Structured binding in C++

Prerequisite : Tuples in C++

Structured binding is one of the newest features of C++17 that binds the specified names to subobjects or elements of initializer. In simple words, Structured Bindings give us the ability to declare multiple variables initialized from a tuple or struct. The main purpose of Structured Bindings in C++ 17 is to make the code clean and easy to understand. Like a reference, a structured binding is an ***alias to an existing object***. Unlike a reference, the type of a structured binding ***does not have to be a reference type***.

Syntax :

```
auto ref-operator(optional)[identifier-list] = expression;

// Or

auto ref-operator(optional)[identifier-list]{expression};

// Or

auto ref-operator(optional)[identifier-list](expression);
```

Parameters :

- **auto :** auto
- **ref operator :** either & or &&
- **identifier-list :** List of comma separated variable names.

- **expression :** An expression that does not have the comma operator at the top level (i.e, an assignment-expression), and has either array or non-union class type.

Basic Type Deduction :

Let **E** denote the type of the ***initializer expression***. E shall be either a specialization of **std::tuple**, or a type whose non-static data members are all accessible and are declared in the same base class of E. A structured binding declaration performs the binding in one of three possible ways, depending on E.

- **Case 1** : if E is an array type, then the names are bound to the array elements.
- **Case 2** : if E is a non-union class type and **tuple_size** is a complete type, then the “tuple-like” binding protocol is used.
- **Case 3** : if E is a non-union class type but **tuple_size** is not a complete type, then the names are bound to the public data members of E.

Let us see the advantage of Structure bindings over tuples with the help of an example :
Example 1 : In C++98

```
#include <bits/stdc++.h>
using namespace std;

// Creating a structure named Point
struct Point {
    int x;
    int y;
};

// Driver code
int main()
{
    Point p = {1, 2};

    int x_coord = p.x;
    int y_coord = p.y;

    cout << "X Coordinate : " << x_coord << endl;
    cout << "Y Coordinate : " << y_coord << endl;

    return 0;
}
```

Output :

```
X Coordinate : 1
Y Coordinate : 2
```

Example 2 : In C++11/C++14

```
#include <bits/stdc++.h>
#include <tuple>
using namespace std;

// Creating a structure named Point
struct Point
{
    int x, y;

    // Default Constructor
    Point() : x(0), y(0)
    {

    }

    // Parameterized Constructor for Init List
    Point(int x, int y) : x(x), y(y)
    {

    }
    auto operator()()
    {
        // returns a tuple to make it work with std::tie
        return make_tuple(x, y);
    }
};

// Driver code
int main()
{
    Point p = {1, 2};
    int x_coord, y_coord;
    tie(x_coord, y_coord) = p();

    cout << "X Coordinate : " << x_coord << endl;
    cout << "Y Coordinate : " << y_coord << endl;

    return 0;
}
```

Output :

```
X Coordinate : 1
```

```
Y Coordinate : 2
```

Example 3 : In C++17

```
#include <bits/stdc++.h>
using namespace std;

struct Point
{
    int x;
    int y;
};

// Driver code
int main( )
{
    Point p = { 1,2 };

    // Structure binding
    auto[ x_coord, y_coord ] = p;

    cout << "X Coordinate : " << x_coord << endl;
    cout << "Y Coordinate : " << y_coord << endl;

    return 0;
}
```

Output :

```
X Coordinate : 1
Y Coordinate : 2
```

Applications : Structured Binding can be used with arrays to get the elements from the array. In this case, E is an array type, hence the names are bound to the array elements. Below is the implementation to show the same :

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{

    int arr[3] = { 1, 2, 3 };

    // Here, E is an array type, hence the
    // names are bound to the array elements.
    auto[x, y, z] = arr;

    cout << x << " " << y << " " << z << endl;

    return 0;
}
```

Output :

```
1 2 3
```

Note : The number of identifiers in the identifier list must be equal to the number of elements in the array. If the number of identifiers in the identifier list is less, then either a compile time error or design time error may occur. This means that we cannot take the specific set of elements from the array.

A more practical example for using the structured bindings is as follows :

```cpp
#include <bits/stdc++.h>
#include <map>
using namespace std;

int main()
{
    // Creating a map with key and value
    // fields as String
    map<string, string> sites;

    sites.insert({ "GeeksforGeeks", "Coding Resources" });
    sites.insert({ "StackOverflow", "Q-A type" });
    sites.insert({ "Wikipedia", "Resources + References" });

    for (auto & [ key, value ] : sites)
    {
       cout << key.c_str() << " " << value.c_str() << endl;
    }

    return 0;
}
```

Output :

```
GeeksforGeeks Coding Resources
StackOverflow Q-A type
Wikipedia Resources + References
```

Note : Qualifiers such as const and volatile can be used along with type to make the declaration variable constant or volatile.

For more details on Structured bindings, you may refer : P0144R0

Recommended Posts:

Early binding and Late binding in C++

fill in C++ STL

Conditional or Ternary Operator (?:) in C/C++

forward_list insert_after() function in C++ STL

Count of distinct remainders when N is divided by all the numbers from the range [1, N]

C++ | asm declaration

Pointers and References in C++

Strings in C++ and How to Create them?

Enum Classes in C++ and Their Advantage over Enum DataType

Deque vs Vector in C++ STL

C++ Programming Basics

Introduction to C++ Programming Language

ios manipulators noshowpos() function in C++

ios rdstate() function in C++ with Examples

Gopi_Krishna_Menon
I am a C/C++ developer along with knowledge in C# XAML VBNET

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeks-forgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags : C++ cpp-advanced cpp-structure

Practice Tags : CPP

1

2.3

To-do Done

Based on **3** vote(s)

Feedback/ Suggest Improvement | Add Notes | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments