Ejercicios Resueltos Listas – Pilas – Colas

Ejercicio 1. Dada la siguiente declaración:

```
struct Componente {          TipoElemento elemento;
                             struct Componente *sig;      }
        typedef Componente *Puntero;
        struct Cola{      Puntero principio, final;       }
```

Implementar las siguientes operaciones:

- Crear una cola: inicializarla
- Primero: devuelve el elemento que ocupe la primera posición
- Encolar (enqueue): introduce un nuevo elemento en la cola
- SacarCola (dequeue): elimina el elemento en la cabeza de la cola
- Vacía: averigua si una cola no tiene nada en su interior

Solución:

```
void CrearCola (TCola *C )
//   Inicializa una cola, recibida como parámetro  con los valores: principio = null; final = null;
{     C->principio = null;
      C->final = null;     }
int Vacía (TCola C )
//   Decide si la cola C tiene algún elemento en su interior
{        Vacía = C->principio == null;     }
 TipoElemento Primero ( TCola C );
//   Devuelve en E el primer elemento de la cola
{        if Vacía( C )       printf ( "Cola vacía\n" );
         else    E = C->principio->elemento;       }
void Encolar ( TCola *C;  TipoElemento E )
//  Almacena el elemento E en la cola C, después del último
{   new ( nuevo );
    nuevo->elemento = E;
    nuevo->sig = null;
    if vacía (C )        C->principio = nuevo;
    else    C->final->sig = nuevo;
    C->final = nuevo;  }

void SacarCola ( TCola *C)
//   Elimina el primer elemento de la cola C
{  Puntero aux;
    if Vacía( C )       printf ( "Cola vacía\n" );
    else
    {     aux = C->principio;
          C->principio = C->principio->sig;
          free ( aux );
            if C->principio == null   C->final = null;    }     }
```

Ejercicio 2. Dada la siguiente declaración de lista:

```
struct Tlista *Artículo;
struct Articulo  {  int Codigo,Cont_Pedida;
                    TLista *Sig;     }
```

Se tiene una lista en la que se han introducido pedidos según el orden de llegada, por lo que puede haber pedidos del mismo artículo. Se pide escribir una función que, dada una lista como la anteriormente descrita, devuelva un único elemento por cada artículo, en el cual el campo **Cant_Pedida** tenga la suma de todas las cantidades pedidas de ese artículo.
No se pueden utilizar estructuras auxiliares, así como tampoco estructuradas de control como while, for o do/while.

Solución:

```
void Elimina(Tlista *L; int Cod; int *Suma);
{  Tlista Aux;
        if L !Null
           if L->Codigo ==Cod
           {     Aux = L;
                 Suma = Suma + L->Cant_Pedida;
                 L = L->Sig;
                 Free(Aux);
                 Elimina(L, Cod, Suma);    }
        else Elimina(L->Sig, Cod, Suma);   }
void Eli_Codigo(Tlista *Lista);
{       if Lista !=Null
        {       Elimina(Lista->Sig, Lista->Codigo, Lista->Cant_Pedida);
                Eli_Codigo(Lista->Sig);  }     }
```

Ejercicio 3. Se dispone de un compilador que únicamente tiene implementada la estructura de datos pila, y las siguientes operaciones asociadas.

- void Inicializar (Tpila *pila);
- void Apilar (Tpila *pila,Telem elem);
- void Sacar (Tpila *pila);
- Telem Cima (Tpila Pila, Telem Elemento);
- BOOLEAN Vacia (Tpila Pila);

Se pide implementar una función que invierta una pila utilizando las operaciones descritas anteriormente. Se podrán utilizar aquellas estructuras auxiliares que tengan implementadas el compilador. Se invertirá el contenido de la pila utilizando una pila auxiliar.

Solución

```
void intercambiar (Tpila * p1, *p2);
{       Telem  Elem;
        WHILE !Vacia(p1)
        {       Cima (p1,Elem);
                Apilar (p2,Elem);
                Sacar (p1);   }   }
main()
{       Tpila p0,p1,p2;
        Inicializar (p1);
        Intercambiar (p0,p1);
        Inicializar (p2);
        Intercambiar (p1,p2);
        Intercambiar (p2,p0);   }
```

Ejercicio 4. Construir un TAD pila en el que se van a introducir números enteros positivos de un dígito. Para ello se dispone del TAD conjunto, de enteros de dos dígitos en el que se han definido las siguientes operaciones:
Tdigitos = 0 .. 99;
Tconjunto= 10 .. 99 ;
- void Inicializar (Tconjunto *conjunto);
- void Introducir (Tconjunto *conjunto; Tdigitos digito);
- void Eliminar (Tconjunto *conjunto; Tdigitos digito);
- BOOLEAN Pertenece (Tconjunto conjunto; Tdigitos digito);

Se pide implementar la operación introducir un elemento en la pila usando el TAD conjunto.

Solución:
Vamos a hacer que la unidad sea el elemento y la decena el orden del elemento, por tanto: 0..9 sería el primer elemento,10..19 el segundo, ... hasta llegar a 90..99 que sería el décimo.

```
struct Tpila {
                Tconjunto conjunto;
                int Nelem;          }
void Apilar (struct Tpila *pila; int Elem)
{       if pila->Nelem >= 10     //   Pila llena
        else {
                  Introducir (pila->conjunto, pila->Nelem*10+Elem);
                   Pila->Nelem = pila->Nelem+1;  }     }
```

Ejercicio 5. Invertir una pila sin ninguna estructura auxiliar, y con las operaciones: cima, sacar, vacía, introducir.

Solución:

```
void Invertir (Tpila *p)
{   Telemento a;
        if !Vacia(p)
        {       Cima(p,a);
                Sacar(p);
                Invertir(p);
                InsertaFinal(p,a);   }    ]
void InsertaFinal (Tpila *p; Telemento a)
{       Telemento elem;
        if !Vacia(p)
        {       Cima(p,elem);
                Sacar(p);
                InsertaFinal(p,a);
                Introducir(p,elem);   }
        else   Introducir (p,a);    }
```