

```
program envio;

  type CanEnt = channel of integer;
  var c1 : CanEnt;

process P ;
  var elem : integer ;
begin
  repeat
    c1 ? elem;
    writeln('Recibido', elem)
  forever
end ;

process Q ;
begin
  repeat
    c1 ! random(5);
  forever
end ;

begin
  cobegin
    P; Q
  coend
end.
```

```
program ejemplo;

  var c: channel of integer;

  process P ;
    var i,v : integer;
  begin
    for i:=1 to 3 do begin
      c ! i;
      c ? v;
      write('P', v:1)
    end
  end ;

  process Q ;
    var i,v : integer;
  begin
    for i:=1 to 3 do begin
      c ? v;
      write('Q', v:1);
      c ! v
    end
  end ;

begin
  cobegin
    P; Q
  coend
end.
```

```
program primos;

const N = 10;

type chan = channel of integer ;
       chans = array [1..N] of chan ;
var   tuberia : array[ 0 .. N] of chan ;
       output : chans ;
       i :integer;

process generador(var out : chan);    (* Genera números consecutivos *)
  var i : integer ;
begin
  i := 2 ;          (* El 1 no se tiene en cuenta *)
  repeat
    out ! i ;
    i := i + 1
  forever
end ;

process consumidor(var in : chan) ; (*Consume los enteros que pasan la criba*)
var   i :integer ;
begin
  repeat
    in ? i
  forever
end ;

process salida(var sals: chans);
  var i, num: integer;
begin
  for i:= 1 to N do begin
    sals[i] ? num;
    writeln(num)
  end
end;
```

```
process type filtro_t(var izqda, dcha, abajo: chan);  
  var p,q: integer;  
begin  
  izqda ? p;  
  abajo ! p;  
  repeat  
    izqda ? q;  
    if (q mod p  $\neq$  0) then dcha ! q  
  forever  
end;  
  
var filtro: array [1..N] of filtro_t;  
  
begin  
  cobegin  
    generador(tuberia[0]);  
    for i:= 1 to N do  
      filtro[i](tuberia[i-1], tuberia[i], salida[i]);  
    coend  
end.
```

```
program barreras;

const N = 10;

type sinc: channel of synchronous;
      Tbarrera: array [1.. N] of sinc;

var barrera : Tbarrera;

process master (var barr: Tbarrera);
  var i: integer;
begin
  repeat
    .....
    (* Espera a que todos hayan acabado esta iteración *)
    for i:= 1 to N do
      barr[i] ? any;
    (* Da paso libre a todos a la vez *)
    for i:= 1 to N do
      barr[i] ! any;
    .....
  forever
end

process type worker_t (var mibarr: sinc);
begin
  repeat
    .....
    (* Aviso de fin de mi bucle *)
    mibarr ! any;
    (* Espera via libre *)
    mibar ? any;
    .....
  forever
end;

var worker: array [1..N] of worker_t;
  i: integer;
begin
  cobegin
    master(barrera);
    for i:= 1 to N do
      worker (barrera[i])
    coend
end.
```

```
program forofosmsg1;

const NPUERTAS = 2;
      N = 20; (* número esperado de espectadores por cada puerta *)

type canal: channel of integer;

var   c: array [1..NPUERTAS] of canal;

process type forofos (id: integer);
      var i: integer;
begin
      for i := 1 to N do c[id] ! 1
end;

process contador;
      var   total, i, j, v : integer;
begin
      total := 0;
      for i:= 1 to N do
        for j:= 1 to NPUERTAS do begin
          c[j] ? v;
          total := total + v
        end;
      writeln('Valor final de total: ', total)
end;

var   i:integer;
      puertas: array [1..NPUERTAS] of forofos;

begin
  cobegin
    contador;
    for i := 1 to NPUERTAS do
      puertas [i](i)
    coend
end.
```

```
program buff1;

const N = 32; (* tamaño del buffer *)

type canal = channel of integer;

process buffer(var cin, cout: canal);

var  buff: array [1 .. N] of integer;
      pEsc, pLec: integer;

begin
  pEsc := 0;
  pLec := 0;
  repeat
    select
      cin ? buff[pEsc];
      pEsc := pEsc mod N + 1;
    or
      cout ! buff[pLec];
      pLec := pLec mod N + 1
    end
  forever
end.
```

```
program buff2;

const N = 32; (* tamaño del buffer *)

type canal = channel of integer;

process buffer(var cin, cout: canal);

    var buff: array [1 .. N] of integer;
        pEsc, pLe, numelems: integer;

begin
    pEsc := 0;
    pLec := 0;
    numelems := 0;
    repeat
        select
            when numelems < N =>
                cin ? buff[pEsc];
                pEsc := pEsc mod N + 1;
                numelems := numelems + 1;
            or
            when numelems > 0 =>
                cout ! buff[pLec];
                pLec := pLec mod N + 1;
                numelems := numelems - 1
        end
    forever
end.
```



```
program buff3;

const N = 32; (* tamaño del buffer *)

type canal = channel of integer;

process buffer(var cin, cout: canal);

    var buff: array [1 .. N] of integer;
        pEsc, pLe, numelems: integer;

begin
    pEsc := 0;
    pLec := 0;
    numelems := 0;
    repeat
        select
            when numelems < N =>
                cin ? buff[pEsc];
                pEsc := pEsc mod N + 1;
                numelems := numelems + 1;
            or
            when numelems > 0 =>
                cout ! buff[pLec];
                pLec := pLec mod N + 1;
                numelems := numelems - 1
            or
            terminate
        end
    forever
end.
```

```
program forofosmsg2;

const NPUERTAS = 2;
      N = 20; (* número esperado de espectadores por cada puerta *)

type canal: channel of integer;
      sinc: channel of synchronous;

var c: array [1..NPUERTAS] of canal;
     final: array [1..NPUERTAS] of sinc;

process type forofos (id: integer);
  var i: integer;
begin
  for i := 1 to N do c[id] ! 1;
  final[id] ! any
end;

process contador;
  var total, i, j, v : integer;
      continua: array[1..2] of boolean;
begin
  total := 0;
  continua[1] := true;
  continua[2] := true;
  while continua[1] or continua[2] do
    select
      c[1] ? v; total := total + v
    or
      c[2] ? v; total := total + v
    or
      final[1] ? any; continua[1] := false
    or
      final[2] ? any; continua[2] := false
    end;
  writeln('Valor final de total: ', total)
end;

var i:integer; puertas: array [1..NPUERTAS] of forofos;
begin
  cobegin
    contador;
    for i := 1 to NPUERTAS do puertas [i](i)
  coend
end.
```

```
program forofosmsg3;

const NPUERTAS = 2;
      N = 20; (* número esperado de espectadores por cada puerta *)

type canal: channel of integer;

var   c: array [1..NPUERTAS] of canal;

process type forofos (id: integer);
      var i: integer;
begin
      for i := 1 to N do c[id] ! 1;
end;

process contador;
      var   total, i, j, v : integer;
begin
      total := 0;
      repeat
        select
          c[1] ? v;
        or
          c[2] ? v;
        or
          terminate
        end;
      total := total + v
      forever
        writeln('Valor final de total: ', total)
end;

var   i:integer; puertas: array [1..NPUERTAS] of forofos;

begin
  cobegin
    contador;
    for i := 1 to NPUERTAS do puertas [i](i)
  coend
end.
```

```
program forofosmsg4;

const NPUERTAS = 2;
      N = 20; (* número esperado de espectadores por cada puerta *)

type canal: channel of integer;

var   c: array [1..NPUERTAS] of canal;

process type forofos (id: integer);
      var i: integer;
begin
      for i := 1 to N do c[id] ! 1;
end;

process contador (var total:integer);
      var   total, i, j, v : integer;
begin
      total := 0;
      repeat
        select
          c[1] ? v; total := total + v
        or
          c[2] ? v; total := total + v
        or
          terminate
        end;
      forever
end;

var   i,suma : integer;
      puertas: array [1..NPUERTAS] of forofos;

begin
      cobegin
        contador (suma);
        for i := 1 to NPUERTAS do puertas [i](i)
      coend;
      writeln('Valor final de total: ', suma)
end.
```

```
program primos2;
...

type cansinc = channel of synchronous;

process generador(var out : chan; var fincan: cansinc);
var i : integer;
    finbucle: boolean
begin
    i := 2 ;
    finbucle := false;
    while not finbucle do
        select
            cansinc ? any;
            finbucle := true;
        or
            out ! i ;
            i := i + 1
        end
    end;

process salida(var sals: chans);
    var i, num: integer;
begin
    for i:= 1 to N do begin
        sals[i] ? num;
        writeln(num)
    end;
    cansinc ! any
end;
```

```
program temporizador;  
  
type canal: channel of integer;  
  
var coms: canal;  
  
process dormilon(var c: canal);  
    var i: integer;  
begin  
    for i := 1 to 10 do begin  
        sleep(random(5));  
        c ! 1  
    end  
end;  
  
process vigia(var c: canal);  
    var rec,temp : integer;  
begin  
    rec := 0;  
    while rec < 10 do  
        select  
            c ? temp;  
            rec := rec + 1;  
            writeln('Recibido el ',temp);  
        or  
            timeout 5;  
            writeln('Tiempo excedido')  
        end  
    end;  
  
begin  
    cobegin  
        dormilon(coms); vigia(coms)  
    coend  
end.
```

```
Kchan ? k;  
repeat  
  select  
    Kchan ? K;  
  or  
    inp ? V;  
    if (V<= K) and (V >= -K) then  
      out ! V  
    end  
forever
```

```
Kchan ? k;  
repeat  
  select  
    Kchan ? K;  
  else  
    null;  
  end;  
  select  
    Kchan ? K;  
  or  
    inp ? V;  
    if (V<= K) and (V >= -K) then  
      out ! V  
    end  
forever
```

```
Kchan ? k;  
repeat  
  pri select  
    Kchan ? K;  
  or  
    inp ? V;  
    if (V<= K) and (V >= -K) then  
      out ! V  
    end  
forever
```

```
program filosofos;

const NumFil = 5;
type cansinc = channel of synchronous;
var cogerizq, cogerder, soltarizq, soltarder: array [1..Numfil] of cansinc;

process type tenedor_t(id: integer);
begin
    repeat
        select
            cogerizq[id] ? any;
            soltarizq[id] ? any;
        or
            cogerder[id] ? any;
            soltarder[id] ? any;
        or
            terminate
        end
    forever
end;

process type filosofo_t (id:integer);
    var i, tenedor1, tenedor2: integer;
begin
    if id mod 2 = 0 then begin
        tenedor1 = (id mod NumFil) + 1; tenedor2 := id
    else begin
        tenedor1 = id ; tenedor2 := (id mod NumFil) + 1
    end;
    repeat
        sleep(random(5)); (* Pensar *)
        cogerizq[tenedor1] ! any;
        cogerder[tenedor2] ! any;
        sleep(random(5)); (* Comer *)
        soltarizq[tenedor1] ! any;
        soltarder[tenedor2] ! any;
    forever
end

var filosofo: array [1..Numfil]of filosofo_t;
    tenedor: array [1.. NumFil] of tenedor_t;
    i: integer;
begin
    cobegin for i:= 1 to NumFil do begin tenedor[i](i); filosofo[i](i) end coend
end.
```