

Introducción a la Programación I – Práctico 3 (Estrategias)

```
Program elp3;
  {pasamos de la pila dada a la pila mayor, el mayor elemento}
uses estructu;
var dada,mayor,aux:pila;
begin
  readpila(dada);
  inicpila(mayor, '');
  inicpila(aux, '');
  if not pilavacia (dada) then
    apilar(mayor, desapilar(dada));
  while (not pilavacia (dada)) and
    (not pilavacia (mayor)) do
    begin
      if (tope (dada) < tope (mayor)) then
        apilar(aux, desapilar(dada));
      if (not pilavacia (dada)) and
        (tope (dada) > tope (mayor)) then
        begin
          while not pilavacia (mayor) do
            apilar(aux, desapilar(mayor));
            apilar(mayor, desapilar(dada));
          end;
        if (not pilavacia (dada)) and
          (tope (dada) = tope (mayor)) then
            apilar(mayor, desapilar(dada));
        end;
      while not pilavacia (aux) do
        apilar(dada, desapilar(aux));
      writepila(mayor);
    end.

  /***** /
```

```
Program e2p3;  
{elimina los elementos repetidos de una pila dada}  
uses estructu;  
var dada,aux1,aux2,eliminar:pila;  
begin  
  readpila(dada);  
  inicpila(aux1, '');  
  inicpila(aux2, '');  
  inicpila(eliminar, '');  
  while not pilavacia (dada) do  
    begin  
      apilar(aux1, desapilar(dada));  
      while not pilavacia (dada) do  
        if (tope (dada) = tope (aux1)) then  
          apilar(eliminar, desapilar(dada))  
        else  
          apilar(aux2, desapilar(dada));  
        while not pilavacia (aux2) do  
          apilar(dada, desapilar(aux2));  
        end;  
      while not pilavacia (aux1) do  
        apilar(dada, desapilar(aux1));  
      writepila(dada);  
    end.  
  
/*****/
```

```
Program e3p3;  
{verifica si una pila es capicua}  
uses estructu;  
var dada, invdada, aux1, aux2, verdadero, falso, resultado:pila;  
begin  
  readpila(dada);  
  inicpila(aux1, '');  
  inicpila(aux2, '');  
  inicpila(invdada, '');  
  inicpila(verdadero, '');  
  inicpila(falso, '');  
  inicpila(resultado, '1');  
  while not pilavacia (dada) do  
    begin  
      apilar(aux1, desapilar(dada));  
      while not pilavacia (dada) do  
        apilar(invdada, desapilar(dada));  
        if (not pilavacia (invdada)) and  
          (tope (invdada) = tope (aux1)) then  
          begin  
            apilar(aux2, desapilar(invdada));  
            while not pilavacia (invdada) do  
              apilar(dada, desapilar(invdada));  
          end;  
        end;  
      if (pilavacia (invdada)) then  
        apilar(verdadero, desapilar(resultado))  
      else  
        apilar(falso, desapilar(resultado));  
      writepila(verdadero);  
      writepila(falso);  
    end.  
  
/*****/
```

```
Program e4p3;  
{divide la pila dada a la mitad respetando el orden en ambas partes}  
uses estructu;  
var dada, invdada, aux1, mitad1, mitad2:pila;  
begin  
  readpila(dada);  
  inicpila(aux1, '');  
  inicpila(mitad1, '');  
  inicpila(invdada, '');  
  inicpila(mitad2, '');  
  while not pilavacia (dada) do  
    begin  
      apilar(aux1, desapilar(dada));  
      while not pilavacia (dada) do  
        apilar(invdada, desapilar(dada));  
        if (not pilavacia (invdada) then  
          apilar(mitad2, desapilar(invdada));  
        while not pilavacia (invdada) do  
          apilar(dada, desapilar(invdada));  
        end;  
      while not pilavacia (aux1) do  
        apilar(mitad1, desapilar(aux1));  
      writepila(mitad1);  
      writepila(mitad2);  
    end.  
  
  /*****/
```

```
program E5P3;
{este programa realiza la union de las pilas 'a' y 'b' formando asi una
pila 'c'}
uses estructu;
var a, b, c:pila;
procedure repetidos;
{este procedimiento saca los elementos de 'a' que ya estan 'b'}
var auxb, auxa, eliminar:pila;
begin
  inicpila(auxb, '');
  inicpila(auxa, '');
  inicpila(eliminar, '');
  while not pilavacia(a) do
  begin
    while not pilavacia(b) do
      if tope(a)=(tope(b)) then
        apilar(eliminar, desapilar(b))
      else
        apilar(auxb, desapilar(b));
      while not pilavacia(auxb) do
        apilar(b, desapilar(auxb));
      apilar(auxa, desapilar(a));
    end;
    while not pilavacia(auxa) do
      apilar(a, desapilar(auxa));
    end;
  begin
    readpila(a);
    readpila(b);
    inicpila(c, '');
    repetidos;
    while not pilavacia(a) do
      apilar(c, desapilar(a));
    while not pilavacia(b) do
      apilar(c, desapilar(b));
    write('pila c: ');
    writepila(c);
  end.

/*****/
```

```
Program e6p3;  
{intercala 2 filas ordenadas en forma creciente, dejando el resultado  
tambien ordenado en forma creciente}  
uses estructu;  
var ordenada1, ordenada2, ordenadafinal:fila;  
begin  
  readfila(ordenada1);  
  readfila(ordenada2);  
  inicfila(ordenadafinal, '');  
  while (not filavacia(ordenada1)) and  
    (not filavacia(ordenada2)) do  
    if (primero(ordenada1) < primero(ordenada2)) then  
      agregar(ordenadafinal, extraer(ordenada1))  
    else  
      agregar(ordenadafinal, extraer(ordenada2));  
  while not filavacia (ordenada1) do  
    agregar(ordenadafinal, extraer(ordenada1));  
  while not filavacia (ordenada2) do  
    agregar(ordenadafinal, extraer(ordenada2));  
  writefila(ordenadafinal);  
end.  
  
/*****/
```

```
Program e7p3;
{ordena la fila origen de forma ascendente dejando el resultado en
destino}
uses estructu;
var origen, destino:fila;

Procedure pasamayor;
{pasa el mayor elemento de origen a destino}
var aux1,aux2:pila;
begin
  inicpila(aux1, '');
  inicpila(aux2, '');
  if not filavacia (origen) then
    begin
      apilar(aux1, extraer(origen));
      while not filavacia (origen) do
        if (primero(origen) > tope(aux1)) then
          apilar(aux2, extraer(origen))
        else
          begin
            apilar(aux2, desapilar(aux1));
            apilar(aux1, extraer(origen));
          end;
        if not pilavacia(aux1) then
          agregar(destino, desapilar(aux1));
        while not pilavacia(aux2) do
          agregar(origen, desapilar(aux2));
        end;
      end;
    end;

begin
  inicfila(destino, '');
  readfila(origen);
  while not filavacia(origen) do
    pasamayor;
  writefila(destino);
end.

/*****/
```

```
Program e8p3;
{ordena la fila origen de forma ascendente dejando el resultado en
destino, por metodo de insercion}
uses estructu;
var origen:fila;
    aux1:pila;

Procedure ordenaux1;
{ordena la pila aux1 de forma creciente de tope a base}
var aux2:pila;
begin
    inicpila(aux2, '');
    if not filavacia (origen) then
        begin
            apilar(aux1, extraer(origen));
            while not filavacia (origen) do
                begin
                    if pilavacia(aux1) then
                        begin
                            apilar(aux1, extraer(origen));
                            while not pilavacia (aux2) do
                                apilar(aux1, desapilar(aux2));
                            end;
                        if (primero(origen) < tope(aux1)) then
                            begin
                                apilar(aux1, extraer(origen));
                                while not pilavacia(aux2) do
                                    apilar(aux1, desapilar(aux2));
                                end
                            else
                                apilar(aux2, desapilar(aux1));
                            end;
                        end;
                    end;
                end;
            end;
        end;
    begin
        readfila(origen);
        inicpila(aux1, '');
        ordenaux1;
        while not pilavacia(aux1) do
            agregar(origen, desapilar(aux1));
        writefila(origen);
    end.

/*****/
```

```
Program e9p3;
{dada la fila origen, separa la secuencia mas larga de elementos no
nulos,
en la fila secmaslarga}
uses estructu;
var origen, secmaslarga:fila;
    aux1, ceros, mayor:pila;
{.....}
Procedure sacaceros;
{saca los ceros de la fila origen}
begin
    while (not filavacia(origen)) and
        (primero(origen)= 0) do
        apilar(ceros, extraer(origen));
end;
{.....}
Procedure sacasegmentos;
{saca secuencias de numeros distintos de 0, de dada, y las envia a la
pila aux1}
begin
    while (not filavacia (origen)) and
        (primero(origen) <> 0) do
        apilar(aux1, extraer(origen));
end;
{.....}
procedure comparamayor;
{compara la pila mayor y aux1 dejando la secuencia mayor en la pila
mayor}
var auxmayor, aux2, descarte:pila;
begin
    inicpila(auxmayor, '');
    inicpila(aux2, '');
    inicpila(descarte, '');
    while (not pilavacia(mayor)) and
        (not pilavacia(aux1)) do
        begin
            apilar(auxmayor, desapilar(mayor));
            apilar(aux2, desapilar(aux1));
        end;
if pilavacia (aux1) then
    begin
        while not pilavacia(auxmayor) do
            apilar(mayor, desapilar(auxmayor));
        while not pilavacia(aux2) do
            apilar(descarte, desapilar(aux2));
            apilar(descarte, desapilar(ceros));
        end;
if pilavacia (mayor) then
    begin
        while not pilavacia(auxmayor) do
            apilar(descarte, desapilar(auxmayor));
```

```
    apilar(descarte, desapilar(ceros));
    while not pilavacia(aux2) do
        apilar(aux1, desapilar(aux2));
    while not pilavacia(aux1) do
        apilar(mayor, desapilar(aux1));
    end;
end;
{.....}
begin
    readfila(origen);
    inicfila(secmaslarga, '');
    inicpila(ceros, '');
    inicpila(mayor, '');
    inicpila(aux1, '');
    sacaceros;
    while (not filavacia(origen)) and
        (primero (origen) <> 0) do
        apilar(mayor, extraer(origen));
    while not filavacia(origen) do
        begin
            sacaceros;
            sacasegmentos;
            comparemayor;
        end;
    while not pilavacia (mayor) do
        agregar(secmaslarga, desapilar(mayor));
    writefila(secmaslarga);
end.

/*****/
```

```
program e10p3;  
{juego de cartas, se reparten 2 cartas por jugador, y gana el jugador  
cuya suma de  
cartas, sea mayor a las del contrincante}  
uses estructu;  
var  
    mazo, jug1, jug2, aux1, aux2, aux3, aux4, puntosjug1, puntosjug2:pila;  
{.....}  
Procedure repartemazo;  
{reparte la pila mazo en jug1 y jug2}  
begin  
    while not pilavacia (mazo) do  
        begin  
            apilar(jug1, desapilar(mazo));  
            if not pilavacia (mazo) then  
                apilar(jug2, desapilar(mazo));  
            end;  
        end;  
    {.....}  
Procedure reparte;  
{separa las cartas en 4 pilas para despues poder comparar topes}  
begin  
    if not pilavacia(jug1) then  
        apilar(aux1, desapilar(jug1));  
    if not pilavacia(jug1) then  
        apilar(aux2, desapilar(jug1));  
    if not pilavacia(jug2) then  
        apilar(aux3, desapilar(jug2));  
    if not pilavacia(jug2) then  
        apilar(aux4, desapilar(jug2));  
end;  
{.....}  
Procedure comparatopes;  
{suma los topes de aux1 y aux2, y los por separado los de aux3 y aux 4 y  
luego los compara}  
begin  
    if (not pilavacia(aux1)) and  
        (not pilavacia(aux2)) and  
        (not pilavacia(aux3)) and  
        (not pilavacia(aux4)) then  
        if (tope(aux1)+tope(aux2)) >= (tope(aux3)+tope(aux4)) then  
            begin  
                apilar(puntosjug1, desapilar(aux1));  
                apilar(puntosjug1, desapilar(aux2));  
                apilar(puntosjug1, desapilar(aux3));  
                apilar(puntosjug1, desapilar(aux4));  
            end  
        else  
            begin  
                apilar(puntosjug2, desapilar(aux1));  
                apilar(puntosjug2, desapilar(aux2));
```

```
        apilar(puntosjug2, desapilar(aux3));
        apilar(puntosjug2, desapilar(aux4));
    end;
end;
{.....}
begin
    inicpila(jug1, '');
    inicpila(jug2, '');
    inicpila(aux1, '');
    inicpila(aux2, '');
    inicpila(aux3, '');
    inicpila(aux4, '');
    inicpila(puntosjug1, '');
    inicpila(puntosjug2, '');
    readpila(mazo);
    repartemazo;
    while (not pilavacia(jug1)) and
        (not pilavacia(jug2)) do
        begin
            reparte;
            comparatopes;
        end;
    writepila(puntosjug1);
    writepila(puntosjug2);
end.

/*****/
```

```
program ellp3;
{este programa separa de la pila dada el menor de sus elementos
y deja los restantes elementos en el mismo orden}
uses estructu;
var dada, copiadata, auxdata, menor, menorcopia:pila;
{.....}
procedure copiadata;
{este procedimiento crea una copia de dada}
begin
  while not pilavacia(dada) do
    begin
      apilar(copiadata, (tope(dada)));
      apilar(auxdata, desapilar(dada));
    end;
    while not pilavacia(auxdata) do
      apilar(dada, desapilar(auxdata));
    end;
  {.....}
procedure sacamenor;
{este procedimiento saca el menor de la pila 'copiadata'}
var auxcopia:pila;
begin
  inicpila(auxcopia, '');
  if not pilavacia(copiadata) then
    begin
      apilar(menorcopia, desapilar(copiadata));
      while not pilavacia(copiadata) do
        if tope(copiadata) < (tope(menorcopia)) then
          begin
            apilar(auxcopia, desapilar(menorcopia));
            apilar(menorcopia, desapilar(copiadata));
          end else
            apilar(auxcopia, desapilar(copiadata));
        end;
      end;
    end;
  {.....}
procedure sacamenordada;
{saca de dada, los elementos iguales al tope de menorcopia}
begin
  while not pilavacia(dada) do
    if (tope(dada)=tope(menorcopia)) then
      apilar(menor, desapilar(dada))
    else
      apilar(auxdata, desapilar(dada));
    while not pilavacia(auxdata) do
      apilar(dada, desapilar(auxdata));
    end;
  {.....}
begin
  inicpila(copiadata, '');
  readpila(dada);
```

```
inicpila(menor, '');  
inicpila(auxdada, '');  
inicpila(menorcopia, '');  
copiadadada;  
sacamenor;  
sacamenordada;  
write('pila dada: ');  
writepila(dada);  
write('pila menor: ');  
writepila(menor);  
end.
```

```
/*  
*****  
*/
```

```
Program E12P3;  
{invierte el orden de una fila usando solo filas auxiliares}  
uses estructu;  
var origen, invertida, aux:fila;
```

```
Procedure Invertir;  
{invierte la fila origen, y la deja en la fila invertida}  
begin
```

```
  while (not filavacia (origen)) do  
    begin  
      agregar(aux, extraer(origen));  
      while (not filavacia(invertida)) do  
        agregar(aux, extraer(invertida));  
      while (not filavacia(aux)) do  
        agregar(invertida, extraer(aux));  
    end;  
end;
```

```
begin  
  readfila(origen);  
  inicfila(invertida, '');  
  inicfila(aux, '');  
  if (not filavacia (origen)) then  
    agregar(invertida, extraer(origen));  
  Invertir;  
  while (not filavacia(invertida)) do  
    agregar(origen, extraer(invertida));  
  writefila(origen);  
end.
```