pair, tuple

09 June 2021 07:07

1. Pair

   couples together a pair of values, which may be of different types
   Constructed using ()
   ```
   std::pair<std::string,double> product2 ("tomatoes",2.30);
   product1 = std::make_pair(std::string("lightbulbs"),0.99);
   ```

2. Member access of pair - first, second, get<number>(pair)
   ```
     product2.first = "shoes";                  // the type of first is string
     product2.second = 39.90;                   // the type of second is double
   ```
   also can be got using
   ```
     get<n>(a)
   ```

3. std::tuple_element<# of the elem, type of tuple>::type

   0th element
   ```
   std::tuple_element<0, decltype(product1)>::type first = std::get<0>(product1);
   ```

4. std::tuple_size<type of tuple>::value

   `std::tuple_size<decltype(tup)>::value //` constexpr value of the unsigned integral type size_t

5. Object which holds collection of elements of different type
   ```
   std::tuple<int, char> a (10, 'a');
   std::tuple<int, char> b = std::make_tuple<int, char>(10, 'a')
   ```

6. Member access of tuple
   ```
   std::get<0>(tuple) //can be LHS and RHS
   ```

7. Relational operators can be used for comparing pair and tuple

   ==, != elementwise comparison in one go

   <, >, <=, >= lexographical comparison,
   involves comparing the elements that have the same position in both tuples sequentially from the beginning to the end using operator< reflexively as long as any such comparison returns true.

8. = operator can be used on tuple/pair for copy/move assignment. tuple1 = tuple2

   I
9. Individual tuple elements returned from function can be got as [a, b,…] = func() OR as a tuple?

   I
10. Creating pair/tuple without specifying type

    Before C++17, we used factory functions such as std::make_pair or std::make_tuple to create a std::pair or a std::tuple with or without specifying the type parameters.
    ```
        std::pair myPair(5, 5.5);           // deduces std::pair<int, double>
        std::tuple myTup(5, myArr, myVec); // deduces std::tuple<int, std::array<int, 3>, std::vector<double>>

        auto pair3 = std::pair(5, arg);// no need to give template args to std::pair
    ```

11. F

    I
12. F

    I
13. F
14. F

    I
15. F

    I
16. F

    I
17. F

    I
18. F

    I
19. F

I

20. D

I

21. F

I

22. F

I

23. F

I

24. F
25. F

I

26. F

I

27. F

I

28. F

I

29. F

I

30. F

I

31. D

I

32. F

I

33. F

I

34. F

I

35. F
36. F

I

37. F

I

38. F

I

39. F

I

40. F

I

41. F

I

42. D

I

43. F

I

44. F

I

45. F

I

46. F
47. F

I

48. F

I

49. F

I

50. F

I

51. F

I