**C++ tuple元组的基本用法(总结)**

**1，元组简介**

    tuple是一个固定大小的不同类型值的集合，是[泛化](https://so.csdn.net/so/search?q=%E6%B3%9B%E5%8C%96&spm=1001.2101.3001.7020)的std::pair。我们也可以把他当做一个通用的结构体来用，不需要创建结构体又获取结构体的特征，在某些情况下可以取代结构体使程序更简洁，直观。std::tuple理论上可以有无数个任意类型的成员变量，而std::pair只能是2个成员，因此在需要保存3个及以上的数据时就需要使用tuple元组了。

    tuple（[元组](https://so.csdn.net/so/search?q=%E5%85%83%E7%BB%84&spm=1001.2101.3001.7020)）在c++11中开始引用的。tuple看似简单，其实它是简约而不简单，可以说它是c++11中一个既简单又复杂的东东，关于它简单的一面是它很容易使用，复杂的一面是它内部隐藏了太多细节，要揭开它神秘的面纱时又比较困难。

**2，tuple的创建和初始化**

2. std::tuple<T1, T2, TN> t1; //创建一个空的tuple对象（使用默认构造），它对应的元素分别是T1和T2...Tn类型，采用值初始化。
3. std::tuple<T1, T2, TN> t2(v1, v2, ... TN); //创建一个tuple对象，它的两个元素分别是T1和T2 ...Tn类型; 要获取元素的值需要通过tuple的成员get<Ith>(obj)进行获取(Ith是指获取在tuple中的第几个元素，请看后面具体实例)。
4. std::tuple<T1&> t3(ref&); // tuple的元素类型可以是一个引用
5. std::make\_tuple(v1, v2); // 像pair一样也可以通过make\_tuple进行创建一个tuple对象

 tuple的元素类型为引用：

1. std::string name;
2. std::tuple<string &, int> tpRef(name, 30);
3. // 对tpRef第一个元素赋值，同时name也被赋值 - 引用
4. std::get<0>(tpRef) = "Sven";
6. // name输出也是Sven
7. std::cout << "name: " << name << '\n';

**3，有关tuple元素的操作**

1. 等价[结构体](https://so.csdn.net/so/search?q=%E7%BB%93%E6%9E%84%E4%BD%93&spm=1001.2101.3001.7020)

      开篇讲过在某些时候tuple可以等同于结构体一样使用，这样既方便又快捷。如：

1. struct person {
2. char \*m\_name;
3. char \*m\_addr;
4. int \*m\_ages;
5. };
7. //可以用tuple来表示这样的一个结构类型，作用是一样的。
8. std::tuple<const char \*, const char \*, int>

    2. 如何获取tuple元素个数

    当有一个tuple对象但不知道有多少元素可以通过如下查询：

1. // tuple\_size
2. #include <iostream> // std::cout
3. #include <tuple> // std::tuple, std::tuple\_size
5. int main ()
6. {
7. std::tuple<int, char, double> mytuple (10, 'a', 3.14);
9. std::cout << "mytuple has ";
10. std::cout << std::tuple\_size<decltype(mytuple)>::value;
11. std::cout << " elements." << '\n';
13. return 0;
14. }
16. //输出结果：
17. mytuple has 3 elements

    3.如何获取元素的值

获取tuple对象元素的值可以通过get<Ith>(obj)方法进行获取；

Ith - 是想获取的元素在tuple对象中的位置。

obj - 是想获取tuple的对象

1. // tuple\_size
2. #include <iostream> // std::cout
3. #include <tuple> // std::tuple, std::tuple\_size
5. int main ()
6. {
7. std::tuple<int, char, double> mytuple (10, 'a', 3.14);
9. std::cout << "mytuple has ";
10. std::cout << std::tuple\_size<decltype(mytuple)>::value;
11. std::cout << " elements." << '\n';
13. //获取元素
14. std::cout << "the elements is: ";
15. std::cout << std::get<0>(mytuple) << " ";
16. std::cout << std::get<1>(mytuple) << " ";
17. std::cout << std::get<2>(mytuple) << " ";
19. std::cout << '\n';
21. return 0;
22. }
24. //输出结果：
25. mytuple has 3 elements.
26. the elements is: 10 a 3.14

   tuple不支持迭代，只能通过元素索引(或tie解包)进行获取元素的值。但是给定的索引必须是在编译器就已经给定，不能在运行期进行动态传递，否则将发生编译错误：

1. for(int i=0; i<3; i++)
2. std::cout << std::get<i>(mytuple) << " "; //将引发编译错误

4.获取元素的类型

 要想得到元素类型可以通过tuple\_element方法获取，如有以下元组对象：

1. std::tuple<std::string, int> tp("Sven", 20);
3. // 得到第二个元素类型
5. std::tuple\_element<1, decltype(tp)>::type ages;  // ages就为int类型
7. ages = std::get<1>(tp);
9. std::cout << "ages: " << ages << '\n';
11. //输出结果：
12. ages: 20

 5.利用tie进行解包元素的值

 如同pair一样也是可以通过tie进行解包tuple的各个元素的值。如下tuple对象有4个元素，通过tie解包将会把这4个元素的值分别赋值给tie提供的4个变量中。

1. #include <iostream>
2. #include <tuple>
3. #include <utility>
5. int main(int argc, char \*\*argv) {
6. std::tuple<std::string, int, std::string, int> tp;
7. tp = std::make\_tuple("Sven", 25, "Shanghai", 21);
9. // 定义接收变量
10. std::string name;
11. std::string addr;
12. int ages;
13. int areaCode;
15. std::tie(name, ages, addr, areaCode) = tp;
16. std::cout << "Output: " << '\n';
17. std::cout << "name: " << name <<", ";
18. std::cout << "addr: " << addr << ", ";
19. std::cout << "ages: " << ages << ", ";
20. std::cout << "areaCode: " << areaCode << '\n';
22. return 0;
23. }
25. //输出结果：
26. Output:
27. name: Sven, addr: Shanghai, ages: 25, areaCode: 21

但有时候tuple包含的多个元素时只需要其中的一个或两个元素，如此可以通过std::ignore进行变量占位，这样将会忽略提取对应的元素。可以修改上述例程：

1. #include <iostream>
2. #include <tuple>
3. #include <utility>
5. int main(int argc, char \*\*argv) {
6. std::tuple<std::string, int, std::string, int> tp;
7. tp = std::make\_tuple("Sven", 25, "Shanghai", 21);
9. // 定义接收变量
10. std::string name;
11. std::string addr;
12. int ages;
13. int areaCode = 110;
15. std::tie(name, ages, std::ignore, std::ignore) = tp;
16. std::cout << "Output: " << '\n';
17. std::cout << "name: " << name <<", ";
18. std::cout << "addr: " << addr << ", ";
19. std::cout << "ages: " << ages << ", ";
20. std::cout << "areaCode: " << areaCode << '\n';
22. return 0;
23. }
25. //输出结果：
26. Output:
27. name: Sven, addr: , ages: 25, areaCode: 110

   6. tuple元素的引用

      前面已经列举了将引用作为tuple的元素类型。下面通过引用搭配make\_tuple()可以提取tuple的元素值，将某些变量值设给它们，并通过改变这些变量来改变tuple元素的值：

1. #include <iostream>
2. #include <tuple>
3. #include <functional>
5. int main(int argc, char \*\*agrv) {
7. std::tuple<std::string, int, float> tp1("Sven Cheng", 77, 66.1);
9. std::string name;
10. int weight;
11. float f;
13. auto tp2 = std::make\_tuple(std::ref(name), std::ref(weight), std::ref(f)) = tp1;
15. std::cout << "Before change: " << '\n';
16. std::cout << "name: " << name << ", ";
17. std::cout << "weight: " << weight << ", ";
18. std::cout << "f: " << f << '\n';
20. name = "Sven";
21. weight = 80;
22. f = 3.14;
24. std::cout << "After change: " << '\n';
25. std::cout << "element 1st: " << std::get<0>(tp2) << ", ";
26. std::cout << "element 2nd: " << std::get<1>(tp2) << ", ";
27. std::cout << "element 3rd: " << std::get<2>(tp2) << '\n';
29. return 0;
30. }
32. //输出结果：
33. Before change:
34. name: Sven Cheng, weight: 77, f: 66.1
35. After change:
36. element 1st: Sven, element 2nd: 80, element 3rd: 3.14