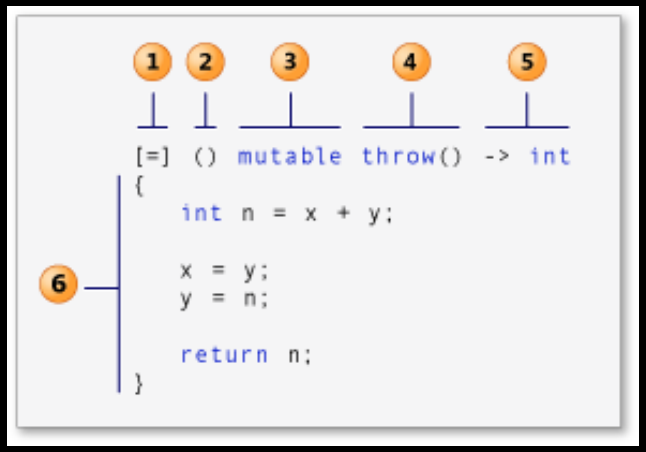
LAMBDA EXPRESSION 

.:: EXAMPLES::.



1. **CLÁUSULA DE CAPTURA** (también conocida como *iniciador de expresión lambda* en la especificación de C++)

2. **LISTA DE PARÁMETROS** Opcional (también conocida como *declarador de expresión lambda*)

3. **ESPECIFICACIÓN MUTABLE** Opcional

4. **ESPECIFICACIÓN DE EXCEPCIÓN** Opcional

5. **TIPO DE VALOR DEVUELTO FINAL** Opcional

6. **CUERPO DE LA EXPRESIÓN** *lambda*

```cpp
#include <iostream>

using namespace std;

int main(){
    int m = 0;
    int n = 0;
    [&m, n] (int a) mutable { m = ++n + a; }(4);
    cout <<"m=> "<< m << endl;
    cout <<"n=> "<< n << endl;

}
```

```
Running /home/ubuntu/work
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]	v by val, rest by ref
[this]	capture the instance. provide access to methods and data members

```cpp
#include<iostream>
#include<algorithm>
#include<vector>

using namespace std;

template <typename T>
void imprimirArr(T a, int n) {
    cout << "[";
    for (int i = 0; i < n; ++i) {
        cout << (i? ", " : "") << a[i];
    }
    cout << "]\n";
};

int main(){
    vector<int> v{44,11,77,55,66,33,22,88};
    //
    sort(
        v.begin(),
        v.end(),
        [&](const int left, const int right){return left>right; }
    );
    //
    imprimirArr(v,v.size());
    //
    return 0;
}
```

```
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

```cpp
#include <algorithm>
#include<vector>
#include<iostream>

using namespace std;

int main(){
    vector<int> vec{1,2,3,4,5,6,7,8,9,10,12,13,15,19,20};
    int multi = 5;
    //
    size_t count = count_if(
        vec.begin(),
        vec.end(),
        [=](int num){return !(num%multi);}
    );

    cout<<"Integer that are multiple of ";
    cout<<multi<<" => " <<count<<endl;
}
```

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

```cpp
#include <algorithm>
#include<vector>
#include<iostream>

using namespace std;

int main(){
    vector<int> vec{1,2,3,4,5};

    //--
    size_t count = count_if(
        vec.begin(),
        vec.end(),
        [](int num)->bool{return num%2;}
    );

    cout<<"Integer that are Odd=> "<<count<<endl;

    count=count_if(
        vec.begin(),
        vec.end(),
        [](int num)->bool{return !(num%2);}
    );
    cout<<"Integer that are Even=> "<<count<<endl;

    return(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]	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
```

```cpp
//factorial

#include <vector>
#include <algorithm>
#include <iostream>
using namespace std;

int main(){

    int arr[]={1,2,3,4,5};
    auto factorial = [](int i, int j) {return i * j;};

    auto res = accumulate(arr, arr+5, 1, factorial);
    cout<<"5!="<<res<<endl; // 120

    return(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]	v by val, rest by ref
[this]	capture the instance. provide access to methods and data members