::cout << "sizeof(arr) = " << sizeof(arr) << "\n"; // 400

    if ( a.type() == typeid(int*))
        std::cout << "pointer tutuyor\n"; // pointer tutuyor çıkar
    else if(a.type() == typeid(int[100]))
        std::cout << "dizi tutuyor\n";
}
```

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

    any a;
    a = 4567;
    a.emplace<Date>(3, 5, 1987);
    make_any<Date>(2, 5, 1987);
}
```

```cpp
// Kullanım Senaryosu
using tvp = std::pair<std::string, std::any>
int main()
{
    using namespace std;
    vector<tvp> vec;

    vec.emplace_back("name", "tamer dundar"s);
    vec.emplace_back("birth_year", 1994);
    vec.emplace_back("month", 11);
    vec.emplace_back("price", 59.97);
    vec.emplace_back("town", "mugla"s);
    vec.emplace_back("gender", "male"s);

    for (const auto&[property, value] : vec)
    {
        if (value.type() == typeid(int))
        {
            cout << property << " " << any_cast<int>(value) << "\n";
        }
        else if (value.type() == typeid(string))
        {
            cout << property << " " << any_cast<string>(value) << "\n";
        }
        else if (value.type() == typeid(double))
        {
            cout << property << " " << any_cast<double>(value) << "\n";
        }
    }
}
```

```cpp
// in_place_type
int main()
{
    using namespace std;
    any ax{in_place_type<Date>, 1, 1, 2024};
}
```

Std::random

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

    default_random_engine // std::mt19937
    cout << (typeid(mt19937) == typeid(default_random_engine)) << "\n";

    mt19937_64 // 64 bitlik üretir

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

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

    mt19937 eng{123};
    for (int i = 0; i < 10000 - 1; ++i)
        (void)eng(); // bütün derleyicilerden aynı olmak zorunda

    cout << "min = " << mt19937::min() << "\n";
    cout << "max = " << mt19937::max() << "\n";
}
```

```cpp
int main()
{
    using namespace std;
    // kopyalama yapmak çok maliyetli
    std::cout << "sizeof(mt19937) = " << sizeof(mt19937) << "\n"; // 5000
}
```

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

    mt19937 eng();
    //// eng böyle kullanırsak kopyalama olur
    generate_n(vec.begin(), 10'000, eng);
    // böyle kullanmnımlaı
    generate_n(vec.begin(), 10'000, ref(eng));

}
```

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

    mt19937 e1(32);
    mt19937 e2(32);
    mt19937 e3(56);

    cout << (e1 == e2) << "\n"; // true
    cout << (e1 == e3) << "\n"; // false

    // rastgele sayı üretince state değişir.
    auto val = e1();
    ++val;
    cout << (e1 == e2) << "\n"; // false
    val = e2();
    cout << (e1 == e2) << "\n"; // true
}
```

```cpp
int main()
{
    mt19937 eng{123};

    for (int i = 0; i < 10; ++i) {
        (void)eng();
    }

    stringstream ss;
    ss << eng;

    mt19937 eng2;
    ss >> eng2; // state kopyalamış olduk
}
```

```cpp
int main()
{
    using namespace std;
    // 10 - 14 arasında değerler veririr uniform olarak
    uniform_int_distribution<int> dist{ 10, 14 };

    mt19937 eng;

    for (int i = 0; i < 100; ++i){
        cout << dist(eng) << ' ';
    }
}
```

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

    map<int, int> cmap;
    uniform_int_distribution dis{0 , 20};

    for (int i = 0; i < 20'000'000; ++i)
    {
        ++cmap[dist(eng)]; // rastgele sayılar birbirine yakın sayıda üretilir
    }
}
```

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

    mt19937 eng;
    uniform_real_distribution dist{ 5.6, 6.1};

    for (int i = 0; i < 100; ++i)
        cout << dist(eng) << "\n";
}
```

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

    mt19937 eng;

    normal_distribution<double> dist(50, 5.);
}
```