```
//////// Lecture 91: Initialiser Lists (std::intialiser_list<>)
// This is not to be confused with member Initialiser lists.
// In the previous videos, we discussed uniform initialisation.
// Syntax introduced in C++11.
// it provides a unified syntax for initialisaing scalar / array / user defined types and objects.
int x {0};
float y { 3.1f};
int arr[5] {3,1,2,3,2};
std::string s {"Hello World!"};

// Uniform initialisation introduced a new class called initialiser list, and it is used to store an array of objects.
#include <intialiser_list>
// then create an instance of this class.
// initialiser list is a class template, so you need to specify the type of elements that its going to store.

int main(){
    std::initialiser_list<int> data = {1,2,3,4};
    // If we use automatic type inference, then even in that case this brace list of elements will be inferred as an initialiser list.
    auto values = {1,2,3,4}; // type of values is an initialiser_list<int>
}
/*
What is the purpose of initialiser list?
- They are commonly used with container classes. Container classes are those classes that can hold objects of other classes.
- We can understand this by creating our own container class, we'll create a class called as Bag.
// This class will act like an array, and will be used to hold objects of other classes.
*/
class Bag {
```

```
28      int arr[10];
29      int m_size{};
30  public:
31      void Add(int val){if (m_size<10) arr[m_size++] =
            value;}
32      void Remove(){--m_size;}
33      int operator[] (int index){return arr[index];}
34      int GetSize()const {return m_size;}
35  }
36  int main()
37  {
38      Bag b;
39      b.Add(3);
40      b.Add(4);
41      b.Add(5);
42      for (int i = 0; i < b.GetSize(); i++)
43          std::cout << b[i] << std::endl;
44  }
45  // what if we want to initialise the bag with some
        predefined values without using Add().
46  // We could do something like this:
47  int main (){
48      Bag b {3,1,0}; // this will not work, but we know
            that this is a brace list of elements and the
            compiler will automatically infer this as an
            initialiser_list.
49      // So, to make this initialisation work, we'll
            add a constructor in Bag that accepts an
            initialiser list as an argument.
50  }
51
52  class Bag {
53      int arr[10];
54      int m_size{};
55  public:
56      Bag (std::initialiser_list<int> &values)
57      {
58          // we can access the elements inside the
                initialiser list using iterators. Iterators
                are provided as classes in C++, and provide
                some overloaded operators that help us
                access the elements inside the container.
```

```
59          // Let's initialise the Bag with the values
              inside the initialiser_list.
60          // For that, we'll have to use the iterators.
61          //So first of all, we'll create the iterator
              to the first element, and we'll do that by
              calling the begin() function of the
              initialiser list.
62          auto it = values.begin();
63          // and then we'll use the while loop, so the
              condition here will be if the beginning
              iterator is not equal to the end iterator
              on values,
64          while (it != values.end()){
65              // To access the values from the
                  iterator, we can use the asterisk
                  operator.
66              // Simply use the Add function here
67              // Imagine that your initialiser list is
                  like an array, and the iterator is just
                  a pointer to the array.
68              Add(*it);
69              // To shift the position of the iterator,
                  we increment it, like this:
70              ++it;
71          }
72      }
73      ... Other methods
74  }
75  int main (){
76      Bag b {3,1,0};// this works now!
77  }
78  // Write a random function called print() accepts an
      initialiser list of integers.
79  void Print(std::initialiser_list<int> values)
80  {
81      for (auto it = values.begin(); it !=
          values.end(); ++it){
82          std::cout << *it << std::endl;
83       }
84  }
85  // Remember that a brace list of elements
      automatically creates an initialiser list object.
```

```
int main(){
    Print({8,3,2,3,4,1,3}); // this works.
}

// Initialiser lists can also be used with a range
  based for loop.
void Print(std::initialiser_list<int> values){
    for (auto &v : values){
        std::cout << v << std::endl;
    }
}
int main(){
    Print({8,3,2,3,4,1,3}); // this works.
}

// We could have even used the curly braces within
  the for loop.
int main(){
    for (auto& v: {8,3,2,3,4,1,3}){
        std::cout << v << std::endl;
    }
}
```