C++ Programming

Structured Binding

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Structured Binding

- It is not an STL topic
- But it can make many code snippets **simpler**!
  - I hope in next C++ versions they make it more nicer!

```
 6    // C++17
 7    // a structured binding is an alias to an existing object.
 8    int arr[] { 2, 3, 5, 7 };
 9
10    auto [a1, b1, c1, d1] = arr;
11
12    cout<<c1<<"\n";          // 5
13    c1 += 20;
14    cout<<arr[2]<<"\n";      // 5
15
16    auto &[a2, b2, c2, d2] = arr;    // must be 4
17
18    cout<<c2<<"\n";          // 5
19    c2 = 20;
20    cout<<arr[2]<<"\n";      // 20
```

With struct

```
5  struct Employee {
6      string name;
7      int salary;
8  };
9
10 int main() {
11     Employee emp { "Mostafa", 20 };
12
13     // bind by value
14     auto [str1, sal1] = emp;
15
16     // bind by reference
17     auto &[str2, sal2] = emp;
18
19     sal2 += 30; // emp.salary now 50
20
21     const auto &[str3, sal3] = emp;
22
```

With pair

```
5  pair gp {5, 6};
6
7  //pair<int, int>& F_by_ref()    {   return gp;  }
8  auto& F_by_ref()    {   return gp;  }
9
10 auto F_by_val()     {   return gp;  }
11
12 int main() {
13     //pair<int, string> p { 33, "Belal" };
14
15     //auto p = pair(33, "Belal");   //Template Argument Deduction
16
17     pair p(33, "Belal");            //Template Argument Deduction
18
19     auto &[sal4, str4] = p;
20
21     auto &[x1, y1] = F_by_ref();    // & for return by reference
22     //auto &[x1, y1] = F_by_val();  // CE
23
24     auto [x2, y2] = F_by_val();     // ok, just waste of time/memory
25     auto &&[x3, y3] = F_by_val();   // && for rvalue ref (later in semantic moves)
26
```

With tuples

```
7     tuple<char, int, string> t = make_tuple('c', 10, "mostafa");
8     auto &[a, b, c] = t;
9
10    // Direct initialization
11    auto [m, n, q] = tuple('a', 123, true);
```

With map

```
int main() {
    multimap<int, int> mp1; // before c++11
    mp1.insert(pair<int, int>(1, 40));
    mp1.insert(pair<int, int>(1, 50));
    mp1.insert(pair<int, int>(3, 60));

    multimap<int, int> mp2 { { 1, 40 }, { 1, 50 }, {3, 60 } };

    for (const auto &[key, value] : mp2)
        cout<<key<<" "<<value<<"\n";

    // return pair of iterators
    auto [it_start, it_end] = mp2.equal_range(1);

    for (auto it = it_start; it != it_end; ++it)
        cout << it->first << " " << it->second << "\n";
        // 1 40     1 50

```

With insertion

```
5  int main() {
6
7      set<int> st {4, 20, 30, 30, 20};
8
9      // can't take a reference for a return by value
10     auto [iter1, is_inserted1] = st.insert(21); // true
11     auto [iter2, is_inserted2] = st.insert(30); // false
12
13     if (auto [iter3, success] = st.insert(15); success) {
14         // here if inserted and we have an iterator
15     }
```

More on initialization

```
int main() {

    string input_str = "I am mostafa\nYour instructor for this course";

    for (auto[iss, line] = pair(istringstream(input_str), string {}); getline(iss, line); ) {
        cout<<line<<"\n";
    }

/*
I am mostafa
Your instructor for this course
 */
```

Struct with Array

```
4
5  struct MyArray {
6      int salaries[3];
7  };
8
9  int main() {
10     int arr[] {1, 2, 3};
11     // CE: invalid conversion from 'int*' to 'int'
12     //int sal[4] {arr};
13
14     auto salPtr {arr};  // this is pointer
15
16     MyArray myarr {3, 4, 5};
17     auto [actuall_arr] = myarr; // NOT a pointer
18     assert(typeid(actuall_arr) == typeid(myarr.salaries));
19
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."