第11章 非受限联合类型

《现代C++语言核心特性解析》 谢丙堃

联合类型在C++中的局限性

- 传统联合类型的成员类型不能是一个非平凡类型

```
union U
{
    int x1;
    float x2;
    std::string x3;
};
```

使用非受限联合类型

- 需要提供联合类型的构造和析构函数

```
union U {
    U() {}      // 存在非平凡类型成员，必须提供构造函数
    ~U() {}     // 存在非平凡类型成员，必须提供析构函数
    int x1;
    float x2;
    std::string x3;
    std::vector<int> x4;
};
int main()
{
    U u;
    u.x3 = "hello world";
    std::cout << u.x3;
}
```

使用非受限联合类型

- 更进一步，实现构造和析构

```
union U
{
    U() : x3() {}
    ~U() { x3.~basic_string(); }
    int x1;
    float x2;
    std::string x3;
    std::vector<int> x4;
};
```

使用非受限联合类型

- 正确的使用方法

```
union U
{
    U() {}
    ~U() {}
    int x1;
    float x2;
    std::string x3;
    std::vector<int> x4;
};
```

```
int main()
{
    U u;
    new(&u.x3) std::string("hello world");
    std::cout << u.x3 << std::endl;
    u.x3.~basic_string();

    new(&u.x4) std::vector<int>;
    u.x4.push_back(58);
    std::cout << u.x4[0] << std::endl;
    u.x4.~vector();
}
```

感谢聆听
欢迎关注