LAMBDA EXPRESSION X

.:: EXAMPLES::.

- 1. <u>CLÁUSULA DE CAPTURA</u> (también conocida como *iniciador de expresión lambda* en la especificación de C++)
- 2. <u>LISTA DE PARÁMETROS</u> Opcional (también conocida como *declarador de expresión lambda*)
- 3. **ESPECIFICACIÓN MUTABLE** Opcional
- 4. ESPECIFICACIÓN DE EXCEPCIÓN Opcional
- 5. <u>TIPO DE VALOR DEVUELTO FINAL</u> Opcional
- 6. CUERPO DE LA EXPRESIÓN lambda



```
1
     #include <iostream>
 3
 4
     using namespace std;
 5
     int main(){
 6
      · int m = 0;
 8
     ---int n = 0;-
         [&m, n] (int a) mutable \{ m = ++n + a; \}(4);
10
        cout <<"m=> "<< m << endl;
        cout << "n=> - "<< -n << < endl;
11
12
13
Running /home/ubuntu/works
```

m=> 5 n=> 0

```
Syntax
           Meaning
           capture nothing
[=]
           All by value
           All by reference
[&]
[v, &r]
           v by val, r by ref
[=, &r] r by ref, rest by val
           v by val, rest by ref
[&, v]
           capture the instance.
           provide access to
[this]
           methods and data
           members
```



```
#include<iostream>
    #include<algorithm>
    #include<vector>
3
4
5
    using namespace std;
6
7
    template <typename T>
    void imprimirArr(T a, int n) {
8
        cout << "[";
9
        for (int i = 0; i < n; ++i) {
10
            cout << (i? ", " : "") << a[i];
11
12
13
        cout << "]\n";
14
    };
15
    int main(){
16
17
        vector<int> v{44,11,77,55,66,33,22,88};
18
        //
        sort(
19
20
            v.begin(),
21
            v.end(),
             [&](const int left, const int right){return left>right; }
22
23
        );
24
        imprimirArr(v,v.size());
25
26
27
        return 0;
28
Running /home/ubuntu/workspace/ESTDA
```

[88, 77, 66, 55, 44, 33, 22, 11]

Syntax	Meaning
[]	capture nothing
[=]	All by value
[&]	All by reference
[v, &r]	v by val, r by ref
[=, &r]	r by ref, rest by val
[&, v]	v by val, rest by ref
[this]	capture the instance. provide access to methods and data members



```
#include <algorithm>
 1
    #include<vector>
 2
 3
     #include<iostream>
 4
 5
     using namespace std;
 6
7
     int main(){-
8
         vector<int> vec{1,2,3,4,5,6,7,8,9,10,12,13,15,19,20};
9
         int multi = 5;
10
         size_t count = count_if(
11
12
             vec.begin(),
13
             vec.end(),
      ------[=](int num){return !(num%multi);}-
14
15
      • • • );-
16
         cout<<"Integer that are multiple of ";</pre>
17
         cout<<multi<<" -> " <<count<<endl;</pre>
18
19
     }
```

Syntax	Meaning
[]	capture nothing
[=]	All by value
[&]	All by reference
[v, &r]	v by val, r by ref
[=, &r]	r by ref, rest by val
[&, v]	v by val, rest by ref
[this]	capture the instance. provide access to methods and data members



```
#include <algorithm>
1
2
    #include<vector>-
    #include<iostream>
3
4
5
    using namespace std;
6
    int main(){-
7
       vector<int> vec{1,2,3,4,5};
8
9
   ---//---
10
11
    size t count = count if(-
    vec.begin(),
12
   vec.end(),
13
    [](int num)->bool{return num%2;}
14
    ····);-
15
16
    cout<<"Integer that are Odd=> "<<count<<endl;</pre>
17
18
19
    count=count if(
    vec.begin(),
20
    vec.end(),
21
            [](int num)->bool{return !(num%2);}
22
23
    ....);-
24
    cout<<"Integer that are Even=> "<<count<<endl;</pre>
25
    return(0);
26
```

27

 \mathbf{F}

Syntax	Meaning
[]	capture nothing
[=]	All by value
[&]	All by reference
[v, &r]	v by val, r by ref
[=, &r]	r by ref, rest by val
[&, v]	v by val, rest by ref
[this]	capture the instance. provide access to methods and data members

Integer that are Odd=> 3
Integer that are Even=> 2



```
2
    //factorial-
 3
 4
    #include <vector>-
 5
    #include <algorithm>
    #include <iostream>
 6
7
    using namespace std;
 8
9
    int main(){-
10
11
    int arr[]={1,2,3,4,5};
        -auto factorial = [](int i, int j) {return i * j;};
12
13
    auto res = accumulate(arr, arr+5, 1, factorial);
14
15
        cout<<"5!="<<res<<endl; // 120
16
    return(0);
17
18
    }-
```

Syntax	Meaning
[]	capture nothing
[=]	All by value
[&]	All by reference
[v, &r]	v by val, r by ref
[=, &r]	r by ref, rest by val
[&, v]	v by val, rest by ref
[this]	capture the instance. provide access to methods and data members

