C++17引入了结构化绑定的语言特性

在C++17前假设我有一个pair,然后我想用x,y得到pair的两个值需要下面的操作

```
#include <iostream>

int main() {
    std::pair<int, int> p{1, 2};
    int x = p.first, y = p.second;
    std::cout << x << " " << y << std::endl; // 1 2
}
```

从C++17起可以用结构化绑定优化这种写法

```
   std::pair<int, int> p{1, 2};
   auto [x, y] = p;
   std::cout << x << " " << y << std::endl; // 1 2
```

结构化绑定也可以用于数组遍历

```
#include <iostream>
#include <vector>

int main() {
    std::vector<std::pair<int, int>> a{{1, 1}, {2, 2}, {3, 3}};
    for (const auto &[x, y] : a) {
        std::cout << x << " " << y << " "; // 1 1 2 2 3 3
    }
}
```

下列三种写法对结构化绑定都成立

```
    std::pair<int, int> a{};
    auto [x1, y1]{a}; // OK
    auto [x2, y2] = a; // OK
    auto [x3, y3](a); // OK
```

结构化绑定过程有一个隐藏的匿名对象,结构化绑定时引入的变量名都指向这个匿名对象的成员

```
//auto [u, v] = ms;
//等价如下:
//auto e = ms;
//alias name u = e.i;
//alias name v = e.s;// u,v是匿名对象的别名
//绑定实际上是对匿名对象的绑定,匿名对象的生命周期与结构化绑定的生命周期相同.
```