

## Examen 2

(30 puntos)

### Introduccion

Las constantes definidas en el enunciado, en mi caso son:  $X = 2$ ,  $Y = 1$  y  $Z=6$  debido a los tres digitos finales de mi carnet institucional 16-10216

### Pregunta 1

De acuerdo a la inicial de mi apellido se escogio el Lenguaje de programacion C++.

#### a Breve descripcion:

C++ es un lenguaje multiparadigma basado en el lenguaje de programacion C. Su fuerte es la velocidad en la ejecucion y la capacidad de manipular a muy bajo nivel la memoria según como el programador desee.

Posee como estructuras de control de flujo los siguientes:

**Asignacion:** representado con el simbolo punto y coma “ ; ” aunque se utiliza mas como un indicador de terminacion de instrucciones, pero la secuenciacion se visualiza al colocar una instrucción justo luego de la anterior.

**Selección:** C++ cuenta con la selección simple **if** y la selección multiple **switch**

- **Simple:** con las instrucciones if - else if – else. Permite cambiar el flujo de ejecucion entre dos o mas opciones. Tiene la estructura siguiente:

```
if(condicion){
    codigo
    ...
}
else if (condicion){
    codigo
    ...
}
else
{
    codigo
    ...
}
```

- **Selección Multiple:** Con la instrucción switch permite cambiar el flujo de ejecución del programa según los diferentes valores posibles que toma una condición dada. Y si ninguno de los valores posibles es alcanzado entonces permite especificar una opción por defecto. Posee la siguiente estructura:

```
switch ( operador ) {

    case valorPosible1 :
        Codigo...
        break;

    case valorPosible2 :
        Codigo...
        break;
    ■
    ■
    ■
    case valorPosibleN :
        Codigo...
        break;

    default:
        Codigo...
}
```

**Estructuras de Repetición:** Estas permiten generar ciclos de código sujetos a ciertas condiciones. Tenemos las determinadas y las indeterminadas

- **Indeterminadas:** Ciclo **while** y **do – while**. Estas estructuras permiten repetir un bloque de código mientras cierta condición siga siendo verdadera. Es responsabilidad del programador verificar que el ciclo se detenga en un momento dado.
- **Determinadas:** Ciclo **for** permite repetir un bloque de código al igual que el while, solo que en su estructura permite ciclar en rangos de valores definirlos e incrementar el contador al mismo tiempo. Sin embargo en C++ el ciclo **for** es simplemente otro sabor del ciclo while. Dado que es posible que todo ciclo while se pueda implementar como ciclo for. Debido a que la semántica del ciclo for no restringe la forma de la condición a evaluar.

**Abstracción Procedural:** Permite el encapsulamiento de código para ser reusado. Esto mediante el uso de funciones. En C++ toda abstracción procedural debe retornar un tipo ya sea unitario o cualquier otro tipo. Además permite la **Recursión** y el compilador es capaz de optimizar una recursión de cola de tal manera que no se desperdicie espacio en la pila y su desempeño sea igual o hasta mejor que si la función hubiera sido iterativa.

En C++ se permite la concurrencia mediante la instrucción **thread** y se manejan las excepciones con **try** y **catch**.

- ii Los operadores se evalúan en diferentes ordenes, algunos operadores usan notación prefija como la negación **!** Otras posfijas como el posincremento **variable++** y otras infijas como la suma, producto, resta... En general las reglas de precedencias son como en la mayoría de lenguajes de programación: a continuación una lista con los operadores ordenados de mayor a menor precedencia. Si están en la misma fila es porque tienen la misma precedencia y se evalúan según sus reglas de asociación.

Operador	Asociatividad
::	ninguna
() [] . -> v++ ..._cast typeid	Izq a Der
- + ~ ! * & ++v --v sizeof new delete(tipo)	Der a Izq
->* .*	Izq a Der
* / %	Izq a Der
+ -	Izq a Der
<< >>	Izq a Der
< <= > >=	Izq a Der
== !=	Izq a Der
&	Izq a Der
^	Izq a Der
	Izq a Der
&&	Izq a Der
	Izq a Der
?:	Der a Izq
= *= /= += -= <<= >>= &=  = ^=	Der a Izq
,	Izq a Der

iii **Tipos de Datos:** C++ posee los siguientes datos fundamentales:

- **Caracter:** **Char** (también se puede entender como su correspondiente entero según su código ASCII), **wchar\_t**
- **Enteros:** **short**, **int**, **long** y **long long** cada uno varía con respecto al otro es en el tamaño de la representación en bytes
- **Números en coma flotante:** **float**, **double**, **long double** de igual forma cada uno varía en el tamaño de su representación a bajo nivel. Esto define el grado de precisión del número de coma flotante
- **Booleanos:** **bool** y los valores posibles son **true** y **false**
- **Unitario:** **void**

**Para definir nuevos tipos de datos** se utiliza la palabra reservada **class** al igual que en muchos otros lenguajes de programación orientados a objetos. Permite polimorfismos mediante herencia y la sobrecarga de constructores de clase. Además permite definir registros con **struct** y los registros variantes con **union**. Algunos tipos compuestos que posee C++ son los arrays, los maps, los stacks entre otros.

**El sistema de tipos** considera que dos tipos de datos compuestos son **equivalentes** si y solo si tienen el mismo nombre y contienen los mismos tipos básicos no necesariamente en el mismo orden. El sistema de tipos considera dos tipos como **compatibles** en algunos casos. En general la mayoría de los tipos en C++ son compatibles con los enteros **int** esto es así porque C++ lo mantuvo de C. Pero adicionalmente como C++ es Orientado a objetos permite la compatibilidad de tipos mediante la

herencia de clases. Por otro lado la **inferencia de tipos** es posible aunque con ciertos riesgos, dado que no siempre es posible para el compilador inferir el tipo de dato resultante para la instanciación de un tipo. La inferencia se hace mediante la palabra **auto** seguida al nombre de la nueva variable y luego su correspondiente inicialización.

b

i La implementación de los Numerales de Church está en el siguiente repositorio de Github:

[https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta\\_1/Church](https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_1/Church)

ii La implementación del Árbol Binario está en el siguiente repositorio de Github:

[https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta\\_1/BinaryTree](https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_1/BinaryTree)

## Pregunta 2

El manejador de expresiones booleanas así como sus pruebas está en el link:

[https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta\\_2/](https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_2/)

Por cierto me encantó hacer este ejercicio, fue muy entretenido : )

## Pregunta 3

El inciso a) y b) son documentos en pdf cargados en el repositorio a continuación junto al inciso c) con sus respectivas pruebas

[https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta\\_3/](https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_3/)

El ejercicio del generador de parentesis fue Mind Blowing pero me ayudó a pensar y programar de un modo diferente a como lo venía haciendo. Pase unos días entretenidos dibujando parentesis en todos lados jajaja.

## Pregunta 4

Según mis correspondientes X Y y Z, la función F corresponde a la función siguiente:

n si  $0 \leq n < 30$

F(n) =

$F(n-5) + F(n-10) + F(n-15) + F(n-20) + F(n-25) + F(n-30)$  si  $n \geq 2$

Los tiempos de ejecucion junto a las graficas, el codigo y las pruebas unitarias estan en el siguiente repositorio. Se observo que el algoritmo recursivo implementado con recursion tal cual la definicion tradicional tiene un desempeño exponencial mientras que las implementaciones tanto iterativa como recursiva de cola mostraron un desempeño lineal.

Repositorio: [https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta\\_4/](https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_4/)

## **Pregunta 5**

Repositorio: [https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta\\_5/](https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_5/)

## **Pregunta 6**

Repositorio: [https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta\\_6/](https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_6/)

**Todo el examen esta en este repositorio:**

<https://github.com/Eycer-usb/ci3641-Examen2/>