Casos de Uso para proyecto de expansión de Triangle: Tipos Enumerados

Integrantes:

José Johel Jara Álvarez

David Morales Vargas

Ejemplo 1: Declaración y uso básico

```
enum Color {Rojo, Verde, Azul};


let c: Color in
c       := Verde;
```

Este ejemplo declara un tipo enumerado llamado Color con tres valores posibles (Rojo, Verde, Azul). Luego, se declara una variable c de tipo Color y se le asigna el valor Verde. Con esto se ejemplifica la declaración de un tipo enumerado y su uso básico en asignación.

Ejemplo 2: Uso en condicional

```
enum Dia {Lunes, Martes, Miercoles, Jueves, Viernes};


let d: Dia in
d       := Martes;
if d = Lunes then putint(1)
else
putint(0);
```

Aquí se define el tipo enumerado Dia que representa los días laborales de la semana. La variable d se asigna con el valor Martes. Posteriormente, se utiliza en una estructura condicional (if) para verificar si el valor de d es Lunes. Dependiendo del resultado, se imprime 1 o 0. Este ejemplo ilustra cómo los enumerados pueden usarse en comparaciones y estructuras de control.

Ejemplo 3: Uso en un arreglo

```
enum Estado {Encendido, Apagado};


let dispositivos: array
[0..2] of Estado in
dispositivos[0] :=
Encendido; dispositivos[1]
:= Apagado; dispositivos[2]
:= Encendido;
```

En este caso se define el tipo enumerado Estado con dos posibles valores (Encendido, Apagado). Luego se declara un arreglo de tamaño 3 donde cada posición almacena un valor de tipo Estado. Finalmente, se asignan valores diferentes a cada posición. Este ejemplo muestra cómo los enumerados pueden ser utilizados como tipo de datos en estructuras compuestas como los arreglos.

Ejemplo 4: Procedimiento con parámetro enumerado

```
enum Nivel {Bajo, Medio, Alto};


proc mostrar(n: Nivel)
if n = Bajo then putint(1)
else if n = Medio then putint(2) else
putint(3)


in
mostrar(Alto);
```

Se declara el tipo enumerado Nivel con tres valores (Bajo, Medio, Alto). Luego, se define un procedimiento mostrar que recibe un parámetro de tipo Nivel y, mediante condicionales, imprime un número asociado a cada nivel. Finalmente, se invoca el procedimiento con el

valor Alto. Este ejemplo ilustra cómo los enumerados pueden ser utilizados como parámetros en procedimientos o funciones.

Ejemplo 5: Enum con parámetros

```
enum Figura {Circulo(Radio: int), Cuadrado(Lado: int, Ancho: int)};

let f: Figura
in
    f := Circulo(10);
```

Este ejemplo muestra cómo un tipo enumerado puede tener parámetros asociados a cada constructor.

Ejemplo 6: Match con parámetros

```
match f of
    Circulo(r) => putint(r * 2); -- diámetro
    Cuadrado(l, a) => putint(l * a); -- área
end
```

Aquí se utiliza pattern matching sobre un enum con parámetros para obtener diferentes comportamientos.

Ejemplo 7: Match no exhaustivo

```
match f of
    Circulo(r) => putint(r);
end -- ERROR: no cubre el caso Cuadrado
```

Este ejemplo muestra un caso donde el match no es válido porque no se cubren todos los posibles constructores.

Ejemplo 8: Función con enum parametrizado

```
func perimetro(fig: Figura): int =>
    match fig of
        Circulo(r) => 2 * 3 * r;
        Cuadrado(l, a) => 2 * (l + a);
    end;
```

Ejemplo de función que recibe un enum con parámetros y retorna un valor según el patrón.

Ejemplo 9: Match en expresión

```
enum Semaforo {Rojo, Amarillo, Verde};

let s: Semaforo
in
    s := Verde;
    putint(match s of
        Rojo => 0
        Amarillo => 1
```

```
        Verde => 2
    end);
```

Este ejemplo ilustra el uso de match dentro de una expresión.

Ejemplo 10: Traslape de parámetros

```
enum Resultado {Exito(Msg: string), Error(Msg: string)};

let r: Resultado
in
    r := Error("falló");
    match r of
        Exito(m) => put(m);
        Error(m) => put(m);
    end
```

Este ejemplo presenta un posible traslape entre parámetros con el mismo tipo, mostrando un caso ambiguo.