52 How to Deal with Multiple Return Values in C++

C++中如何处理多返回值
本课将讨论什么是*tuple*（元组），什么是*pair*（键值对），如何在C++中处理多个返回类型。

1. 怎么返回不同类型?

我们有一个函数，这个函数需要返回两个字符串，或者要返回一个*integer*+一个*string*，而C++的默认情况下函数只能返回一种类型，一个特定的变量，这种情况下就会遇到麻烦。
如果有一个函数需要返回两个或多个相同类型的变量，则可以返回`vector`或者数组，不过出于一些原因这也不是最好的做法。

Cherno最喜欢的解决方法是传建一个只包含要返回内容的结构体。

C++提供了多种方法。

1. 指针和引用

```C++
#include <iostream>

void returnWithReference(std::string& str, int& num)
{
    str = "Hello";
    num = 42;
}

int main()
{
    std::string str;
    int num;
    returnWithReference(str, num);
    std::cout << str << ", " << num << std::endl;
    return 0;
}
```

2. array和vector

Array和vector的区别：array会在栈上创建，而vector会把它的底层存储在堆上，所以从技术上讲返回`std::array`会更快。

```cpp
#include <iostream>
#include <array>
#include <vector>

std::array<int, 2> returnWithArray()
{
    std::array<int, 2> result;
    result[0] = 42;
    result[1] = 56;
    return result;
}

std::vector<int> returnWithVector()
{
    std::vector<int> result;
    result.push_back(42);
    result.push_back(56);
    return result;
}

int main()
{
    std::array<int, 2> arrResult = returnWithArray();
    std::cout << arrResult[0] << ", " << arrResult[1] << std::endl;

    std::vector<int> vecResult = returnWithVector();
    std::cout << vecResult[0] << ", " << vecResult[1] << std::endl;

    return 0;
}
```

但显然这两种方法只有在类型相同的情况下才有效。

3. tuple和pair

tuple基本上是一个类，它可以包含x个变量，但不关心类型，

```cpp
#include <iostream>
#include <tuple>
#include <utility>

std::tuple<std::string, int> returnWithTuple()
{
    return std::make_tuple("Hello", 42);
}

std::pair<std::string, int> returnWithPair()
{
    return std::make_pair("Hello", 42);
}

int main()
{
    std::tuple<std::string, int> tupleResult = returnWithTuple();
    std::cout << std::get<0>(tupleResult) << ", " << std::get<1>(tupleResult) <<
std::endl;

    std::pair<std::string, int> pairResult = returnWithPair();
    std::cout << pairResult.first << ", " << pairResult.second << std::endl;

    return 0;
}
```

但是返回时语法上不能让我们知道变量是什么，所以Cherno总是用*struct*（结构体）来做。

4. struct

C++

```cpp
#include <iostream>

struct Result
{
    std::string str;
    int num;
};

Result returnWithStruct()
{
    Result result;
    result.str = "Hello";
    result.num = 42;
    return result;
}

int main()
{
    Result structResult = returnWithStruct();
    std::cout << structResult.str << ", " << structResult.num << std::endl;

    return 0;
}
```