Clase #12

Martes 29 de Octubre de 2013

*Listas doblemente enlazadas. Listas circulares. Listas de listas.*

# Funciones para el trabajo con listas

### Borrar primer nodo

void borrarprimernodo(tpuntero \*cabeza){

tpuntero actual;

if(\*cabeza!=NULL){

actual=\*cabeza;

\*cabeza=(\*cabeza)->sig;

free(actual);

}

} //es igual a la de borrar lista solo que en vez de un while tiene un if

### Insertar al final

void insertaralfinal(tpuntero \*cabeza, int dato){

tpuntero actual, anterior, nuevo;

anterior=NULL;

actual=\*cabeza;

while(actual!=NULL){

anterior=actual;

actual=actual->sig;

}

nuevo= malloc (sizeog(tnodo));

nuevo->valor=dato;

nuevo->sig=NULL;

if(anterior!=NULL){

anterior->sig=nuevo;

}else{

\*cabeza=nuevo;

}

}

### Borrar el último

void borrarultimonodo(tpuntero \*cabeza){

tpuntero anteanterior, anterior, actual;

actual=\*cabeza;

anterior=NULL;

anteanterior=NULL;

while(actual!=NULL){

anteanterior=anterior;

anterior=actual;

actual=actual->sig;

}

if(anteanterior!=NULL){ //entonces la lista tiene 2 o más nodos

free(anterior);

anteanterior->sig=NULL;

}else if (anterior!=NULL){ //un solo nodo

free(anterior);

\*cabeza=NULL;

}

}

### Creación de una lista ordenada

void insertarenorden(tpuntero \*cabeza, int dato){

tpuntero actual, anterior, nuevo;

actual=\*cabeza;

anterior=NULL;

while(actual!=NULL && actual->valor < dato){

anterior=actual;

actual=actual->sig;

}

nuevo=(tpuntero)malloc(sizeof(tnodo));

nuevo->valor=dato;

nuevo->sig=actual;

if (anterior!=NULL){

anterior->sig=nuevo;

}else{

\*cabeza=nuevo;

}

}

# Listas enlazadas dobles

Las listas enlazadas dobles son listas que tienen enlaces en los dos sentidos. Para suplir las falencias de las listas simples, que no pueden recorrer la lista de modo inverso, solo lo recorren en una dirección. Las listas enlazadas dobles recorren la lista bidireccionalmente, es decir en ambos sentidos.

typedef struct snodod{

int valor;

struct snodod \*ant, \*sig;

}tnodod;

typedef tnodod \*tpunterod;

void insrtaralprincipiod(tpuntero \*cabeza, int dato){

tpunterod nuevo;

nuevo= malloc(sizeof(tnodod));

nuevo->valor=dato;

nuevo->ant=NULL;

nuevo->sig=\*cabeza;

if (\*cabeza!=NULL){

(\*cabeza)->ant=nuevo;

}

\*cabeza=nuevo;

}

void insertaralfinald(tpunterod \*cabeza, int dato){

tpunterod actual, anterior, nuevo;

actual=\*cabeza;

anterior=NULL;

while(actual!=NULL){

anterior=actual;

actual=actual->sig;

}

nuevo=malloc(sizeof(tnodod));

nuevo->valor=dato;

nuevo->ant=anterior;

nuevo->sig=NULL;

if(anterior!=NULL){

anterior->sig=nuevo;

}else{

\*cabeza=nuevo;

}

}

void insertarenordend(tpunterod \*cabeza, int dato){

tpunterod actual, anterior, nuevo;

actual=\*cabeza;

anterior=NULL;

while(actual!=NULL && actual->valor < dato){

anterior=actual;

actual=actual->sig;

}

nuevo=malloc(sizeof(tnodod));

nuevo->valor=dato;

nuevo->ant=anterior;

nuevo->sig=actual;

if(anterior!=NULL){

anterior->sig=nuevo;

}else{

\*cabeza=nuevo;

}

if(actual!=NULL){

actual->ant=nuevo;

}

}

void imprimirlistad (tpunterod cabeza){

tpunterod anterior;

anterior=NULL;

puts(“IDA”);

while(cabeza!=NULL){

printf(“%4d”,cabeza->valor);

anterior=cabeza;

cabeza=cabeza->sig;

}

puts(“REGRESO”);

while(anterior!=NULL){

printf(“%4d”,anterior->valor);

anterior=anterior->ant;

}

}

### Listas circulares

Son listas donde el último campo está dirigido al primero, de modo que se hace un circuito y no tendrá fin.

Ejercicio:

Dada una empresa de n vendedores, se posee la información de facturación por cada uno de los vendedores. Mostrar el total facturado por vendedor.

*Ver programa (03\_primer\_programa\_con\_lista\_simple.c)*