

```

    P->cima = -1;
}
/* poner un elemento en la pila */
void Insertar(Pila* P, const TipoDato elemento)
{
    /* si la pila está llena, termina el programa */
    if (PilaLlena (*P))
    {
        puts("Desbordamiento pila");
        exit (1);
    }
    /* incrementar puntero cima y copiar elemento en listapila */
    P->cima++;
    P->listapila[P->cima] = elemento;
}
/* Quitar un elemento de la pila */
TipoDato Quitar(Pila* P)
{
    TipoDato Aux;
    /* si la pila está vacía, termina el programa */
    if (PilaVacía (*P))
    {
        puts("Se intenta sacar un elemento en pila vacía");
        exit (1);
    }
    /* guardar elemento de la cima */
    Aux = P->listapila[P->cima];

    /* decrementar cima y devolver valor del elemento */
    P->cima--;
    return Aux;
}
/* verificar pila vacía */
int PilaVacía(Pila P)
{
    /* devuelve el valor lógico resultante de expresión cima == -1 */
    return P.cima == -1;
}
/* verificar si la pila está llena */
int PilaLlena(Pila P)
{
    return P.cima == MaxTamaPila-1;
}
/* quitar todos los elementos de la pila */
void LimpiarPila(Pila* P)
{
    P->cima = -1;
}
TipoDato Cima(Pila P)
{
    if (P.cima == -1)
    {
        puts("Se intenta sacar un elemento en pila vacía");
    }
}

```

```

exit (1);
}
return P.listapila[P.cima];
}
/* Archivo paldromo.c */
typedef char TipoDato;
#include "pilarray.h"
#include <ctype.h>
int main()
{
    char palabra[100], ch;
    Pila P;
    int j, palmo;
    CrearPila(&P);
    /* Lee la palabra */
    do {
        puts("\n Palabra a comprobar si es palíndromo");
        for (j = 0; (ch = getchar()) != '\n'; )
        {
            palabra[j++] = ch;
            Insertar(&P, ch); /* pone en la pila */
        }
        palabra[j] = '\0';
        /* comprueba si es palíndromo */
        palmo = 1;
        for (j = 0; palmo && !PilaVacía(P); )
        {
            palmo = palabra[j++] == Quitar(&P);
        }
        LimpiarPila(&P);
        if (palmo)
            printf("\n La palabra %s es un palíndromo \n", palabra);
        else
            printf("\n La palabra %s no es un palíndromo \n", palabra);
        printf("\n ¿ Otra palabra?: "); scanf("%c", &ch);
        while (tolower(ch) == 's');
    }
    return 0;
}

```

19.3. COLAS

Una cola es una estructura de datos que almacena elementos en una lista y permite acceder a los datos por uno de los dos extremos de la lista (Fig. 19.4). Un elemento se inserta en la cola (parte final) de la lista y se suprime o elimina por la frente (parte inicial, cabeza) de la lista. Las aplicaciones utilizan una cola para almacenar elementos en su orden de aparición o concurrencia.

Los elementos se eliminan (se quitan) de la cola en el mismo orden en que se almacenan y, por consiguiente, una cola es una estructura de tipo **FIFO** (*first-in/first-out*, primero en entrar/primero en salir o bien primero en llegar/primero en ser servido). El servicio de atención a clientes en un almacén es un ejemplo típico de cola. La acción de gestión de memoria intermedia (*buffering*) de trabajos o tareas de impresora en un distribuidor de