Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3641 – Lenguajes de Programación I
Abril–Julio 2022

Profesor: Ricardo Monascal Eros Cedeño 16-10216

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 lengaje 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.

Seleccion: 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
    ...
}
```

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

```
switch ( operador ) {

    case valorPosible1 :
        Codigo...
        break;

    case valorPosible2 :
        Codigo...
        break;
    ▪
    ▪
    ▪
    case valorPosiblen :
        Codigo...
        break;

    default:
        Codigo...
}
```

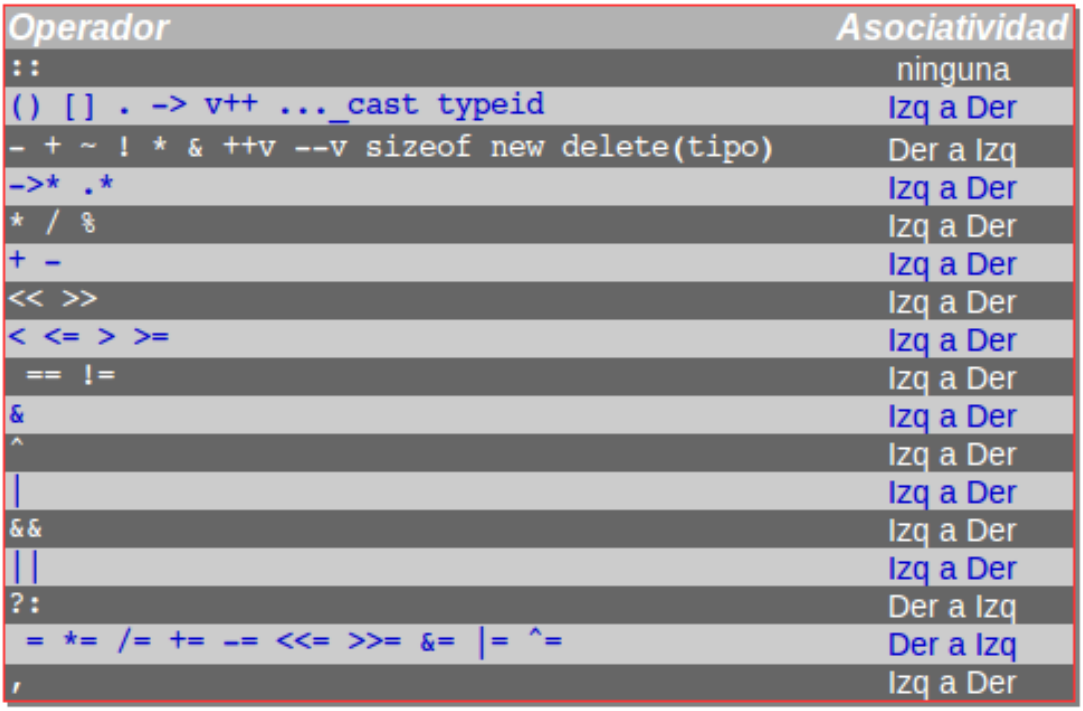
Estructuras de Repeticion: Estas permiten generar ciclos de codigo sujetas a ciertas condiciones. Tenemos las determinadas y las indeterminadas

- **Indeterminadas:** Ciclo **while** y **do – while**. Estas estructuras permiten repetir un bloque de codigo mientras cierta condicion siga siendo verdadera. Es reponsabilidad del programador verificar que el ciclo se detenga en un momento dado.
- **Determinadas:** Ciclo **for** permite repetir un bloque de codigo 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 semantica del ciclo for no restringe la forma de la condicion a evaluar.

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

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

ii Los operadores se evaluan en diferentes ordenes, algunos operadores usan notacion prefija como la negacion **!** Otras posfijas como el posincremento **variable++** y otras infijas como la suma, producto, resta… En general las reglas de precedencias son como en la mayoria de lenguajes de programacion: acontinuacion una lista con los operadores ordenados de mayor a menor precedencia. Si estan en la misma fila es porque tienen la misma presedencia y se evaluan según sus reglas de asociacion.



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** (tambien se puede entender como su correspondiente entero segun su codigo ascii), wchar_t
- **Enteros: short, int, log** y **long long** cada uno varia con respecto al otro es en el tamaño de la representacion en bytes
- **Números en coma flotante: float, double, long double** de igual forma cada uno varia en el tamaño de su representacion a bajo nivel. Esto define el grado de precision del numero de como 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 programacion orientados a objetos. Permite polimorfismos mediante herencia y la sobrecarga de constructores de clase. Ademas permite definir registros con **struct** y los registros variantes con **union.** Algunas tipos compuestos que posee C++ son los array, 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 basicos no necesariamente en el mismo orden. El sistema de tipos considera dos tipos como **compatibles** en algunos casos. En general la mayoria de los tipos en C++ con compatibles con los enteros **int** esto es asi 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 instanciacion de un tipo. La inferencia se hace mediante la palabra **auto** seguida al nombre de la nueva variable y luego su correspondiente inicializacion.

b

i La implementacion de los Numerales de Church esta en el siguiente repositorio de Github:

https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_1/Church

ii La implementacion del Arbol Binario esta en el siguiente repositorio de Github:

https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_1/BinaryTree

Pregunta 2

El manejador de expresiones booleanas asi como sus pruebas esta en el link:

https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_2/

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

Pregunta 3

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

https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_3/

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

Pregunta 4

Segun mis correspondientes X Y y Z, la funcion F corresponde a la funcion siguiente:

$$F(n) = \begin{cases} n & \text{si } 0 <= n < 30 \\ F(n-5) + F(n-10) + F(n-15) + F(n-20) + F(n-25) + F(n-30) & \text{si } n >= 2 \end{cases}$$

Los tiempos de ejecucion junto a las graficas, el codigo y las pruebas unitarias estan en el siguiente repositorio

Repositorio: **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/**

Pregunta 6

Repositorio: **https://github.com/Eycer-usb/ci3641-Examen2/tree/main/pregunta_**6/