UNIVERSIDAD NEBRIJA

Escuela Politécnica Superior

IIN151 – Estructuras de datos y algoritmos

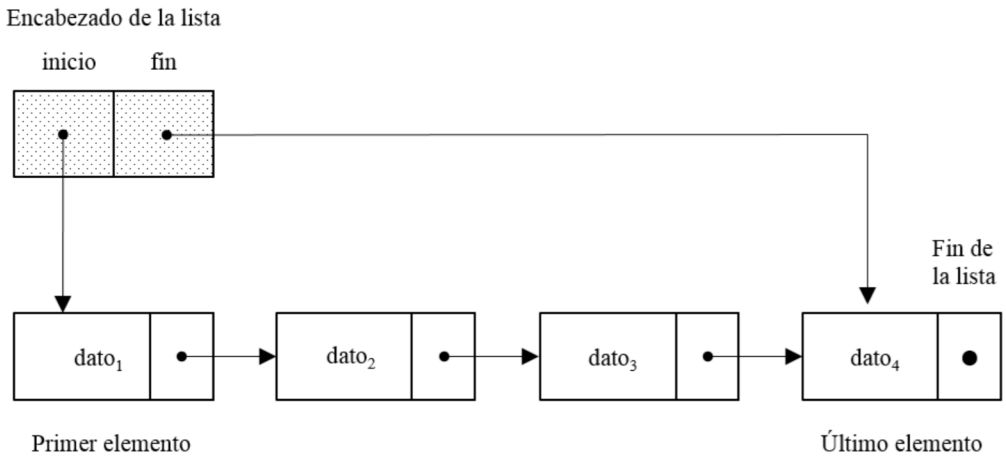
Prácticas de laboratorio 2

Nombre: _____

Una lista es un tipo abstracto de datos lineal. El TAD lista es una estructura de datos básica con múltiples aplicaciones. Además, una lista se puede utilizar para implementar estructuras de datos más complejas.

Las listas son estructuras de almacenamiento dinámicas, no tienen las limitaciones de los arrays. Una lista es una colección de nodos que almacenan datos y un puntero al nodo siguiente. Si el puntero al nodo siguiente es nulo, entonces se ha llegado al final de la lista.

El encabezado de la lista almacena un puntero al primer nodo y un puntero al último nodo de la lista.



La interfaz del TAD lista.

```
inicializa(L) Inicializa L a una lista vacía

  Precondición:  Ninguna
  Postcondición: Devuelve la lista L vacía

vacia(L) Devuelve verdadero si la lista L está vacía y falso en
         cualquier otro caso

  Precondición:  Ninguna
  Postcondición: true si la lista L está vacía y false e.o.c.

inicio(L) Devuelve el puntero al primer elemento de la lista

  Precondición:  Ninguna
  Postcondición: Devuelve un puntero al nodo que almacena el
                 primer elemento de la lista L

fin(L) Devuelve el puntero al último elemento de la lista

  Precondición:  Ninguna
  Postcondición: Devuelve un puntero al nodo que almacena el
                 último elemento de la lista L

busca(L, x) Devuelve el puntero al nodo que almacena el valor x en
            la lista y NULL en cualquier otro caso

  Precondición:  Ninguna
  Postcondición: Devuelve el puntero al nodo que almacena el
                 valor x en la lista L, si no lo encuentra
                 devuelve NULL

insertaInicio(L, x) Inserta el elemento x en la primera posición de
                    la lista

  Precondición:  Ninguna
  Postcondición: El puntero inicio de la lista apunta al
                 nuevo nodo que almacena el elemento x

insertaFin(L, x) Inserta el elemento x en la última posición de la
                 lista

  Precondición:  Ninguna
  Postcondición: El puntero fin de la lista apunta al
                 nuevo nodo que almacena el elemento x
```

```
insertaAntes(L, x, p) Inserta el elemento x en la posición p de la
                      lista

  Precondición:  Ninguna
  Postcondición: La lista tiene un nuevo nodo con el
                 elemento x antes del puntero p

insertaDespues(L, x, p) Inserta el elemento x en la posición
                        siguiente a p de la lista

  Precondición:  Ninguna
  Postcondición: La lista tiene un nuevo nodo con el
                 elemento x después del puntero p

elimina(L, p) Elimina el nodo p de la lista

  Precondición:  Ninguna
  Postcondición: La lista L no contiene el nodo p

elimina(L, x) Elimina el elemento x de la lista

  Precondición:  Ninguna
  Postcondición: La lista no contiene la primera ocurrencia
                 del nodo con el elemento x

imprime(L) Imprime los elementos de la lista L en la consola

  Precondición:  Ninguna
  Postcondición: Ninguna
```

Implemente el TAD lista utilizando el siguiente código.

```
struct NodoLista {
   int dato;
   NodoLista *siguiente;
};

struct TLista {
   NodoLista  *inicio, *fin;
};

void inicializa(struct TLista *L) {
   L->inicio = NULL;
   L->fin = NULL;
}

bool vacia(struct TLista *L) {
   return (L->inicio == NULL);
}

NodoLista* inicio(struct TLista *L) {
   return L->inicio;
}

NodoLista* fin(struct TLista *L) {
   return L->fin;
}

NodoLista* busca(struct TLista *L, int v) {
   NodoLista *p = L->inicio;

   if (p != NULL)
      while (p != NULL) {
         if (p->dato == v)
            break;

         p = p->siguiente;
      }

   return p;
}
```

```
void insertaInicio(struct TLista *L, int v) {
   NodoLista *tmp = new NodoLista;
   tmp->dato = v;

   if (vacia(L)) {
      tmp->siguiente = NULL;
      L->inicio = tmp;
      L->fin = tmp;
   }
   else {
      tmp->siguiente = L->inicio;
      L->inicio = tmp;
   }
}

void insertaFin(struct TLista *L, int v) {
   NodoLista *tmp = new NodoLista;
   tmp->dato = v;

   if (vacia(L)) {
      tmp->siguiente = NULL;
      L->inicio = tmp;
      L->fin = tmp;
   }
   else {
      tmp->siguiente = NULL;
      L->fin->siguiente = tmp;
      L->fin = tmp;
   }
}

void insertaAntes(struct TLista *L, int v, NodoLista *p) {
   if (p != NULL)
      if (p == L->inicio) {
         insertaInicio(L, v);
      }
      else {
          NodoLista *q = L->inicio;

          while (q->siguiente != p)
             q = q->siguiente;

          if (q->siguiente == p) {
             NodoLista *tmp = new NodoLista;
             tmp->dato = v;
             tmp->siguiente = p;
             q->siguiente = tmp;
          }
      }
}
```

```
void insertaDespues(struct TLista *L, int v, NodoLista *p) {
   if (p != NULL)
      if (p == L->fin) {
         insertaFin(L, v);
      }
      else {
         NodoLista *tmp = new NodoLista;
         tmp->dato = v;
         tmp->siguiente = p->siguiente;
         p->siguiente = tmp;
   }
}

void elimina(struct TLista *L, NodoLista *p) {

   // escriba el código para eliminar el nodo p de la lista
   // considerando los siguientes casos:
   //
   //   p es el inicio de la lista
   //   p es el fin de la lista
   //   p es un nodo intermedio de la lista

}

void elimina(struct TLista *L, int v) {

   // escriba el código para eliminar el nodo de la lista
   // que almacena el valor v

}

void imprime(struct TLista *L) {
   NodoLista *p = L->inicio;

   if (p == NULL)
      std::cout << "Lista vacia " << std::endl;
   else {
      std::cout << "Lista {";

      while (p != NULL) {
         std::cout << p->dato << ",";
         p = p->siguiente;
      }

      std::cout << "}" << std::endl;
   }
}
```

Modifique el TAD lista y defina las siguientes funciones:

```
void insertaOrdenado(struct TLista *L, int v) {

   // inserta un nodo con el valor v ordenado de forma ascendente

}

void elimina(struct TLista *L, int v) {

   // elimina todos los nodos de una lista que tengan el valor v

}

void concatena(struct TLista *L1,
               struct TLista *L2,
               struct TLista *C) {

   // dadas dos listas L1 y L2, devuelve la lista L3, resultado de
   // la concatenación de L1 y L2

}

void invierte(struct TLista *L1, struct TLista *L2 {

   // dada una lista L1, devuelve la lista L2 con los mismos nodos
   // que L1 pero en el orden inverso

}
```

Implemente una pila utilizando el TAD lista basado en punteros. Defina las siguientes funciones:

```
inicializa(P)
vacia(P)
apila(P, x)
desapila(P, x)
tope(P, x)
```

Para comprobar que el TAD pila funciona correctamente, desarrolle una función para calcular el valor de una expresión aritmética en notación postfija. Por ejemplo, el resultado de evaluar la expresión postfija 3 2 + 4 5 * + es 25.

```
void eval(char expresion[])
```