**FUNCIONES LAMBDA**

Las expresiones lambda son conocidas como funciones anónimas ya que su característica fundamental es definir una función sin nombre de forma concisa

Esta se compone de tres partes principales; lista de parámetros (), operador flecha (→ o => o ->) que separa los parámetros del cuerpo, cuerpo de la función está siendo una expresión o un bloque de instrucciones.

Estas permiten pasar comportamientos como valores trabajando con funciones de orden superior.

Siendo su estructura general en C++ la siguiente:

* [captura]: Especifica qué variables del entorno pueden ser utilizadas dentro de la lambda.
* (parámetros): Lista de parámetros de entrada.
* → tipo\_retorno (opcional): Define el tipo de dato devuelto.
* { cuerpo }: Contiene las instrucciones que se ejecutan cuando la lambda es invocada.

Está caracterizándose por su anonimato ya que no tiene un nombre asociado, reduciendo su cantidad de código, puede acceder a variables de entorno donde se definen (captura de contexto).

Normalmente se usa auto el cual permite deducir el tipo de lambda de manera automática.

Captura de Variables

En c++ las funciones lambda son distintivas por su capacidad para capturar variables del entorno donde se definen, permitiendo que acceda o modifique valores externos a si misma, esta tiene tipos como [=] captura todas las variables externas por valor, [&] captura todas las variables externas por referencia, [x] captura solo la variable x por valor, [&x] captura solo la variable x por referencia, [=, &x] captura por valor, excepto x por referencia, [&. x] captura por referencia, excepto x por valor.

**Ejemplo de captura por valor:**

#include <iostream>

using namespace std;

int main() {

int valor = 10;

auto mostrar = [valor]() {

cout << "Valor capturado: " << valor << endl;

};

mostrar();

}

**Ejemplo de captura por referencia:**

#include <iostream>

using namespace std;

int main() {

int contador = 0;

auto incrementar = [&contador]() { contador++; };

incrementar();

cout << "Contador: " << contador << endl;

}

También existen tipos de retorno siendo uno deducido automáticamente por el compilador si esta tiene como instrucción return y definirise explícitamente usando -> tipo.

**Ejemplo por deducido:**

auto cuadrado = [](int n) { return n \* n; };

Ejemplo con tipo explicito:

auto dividir = [](double a, double b) -> double {

return a / b;

};

Lambdas en clases, captura de this

Cuando se define lambda dentro de una clase se puede capturar el puntero this para acceder a los atributos del objeto.

También en Biblioteca Estandar (STL) se usan en for\_each, sort, find\_if, count\_if, etc, el closure es una función lambda junto con el entorno de variables de captura, entonces suventaja es que reduce la necesidad de funciones auxiliares, su desventaja es que si se abusa de ello pierde legibilidad el código.