**Universidad Centroamericana “José Simeón Cañas”**

**Análisis de Algoritmos  Ciclo 02/2019**

**Semana 2 - 19 al 23 de agosto del 2019**

# Guia de laboratorio 1

Repaso:

*Arreglos y Matrices*

*Punteros*

*Recursión por Cola y Posposición*

## Arreglos

Un arreglo es una colección que puede almacenar N elementos del **mismo tipo**, dispuestos en una secuencia.

Sintaxis:

- Inicializar un arreglo

```
<tipo> <identificador>[<num_elementos>];
```

```
int vector[10];
```

```
int N[] = {1, 2, 3, 6};
```

- Calcular el tamaño de un arreglo

```
int main()
{
    array<int,5> myarray{ 1, 2, 3, 4, 5 };
    cout << myarray.size();
    return 0;
}
```

- Como parámetro en una función

```
int SumaDeDatos(int datos[], int n)
{
      int suma=0;
      while (n > 0)
            suma += datos[--n];
      return suma;
}
```

## Matrices

Una matriz no es más que un arreglo **multidimensional**, son aquellos arreglos que tienen más de una dimensión y, en consecuencia, más de un índice. En este caso solo son arreglos de dos dimensiones.

Sintaxis:

- Inicializando una matriz

```
int matriz[2][3] = {51, 52, 53, 54, 55, 56};
```

- Como parámetro en una función

```
void gen_mat (int a[][N], int n)
{
      int i, j;
      for(i=0; i<n; i++)
            for(j=0; j<n; j++)
                  a[i][j]=random(N);
}
```

## Punteros

Un puntero es una variable que guarda una dirección en memoria que contenga algún tipo de información y son generalmente usados para la creación de listas personalizadas o árboles según la necesidad del programador.

Sintaxis:

- Definir una clase Nodo con su puntero

```
typedef struct Nodo {
      int dato;
      struct Nodo* sig;
} Nodo;
```

- Inicializar un la primera dirección como Null

```
Nodo* L = NULL;
```

- Crear una función para la asignación de la memoria para el Nodo

```
Nodo* crearNodo() {
      Nodo* n = (Nodo*)malloc(sizeof(Nodo));
      if (n == NULL) {
            printf("Error de memoria :(");
            exit(0);
      }
```

```
        return n;
}
```

- Después se crearán los métodos para recorrer la lista y llenarla según las necesidades del usuario que podría variar desde una lista de un tamaño predeterminado a una que inserte datos según lo que pida el usuario. En el siguiente fragmento de código es un ejemplo simple de como insertar un nodo nuevo al final de una lista.

```
void insertarFLista(Nodo* n,Nodo* v){
        while(v->sig!=Null){
                v=v->sig;
        }
        v->sig=n;
}
```

## Recursión por Posposición

Se divide el problema en dos partes, una parte pequeña cuya solución es conocida, llamada **caso trivial**, y otra parte grande por resolver.
La solución del problema se obtiene al combinar la solución de ambas partes.

La parte grande se resuelve mediante recursión dividiéndola nuevamente en una parte conocida y otra más pequeña por resolver.

***Ejemplo 1 – Factorial de un número***

$$n! = \begin{cases} 1 & n = 0 \\ n * (n-1)! & n > 0 \end{cases}$$

***Ejemplo 2 – Sucesión de Fibonacci***

$$f(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ (n-1) + (n-2) & n \geq 2 \end{cases}$$

***Ejemplo 3 – Suma de un arreglo de números***

$$Suma(A,n) = \begin{cases} A[0] & n = 1 \\ Suma(A,n-1) + A[n-1] & n > 1 \end{cases}$$

## Recursión por Cola

El resultado se calcula con los datos que se encuentren disponibles hasta el momento, de tal manera que cuando se llega al caso trivial, puede obtenerse directamente la respuesta.

Suelen utilizarse dos funciones:
La primer función generalmente se limita a llamar a la segunda, pasando, además de sus parámetros formales, una o más variables que contienen el valor que la función asignaría en el caso trivial.

La segunda función es la única función recursiva del problema.

***Ejemplos***: los mismos ejemplos de recursión por posposición.

## Tarea

1. Realizar un programa que realice la suma de matrices de forma recursiva de manera que C = A + B , e imprima la matriz C; las matrices son de 3x3.
2. Realizar un programa para calcular que dado un número x y un número n, calcule x^n de manera recursiva.
3. Realizar un programa que encuentre de manera recursiva, el menor valor de un arreglo.