Universidad de Panama

Facultad de Informática Electrónica y Comunicación

Licenciatura en Ingeniería en Informática

Estudiante:

Irving Villarreal 8-1063-2312

Curso:

Programación II

Profesor:

Álvaro Pino

Tema:

Algoritmo de los 6 primeros ejercicios del libro

Fecha de Entrega:

27-06-2023

Ejercicio No. 1

Inserta Antes de una lista **Ordenada**

Este algoritmo inserta un nodo en una lista ordenada, mas no desordena la lista

* creamos **i** como apuntador del primer nodo

* dato representa el contenido del nuevo nodo

* **v, k y l** son variables apuntador, las cuales se van a desplazar en la lista.

* **num** y liga son campos de los nodos de la lista

1.Hacer band<-1 y v<-i

2.mientras((v->num ≠ x) && (band == 1)) Repetir

  2.1Si (v^. liga ≠NULL)

    entonces

      l<-v

      v<-v^. liga

    else

      band<-0

      break

  2.2 fin del condicional del paso 2.1

3.fin del ciclo del paso 2

4.Si(band=1)

  entonces

4.1Si(i=v)

entonces

k<-creaNodo ()

k^.num<-dato

k^. liga<-i

i<-k

Si no

4.1.1. Si((l^.num> dato) &&(v^.num<dato))

entonces

k<-creaNodo ()

k^.num<-dato

k^. liga<-k

k^. liga<-v

Sino

Escribir ("El número que ingreso es invalido")

4.1.2 fin de la condicional del paso 4.1.1.

4.2 fin del condicional 4.1

else

Escribir ("El numero dado no se encuentra")

5. fin del condicional del paso 4.

Ejercicio_2

Inserta Despues de una lista **Ordenada**

Este algoritmo insertar un nodo despues del otro dad como referencia, si la referencia es menor al nuevo nodo,

* creamos **i** como apuntador del primer nodo
* **x** representa el contenido del nuevo nodo
* **v y l** son variables apuntador, las cuales se van a desplazar en la lista.
* num y liga son campos de los nodos de la lista

1. hacer que band<-1 y v<-i
2. Mientras ((v^.num ≠ x) y(band=1)) repetir
   2.1si(v->liga! =NULL)
   entonces
      hacer v<-v^. liga;
   si no
      Hacer band<-0;
   2.2 Fin del condicional del paso 2.1
3. fin del ciclo del paso 2.
4. si (band=1)
   entonces
      4.1 si (v^.num < dato)

entonces

crea(l)

Hacer l^.num<-dato, l^. liga<-v^. liga y v^. liga<-l;

si no

Escribir ("\n\n***\tEl número que ingreso es invalido\n\n");

4.2 fin del ciclo del paso 4.1.

si no

Escribir ("\ndado como referencia es nulo");

5. fin del ciclo del paso 4.

Ejercicio 3

Insertar, en una lista **Ordenada**

Este algoritmo insertar un nodo en una lista ordenada, mas no desordena la lista

* creamos **i** como apuntador del primer nodo

* **dato** representa el contenido del nuevo nodo

* **v, k y l** son variables apuntador, las cuales se van a desplazar en la lista.

* num y liga son campos de los nodos de la lista

1.hacer band <- 1 y v<-i

2.Hacer((v->num<dato) y (band = 1)) mientras

2.1. Si (v^. liga≠NULL)

entonces

l<-v;

v<-v^. liga;

Si no

band<-0;

quebrar;

2.2 fin de la condición del paso 2.1

3. Fin del ciclo del paso 2

4.Si band=1

entonces

4.1. Si ((l^.num < dato) y (v^.num > dato) y (v ≠ i))

entonces

crear(k);

k^.num<-dato;

k^. liga<-v;

l^. liga<-k;

Si no

4.1.1. Si (v^.num>dato)

entonces

crear(k)

k^.num<-dato

l^. liga<-k

v<-k

i<-v

Sino

Escribir “Ya existe ese número"

4.1.2. fin del condicional del paso 4.1.1.

4.2. fin del condicional del paso 4.1.

Sino

Si ((v^.num < dato) y (v^. liga = NULL))

entonces

crear(k);

k^.num<-dato;

k^. liga<-NULL;

v^. liga<-k;

5. Fin del condicional del paso 4

Ejercicio 4

Eliminar, en una lista **Ordenada**

Este algoritmo inserta un nodo despues del otro dad como referencia, si la referencia es menor al nuevo nodo,

* creamos **i** como apuntador del primer nodo
* **x** representa el contenido del nuevo nodo
* **q y l** son variables apuntador, las cuales se van a desplazar en la lista.
* **num y liga** son campos de los nodos de la lista

1.hacer band <-1 y q<-i;
2.mientras ((q^.num ≠ x) && (band = 1)) repetir
  2.1si (q^. liga ≠ NULL)
    entonces
      l<-q;
      q<-q^. liga
    Si no
      band<-0
  2.2fin del condicional del paso 2.1
3.fin del ciclo del paso 2.
4.si(band=0)
  entonces

Escribir ("\n\nEl dato %d no se encuentra para eliminar\n", x);

Si no

4.1si(i=q)

entonces

i<-q^. liga;

Si no

l^. liga<-q^. liga;

4.2 fin del condicional del paso 4.1

5. fin del condicional del paso 4.

6.quitar(q);

Ejercicio 5

Mezclar dos listas ordenada y que devuelva ordenada

Este algoritmo insertar un nodo en una lista ordenada, mas no desordena la lista

* creamos p como apuntador del primer nodo de la primera lista
* creamos f como apuntador del primer nodo de la segunda lista
* creamos k como apuntador del primer nodo de la lista mezcladas de las dos listas
* dato representa el contenido del nuevo nodo
* k, w, r, q, t, x son variables apuntador, las cuales se van a desplazar en la lista.
* num y liga son campos de los nodos de la lista

1.hacer q<-p y t<-f

2. hacer ((q != NIL) && (t != NIL)) mientras

crear(w);

2.1 si (q^.num < t^.num)

entonces

w^.num <- q^.num;

q <- q^. liga

sino

w^.num <- t^. num.

t <- t^. liga.

2.2fin de la condicional del paso 2.1

w^. liga <- NIL;

2.3 si (k = NIL)

entonces

k <- w;

r <- w;

sino

r^. liga <- w;

r <- w;

2.4 fin del condicional del paso 2.3

3. si (q != NIL)

3.1. mientras (q != NIL) repetir

3.1.1. si (q != NIL)

crear(x);

x^.num <- q^.num;

x^.liga <- NIL;

r^. liga <- x;

3.1.2. fin del condicional del paso 3.1.1.

r <- x;

q <- q^. liga;

3.2 fin del ciclo del paso 3.1.

si sino (t != NIL)

3.3. mientras (t != NIL) repetir

3.3.1. si (t != NIL)

entonces

crear(x);

x^.num <- t^.num;

x^. liga <- NIL;

r^. liga <- x;

3.3.2. fin de la condicional del paso 4.1.1.

3.3.3. r <- x;

3.3.4. t <- t^. liga;

4. fin de la condicional del paso 3.

Ejercicio 6

Mezclar dos listas ascendentes y que devuelva una descendente

Este algoritmo insertar un nodo despues del otro dad como referencia, si la referencia es menor al nuevo nodo,

* creamos p como apuntador del primer nodo
* creamos f como apuntador del segundo nodo
* **z** representa el contenido del nuevo nodo
* **r, q, t** y quiere son variables apuntador, las cuales se van a desplazar en la lista.
* num y liga son campos de los nodos de la lista

1. hacer q<-q y t<-f
2. si (q = NULL)
   entonces
      retorna t;
3. fin de la condición del paso 1
4. si (t = NULL)
   entonces
      retorna q;
5. fin de la condición del paso 1
6. si (q^.num > t^.num)
   entonces

crear(r);

crear(z);

z^. num <- q^.num

r^. num <- t^.num

z^. liga <- r

r^. liga <- lista_3(q^. liga, t^. liga);

sino

crear(r);

crear(z);

r^.num <- t^.num

z^.num <- q^.num

z^. liga <- r

r^. liga <- lista_3(q^. liga, t^.liga);

7. fin de la condicional del paso 6