std::variant

- std::varian је генерички облик уније.
- Унија омогућава да исто парче меморије посматрамо на различите начине (као да је различитог типа). Свако поље уније представља један начин гледања.
- Варијанта додатно садржи информацију ког типа је вредност тренутно у њега постављана. Другим речима, варијанта садржи информацију о типу објекта који се ту тренутно налази.
- “Збир типова”

```
union X {
    int a1;
    float a2;
};

union Y {
    double a1;
    float a2;
    long a3;
};
```

```
std::variant<
    int,
    float> x;

std::variant<
    double,
    float,
    long> y;
```

std::variant

- За приступ пољима варијанте, користи се get функција.

```
std::variant<int, float> x;

std::cout << std::get<1>(x);
std::get<1>(x) = 5.7f;
```

- Уколико x тренутно садржи алтернативу са индексом 1, биће враћена референца на вредност смештену у x. У супротном, биће бачен овај изузетак: std::bad_variant_access

```
std::variant<int, float> x;

std::cout << std::get<float>(x); // добавља & или баца изузетак
std::get<int>(x) = 5.7f; // грешка током превођења
```

std::variant

- Још корисних функција:

```
std::variant<int, float> x;

x.index(); // индекс тренутне алтернативе

std::holds_alternative<float>(x);
    // true уколико је float тренутна алтернатива


auto p = std::get_if<1>(x);
if (p)
    std::cout << *p;

auto p = std::get_if<float>(x);
if (p)
    std::cout << *p;
```

Торке - std::tuple

- std::tuple је, на неки начин, супротно од std::variant -- “Производ типова”
- Оно што је std::varian наспрам уније, то је std::tuple наспрам структуре.

```
struct X {
    int a1;
    float a2;
};

struct Y {
    double a1;
    std::string a2;
    long a3;
};

struct Z {
    long long a1;
    const double* a2;
    char* a3;
    std::vector<int> a4;
};
```

```
std::tuple<int, float> x;
// или
auto x = std::make_tuple(5, 0.5);


std::tuple<
    double,
    std::string,
    long> y;


std::tuple<
    long long,
    const double*,
    char*,
    std::vector<int>> z;
```

Торке - std::tuple

- Приступ елементима торке:

```
std::tuple<int, float, int> x;
std::get<1>(x)
std::get<0>(x)
std::get<float>(x)
std::get<int>(x) // грешка!!! који елемент типа int? има их два
```

- Постоји могућност и да се елементи торке „распакују“ у појединачне промењиве:

```
std::tuple<int, float, int> x;
int a1, a3;
float a2;
std::tie(a1, a2, a3) = x;
// а може и овако:
std::tie(std::ignore, a2, std::ignore) = x;
```

Торке - std::tuple

- Аутоматско распакивање торки

```
std::tuple<int, float, int> foo() {
    ...
    return {5, x, y};
}
auto [a1, a2, a3] = foo();
// auto [a1, a2, a3] {foo()};
// auto [a1, a2, a3] (foo());
```

Један начина коришћења торки је за враћање више повратних вредности из функције:

```
std::tuple<int, float, int> foo() {
    ...
    return {5, x, y}; // до Це++17 је морало овако да се пише:
    // return std::tuple<int, float, int>{5, x, y};
}
int a1, a3;
float a2;
std::tie(a1, a2, a3) = foo();
```