**PRIMITIVAS DE PILAS**

*-Estática:*

**void crearPila(t\_pila \*p)**

{

p->tope =0 ;

}

**int pilaVacia(const t\_pila \*p )**

{

return p->tope == 0;

}

**int pilaLlena(const t\_pila \*p)**

{

return p->tope == TAM;

}

**int hay\_dato(void)**

{ char resp;

printf("\n Desea insertar un numero en el vector: ");

do{ fflush(stdin);

scanf("%c",&resp);

}while(resp!='s'&& resp!='n');

if(resp=='s')

return 1;

else return 0;

}

**void cargarInfo(t\_info \*d)**

{

puts("Cargar la informacion(un numero entero)");

scanf("%d",&(d->info));

}

**int apilar(t\_pila \*p, const t\_info \*d)**

{

if(p->tope!=TAM)

{

p->info[p->tope]= \*d;

p->tope++;

return 1;

}

else return 0;

}

**int desapilar(t\_pila \*p, t\_info \*d)**

{

if(p->tope==0)

return 0;

(p->tope)--;

\*d= p->info[p->tope];

return 1;

}

**int verTope(const t\_pila \*p, t\_info \*d)**

{

if(p->tope==0)

return 0;

\*d = p->info[p->tope-1];

return 1;

}

**void vaciarPila(t\_pila \*p)**

{

p->tope=0;

}

*-Dinámica:*

**void crearPila(t\_pila \*p)**

{

\*p = NULL;

}

**int pilaVacia(const t\_pila \*p)**

{

return \*p==NULL;

}

**int pilaLlena(const t\_pila \*p)**

{

void \*aux = malloc(sizeof(t\_nodo));

free(aux);

return aux==NULL;

}

**void cargarInfo(t\_info \*d)**

{

puts("Cargar la informacion(un numero entero)");

scanf("%d",&(d->info));

}

**int apilar(t\_pila \*p, const t\_info \*d)**

{ t\_nodo \*nue = (t\_nodo\*) malloc(sizeof(t\_nodo));

if(nue==NULL)

return 0;

nue->info = \*d;

nue->sig=\*p;

\*p=nue;

return 1;

}

**int desapilar(t\_pila \*p, t\_info \*d)**

{

t\_nodo \*aux;

if(\*p==NULL)

return 0;

aux=\*p;

\*d=(aux->info);

\*p=(aux->sig);

free(aux);

return 1;

}

**int verTope(const t\_pila \*p, t\_info \*d)**

{

if(\*p==NULL)

return 0;

\*d=(\*p)->info;

return 1;

}

**void vaciarPila(t\_pila \*p)**

{

t\_nodo \*aux;

while(\*p != NULL)

{

aux=\*p;

\*p=aux->sig;

free(aux);

}

}

**PRIMITIVAS COLA**

*-Estática:*

**void crearcola(t\_cola \*p)**

{

p->ult=-1;

p->pri=0;

}

**int colavacía(const t\_cola \*p)**

{

return p->ult==-1;

}

**void vaciarcola(t\_cola \*p)**

{

p->ult=-1;

p->pri=0;

}

FALTA VER TOPE,ACOLAR Y DESACOLAR!!!

*-Dinámica:*

**void crearCola(t\_cola \*c)**

{

c->pri=NULL;

c->ult=NULL;

}

**int colaVacia (const t\_cola \*c)**

{

return (c->pri==NULL && c->ult==NULL);

}

**int colaLlena(const t\_cola \*c)**

{

void \*aux = malloc(sizeof(t\_nodo));

free(aux);

return aux==NULL;

}

**void cargarInfo(t\_info \*d)**

{

puts("Cargar la informacion(un numero entero)");

scanf("%d",&(d->info));

}

**int VerPrimero(const t\_cola \*c, t\_info \*d)**

{

if(c->pri == NULL)

return 0;

\*d = c->pri->info;

return 1;

}

**int Acolar(t\_cola \*c, const t\_info \*d )**

{

t\_nodo \*nue = (t\_nodo\*)malloc(sizeof(t\_nodo));

if(nue == NULL)

return 0;

nue->info = \*d;

nue->sig = NULL;

if(c->pri == NULL)

c->pri = nue;

else

c->ult->sig=nue;

c->ult=nue;

return 1;

}

**int Desacolar(t\_cola \*c,t\_info \*d)**

{

t\_nodo \* aux;

if(c->pri==NULL)

return 0;

aux = c->pri;

\*d = aux->info;

c->pri = aux->sig;

free(aux);

if(c->pri==NULL)

c->ult=NULL;

return 1;

}

**void VaciarCola(t\_cola \*c)**

{

t\_nodo \*aux;

while(c->pri != NULL)

{

aux=c->pri;

c->pri=aux->sig;//aux->sig = c->pri->sig

free(aux);

}

c->ult=NULL;

}

**PRIMITIVAS LISTA SIMPLEMENTE ENLAZADA**

**int insertaralfinal(t\_lista\* p,t\_info\* d)**

{

t\_nodo\* nue=(t\_nodo\*)malloc(sizeof(t\_nodo));

if(nue!=NULL)

return 0;

nue->info=\*d;

nue->sig=NULL;

while(\*p)

p=&(\*p)->sig;

\*p=nue;

return 1;

}

**int insertaralprincipio(t\_lista\* p,t\_info\* d)**

{

t\_nodo\* nue=(t\_nodo\*)malloc(sizeof(t\_nodo));

if(nue==NULL)

return 0;

nue->info=\*d;

nue->sig=\*p;

\*p=nue;

return 1;

}

**int insertarsinohayclave(t\_lista\* p,t\_info\* d)**

{

t\_nodo\* nue=(t\_nodo\*)malloc(sizeof(t\_nodo));

if(nue!=NULL)

return 0;

nue->info=\*d;

nue->sig=NULL;

while(\*p)

{

if(p->info.clave==nue->info.clave)

{

p->info.cantrep++;

free(nue);

return 1;

}

p=&(\*p)->sig;

}

\*p=nue;

return 1;

}

**int insertarenlistasinohayclavetrenchi(t\_lista\* p,t\_info\* d)**

{

while(\*p&&(\*p)->info.clave!=clave)

p=&(\*p)->sig;

if(\*p)

{

((\*p)->info.cantrep)++;

return 1;

}

\*p=(t\_nodo\*)malloc(sizeof(t\_nodo));

if(\*p==NULL)

return 0;

(\*p)->info.clave=clave;

(\*p)->info.cantrep=0;

(\*p)->sig=NULL;

return 1;

}

**int insertarordenadoactualizar(t\_lista \*p,const t\_info \*d, int(\*cmp)(const t\_info\*,const t\_info\*),void(\*ac)(t\_info\*))**

{

t\_nodo\* nue;

while(\*p&&cmp(&(\*p)->info,d)<=0)

p=&(\*p)\_>sig;

if(\*p&&cmp(&(\*p)->info,d)==0)

{

act(&(\*p)->info);

return TODO\_OK;

}

nue=(t\_nodo\*)malloc(sizeof(t\_nodo));

if(!nue)

return SIN\_MEMORIA;

nue->info=\*d;

nue->sig=\*p;

\*p=nue;

return TODO\_OK;

}

**int eliminartodaslasrepeticionesdeunalistadesordenado(t\_lista\* p,const t\_info\* info)**

{

t\_nodo\* nodo;

int cantel=0;

while(\*p)

{

if(cmp(&(\*p))->info,info==0)

{

cantel++;

aux=\*p;

\*p=aux->sig;

free(aux);

}

else

p=&(\*p)->sig;

}

return cantel;

}