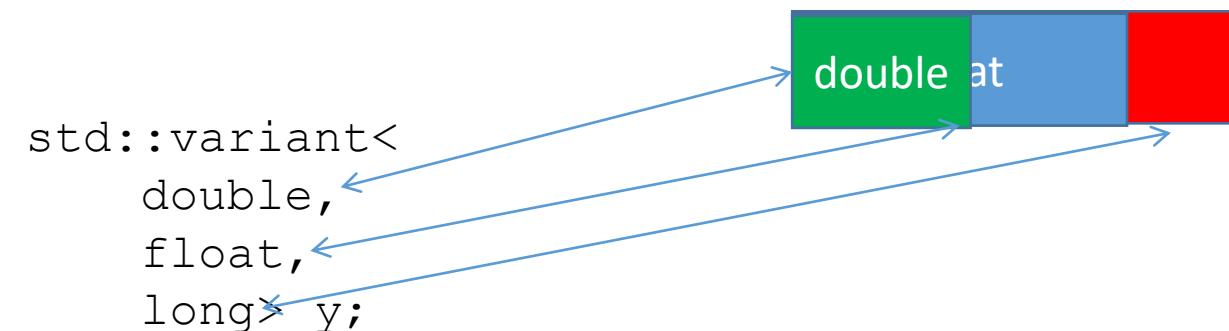


# std::variant

- 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 {                      std::variant<
    int a1;                    int,
    float a2;                  float> x;
};
```

```
union Y {                      std::variant<
    double a1;                 double,
    float a2;                  float,
    long a3;                  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::variant наспрам уније, то је std::tuple наспрам структуре.

```
struct X {                                std::tuple<int, float> x;
    int a1;                                // или
    float a2;                             auto x = std::make_tuple(5, 0.5);
};

struct Y {                                std::tuple<
    double a1;                            double,
    std::string a2;                      std::string,
    long a3;                            long> y;
};

struct Z {                                std::tuple<
    long long a1;                         long long,
    const double* a2;                     const double*,
    char* a3;                           char*,
    std::vector<int> a4;                  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 променљиве:

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