PRÁCTICA 4. ANÁLISIS Y DISEÑO. Rubén Medina Zamorano

1 - Ejercicio 5 -> Función de evaluar polinomio

2 – Datos de entrada

Cabeza --> Tipo struct lista *. Puntero al primer elemento de la lista
X --> Tipo float. Valor de la incógnita para evaluar el polinomio.

3 – Datos de salida

Resultado --> Tipo float. Valor obtenido al evaluar el polinomio. Se debe inicializar a cero.

4 – Datos auxiliares

```
Struct lista{
        Float coef;
        Int exp;
        Struct lista *sig;
}//Estructura para la lista del polinomio
```

aux –> Tipo Struct lista*. Variable auxiliar para recorrer la lista. Se pone apuntando a NULL.

5 – Descripción del algoritmo

La función recoge la dirección de cabeza y el valor de x, después se declara una variable auxiliar aux para recorrer la lista y un resultado inicializado para almacenar la suma del polinomio. Se le pasa la dirección de cabeza al auxiliar para que apunte al primer elemento de la lista y entra en un bucle. El resultado se va incrementando según el producto del coeficiente del elemento apuntado y la potencia del valor x al exponente del elemento apuntado (No olvidar implementar la librería math.h). Finalmente aux va apuntado al siguiente elemento en cada vuelta hasta que aux sea NULL, momento en el que se sale del bucle y se devuelve el resultado.

6 – Pseudocódigo

```
Float evaluarPolinomio(struct lista* cabeza, float x){
        Struct lista* aux<-NULL;
        Float Resultado<-0;

        Aux<-cabeza;
        Mientras(aux!=NULL){
                Resultado<- (resultado+ (aux->coef)*(pow(x,aux->exp)));
                Aux<- (aux->sig);
```

```
        }
        Devolver resultado;
}

/*********************************************************************/
```

1 - Ejercicio 6 -> Función que comprueba si un contenedor está en la pila

2 – Datos de entrada

Cabeza --> Tipo struct pila *. Puntero que apunta al último elemento que ha entrado en la pila.
X --> Tipo entero. Código del contenedor para buscar

3 – Datos de salida

Encontrado --> Tipo int. Devuelve 1 o 0 dependiendo si lo encuentra o no.

4 – Datos auxiliares

```
Struct pila{
        int x;
        struct pila * sig;
}
```

5 – Descripción del algoritmo

Recibe cabeza y el valor del código x, además declara un auxiliar aux para recorrer la pila y un valor encontrado para devolver si está o no está el código. Aux se pone apuntado al elemento que apunta la cabeza. Después, se entra en un bucle comparando el valor de aux y encontrado. Si el código de aux es el valor pedido, ponemos encontrado a 1 y salimos del bucle, sino seguimos recorriendo la pila. Finalmente, devolvemos en valor de encontrado que sería 0 si no lo encuentra y 1 si encuentra el código.

6 – Pseudocódigo

```
Int compruebaContenedor(struct pila* cabeza, int x){
        Struct pila* aux;
        Int encontrado=0;

        Aux<-cabeza;
        Mientras((aux!=NULL)&&(encontrado==0)){
                Si((aux->x)==x){
                        Encontrado<-1;
                }sino{
```

```
                    Aux<-(aux->sig);
                }

        }
        Devolver (encontrado);
}
```