Laboratorio 18

Metaprogramación

1. Competencias

1.1. Competencias del curso

Conoce, comprende e implementa programas usando metaprogramación del lenguaje de programación C++.

1.2. Competencia del laboratorio

Conoce, comprende e implementa programas usando metaprogramación del lenguaje de programación C++.

2. Equipos y Materiales

- Un computador.
- IDE para C++.
- Compilador para C++.

3. Marco Teórico

3.1. Metaprogramación

La metaprogramación o metaprogramming como decimos en inglés, es uno de los términos más desafiantes que un desarrollador puede encontrar, en resumen, solemos pensar en metaprogramación como la habilidad de usar código para generar código, el problema es que ir de, con metaprogramación puedo generar código a efectivamente aprovechar la implementación de metaprogramación de un lenguaje, hay un largo camino.

Algo muy importante que necesitas saber es que cada lenguaje tiene implementaciones distintas de metaprogramación, y que la línea que separa la metaprogramación del código dinámico, es muy difusa, lo que algunos consideran metaprogramación, otros no. Así que, ten eso en cuenta.

La metaprogramación es la escritura de programas de computación que a su vez escriben o manipulan otros programas (o a ellos mismos), tanto datos como código. Esta manipulación puede suceder tanto en tiempo de compilación como en tiempo de ejecución.

En algunos casos esta técnica posibilita reducir el número de líneas de código requeridas para expresar una solución. También permite escribir programas con una gran flexibilidad para manejar eficientemente nuevas situaciones sin la necesidad de recompilar.

Ejemplo:

El siguiente código es la implementación en C++ de la conocida función factorial:

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int factorial(int n)
{
	if(n == 0)
		return 1;
	return n * factorial(n - 1);
}
void main()
{
	int x = factorial(4);
	cout << x << endl;
	system("pause");
}
```

El resultado sería el siguiente:

```
24
```

En el caso anterior, se evalúa la función factorial con una constante, por lo tanto el cálculo de factorial(4) se podría hacer en tiempo de compilación usando metaprogramación con templates:

```
#include "stdafx.h"
#include <iostream>
using namespace std;

template <int N>
struct Factorial
{
	enum { value = N * Factorial <N - 1 >::value };
};

template <>
struct Factorial <0>
{
	enum { value = 1 };
};

void main()
{
	int x = Factorial<4>::value;
	cout << x << endl;
	system("pause");
}
```

El resultado sería el siguiente:

24

Se han usado templates y especialización de templates para recrear la función factorial, y que esta se calcule en tiempo de compilación.

El tipo enumerado (enum) es un tipo de datos definido por el usuario al que se le pueden asignar algunos valores limitados. Estos valores los define el programador al momento de declarar el tipo enumerado.

Sintaxis de enumerados:

La palabra clave enum se usa para declarar tipos enumerados después de que se escribió el nombre del tipo enumerado y luego, entre corchetes, se definen los valores posibles. Después de definir, se crean las variables de tipo enumerado.

enum nombre-tipo-enumerado {valor1, valor2, valor3… ..valorN};

4. Ejercicios

Resolver los siguientes ejercicios planteados:

1. Suma los dígitos de un numero de forma recursiva utilizando metaprogramación.

2. Calcular el valor de la posición fibonacci usando recursividad utilizando metaprogramación.

3. Calcula la potencia de un numero de forma recursiva utilizando metaprogramación.

4. Construya una función recursiva que convierta un número decimal en una cadena que represente el valor del número en hexadecimal (base 16) utilizando metaprogramación.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

5. Ingresar un número y mostrar su equivalente en binario usando una función recursiva utilizando metaprogramación.

0 0 0 0	0 0	0	0 0
0 0 0 1	0 1	1	0 1
0 0 1 0	0 2	2	0 2
0 0 1 1	0 3	3	0 3
0 1 0 0	0 4	4	0 4
0 1 0 1	0 5	5	0 5
0 1 1 0	0 6	6	0 6
0 1 1 1	0 7	7	0 7
1 0 0 0	1 0	8	0 8
1 0 0 1	1 1	9	0 9
1 0 1 0	1 2	A	1 0
1 0 1 1	1 3	B	1 1
1 1 0 0	1 4	C	1 2
1 1 0 1	1 5	D	1 3
1 1 1 0	1 6	E	1 4
1 1 1 1	1 7	F	1 5

5. Entregables

Al final estudiante deberá:

1. Compactar el código elaborado y subirlo al aula virtual de trabajo. Agregue sus datos personales como comentario en cada archivo de código elaborado.
2. Elaborar un documento que incluya tanto el código como capturas de pantalla de la ejecución del programa. Este documento debe de estar en formato PDF.
3. El nombre del archivo (comprimido como el documento PDF), será su LAB18_GRUPO_A/B/C_CUI_1erNOMBRE_1erAPELLIDO.

   (Ejemplo: LAB18_GRUPO_A _2022123_PEDRO_VASQUEZ).
4. Debe remitir el documento ejecutable con el siguiente formato:

   LAB18_GRUPO_A/B/C_CUI_ EJECUTABLE_1erNOMBRE_1erAPELLIDO

   (Ejemplo: LAB18_GRUPO_A_EJECUTABLE_2022123_PEDRO_VASQUEZ).

En caso de encontrarse trabajos similares, los alumnos involucrados no tendrán evaluación y serán sujetos a sanción.