Processus concurrents et parallélisme

Chapitre 3 - Synchronisation

Gabriel Girard

17 octobre 2022

Chapitre 3 - Synchronisation

Chapitre 3 - Synchronisation

```
type   item = ...
var    tampon : array[0..n-1] of item;
       in, out : 0..n;
       nextp, nextc : item;

in := 0; out := 0;
parbegin
```

```
//      Producteur

while (true)
   ...
   produire un item
   ...
   while((in+1 %n)=out);
   tampon[in] := nextp;
   in := in + 1 mod n;
```

```
//      Consommateur

While (true)
    while(in=out) ;
    nextc := tampon[out]
    out := out+1 mod n;
    ...
    traiter un item
    ...
```

```
parend
```

```
type   item = ...
var    tampon : array[0..n-1] of item;
       in, out : 0..n;
       nextp, nextc : item;

in := 0; out := 0;
parbegin
```

// Producteur	// Consommateur

while (true)
  ...
  produire un item
  ...
  while((in+1 %n)=out);
  tampon[in] := nextp;
  in := in + 1 mod n;
	While (true)
	  while(in=out) ;
	  nextc := tampon[out]
	  out := out+1 mod n;
	  ...
	  traiter un item
	  ...

```
parend
```

```
type   item = ...
var    tampon : array[0..n-1] of item;
       in, out : 0..n;
       nextp, nextc : item;

in := 0; out := 0;
parbegin
```

```
//      Producteur

while (true)
   ...
   produire un item
   ...
   while((in+1 %n)=out);
   tampon[in] := nextp;
   in := in + 1 mod n;
```

```
//      Consommateur

While (true)
    while(in=out) ;
    nextc := tampon[out]
    out := out+1 mod n;
    ...
    traiter un item
    ...
```

```
parend
```

```
type   item = ...
var    tampon : array[0..n-1] of item;
       in, out : 0..n;
       nextp, nextc : item;

in := 0; out := 0;
parbegin
```

// Producteur	// Consommateur

while (true)
  ...
  produire un item
  ...
  while((in+1 %n)=out);
  tampon[in] := nextp;
  in := in + 1 mod n;
	While (true)
	  while(in=out) ;
	  nextc := tampon[out]
	  out := out+1 mod n;
	  ...
	  traiter un item
	  ...

```
parend
```

```
type   item = ...
var    tampon : array[0..n-1] of item;
       in, out : 0..n;
       nextp, nextc : item;

in := 0; out := 0;
parbegin
```

// Producteur	// Consommateur

while (true)
  ...
  produire un item
  ...
  while((in+1 %n)=out);
  tampon[in] := nextp;
  in := in + 1 mod n;
	While (true)
	  while(in=out) ;
	  nextc := tampon[out]
	  out := out+1 mod n;
	  ...
	  traiter un item
	  ...

```
parend
```

```
type   item = ...
var    tampon : array[0..n-1] of item;
       in, out : 0..n;
       nextp, nextc : item;

in := 0; out := 0;
parbegin
```

```
//      Producteur

while (true)
   ...
   produire un item
   ...
   while((in+1 %n)=out);
   tampon[in] := nextp;
   in := in + 1 mod n;
```

```
//      Consommateur

While (true)
    while(in=out) ;
    nextc := tampon[out]
    out := out+1 mod n;
    ...
    traiter un item
    ...
```

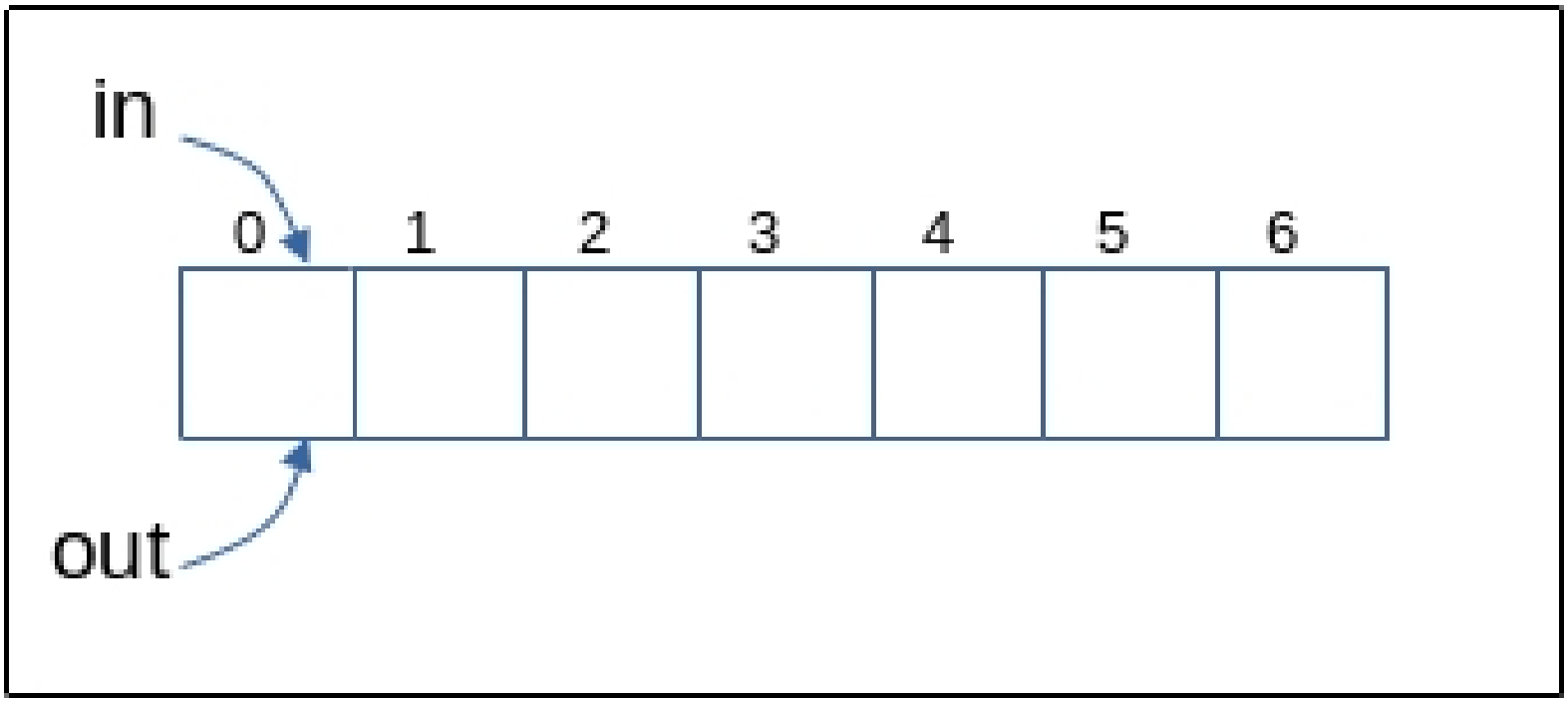
```
parend
```

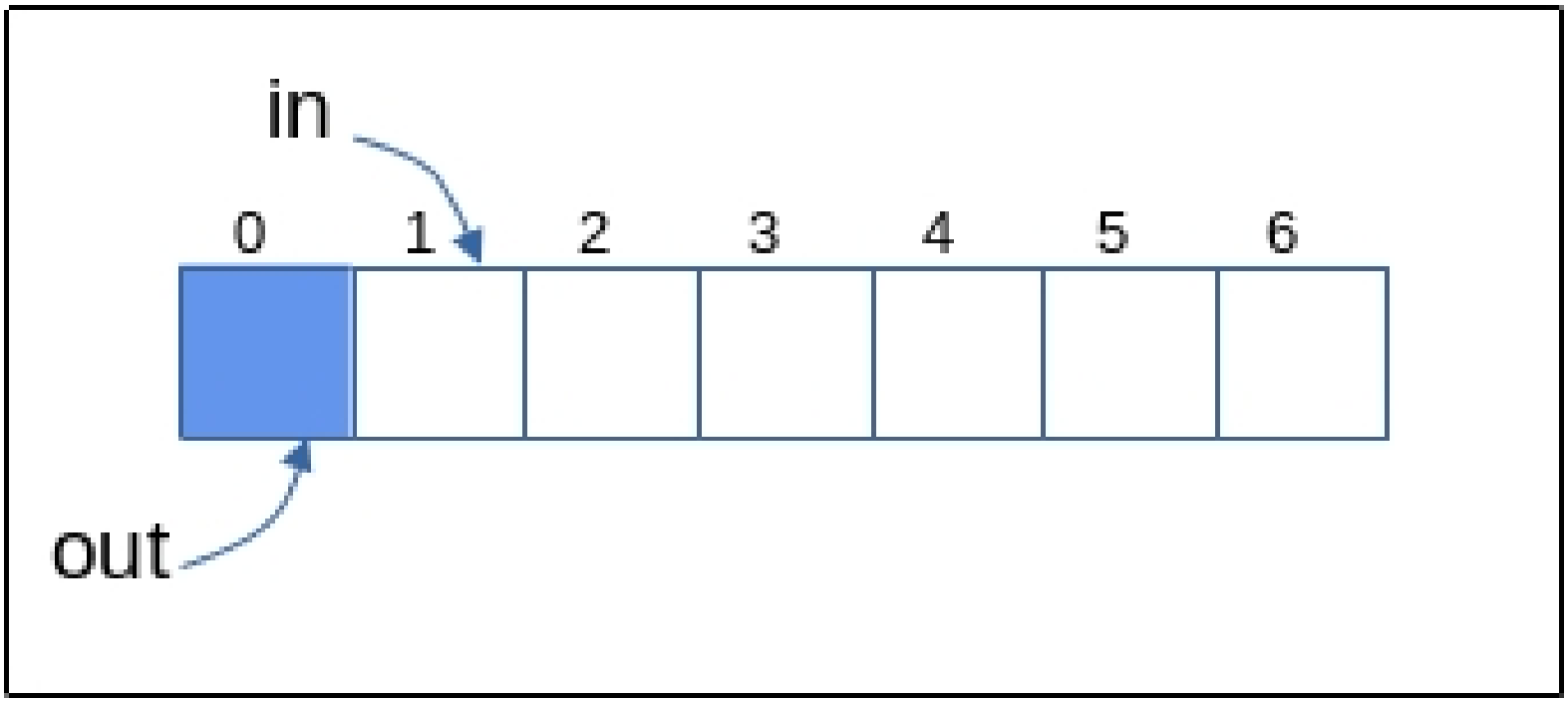
```
type   item = ...
var    tampon : array[0..n-1] of item;
       in, out : 0..n;
       nextp, nextc : item;

in := 0; out := 0;
parbegin
```

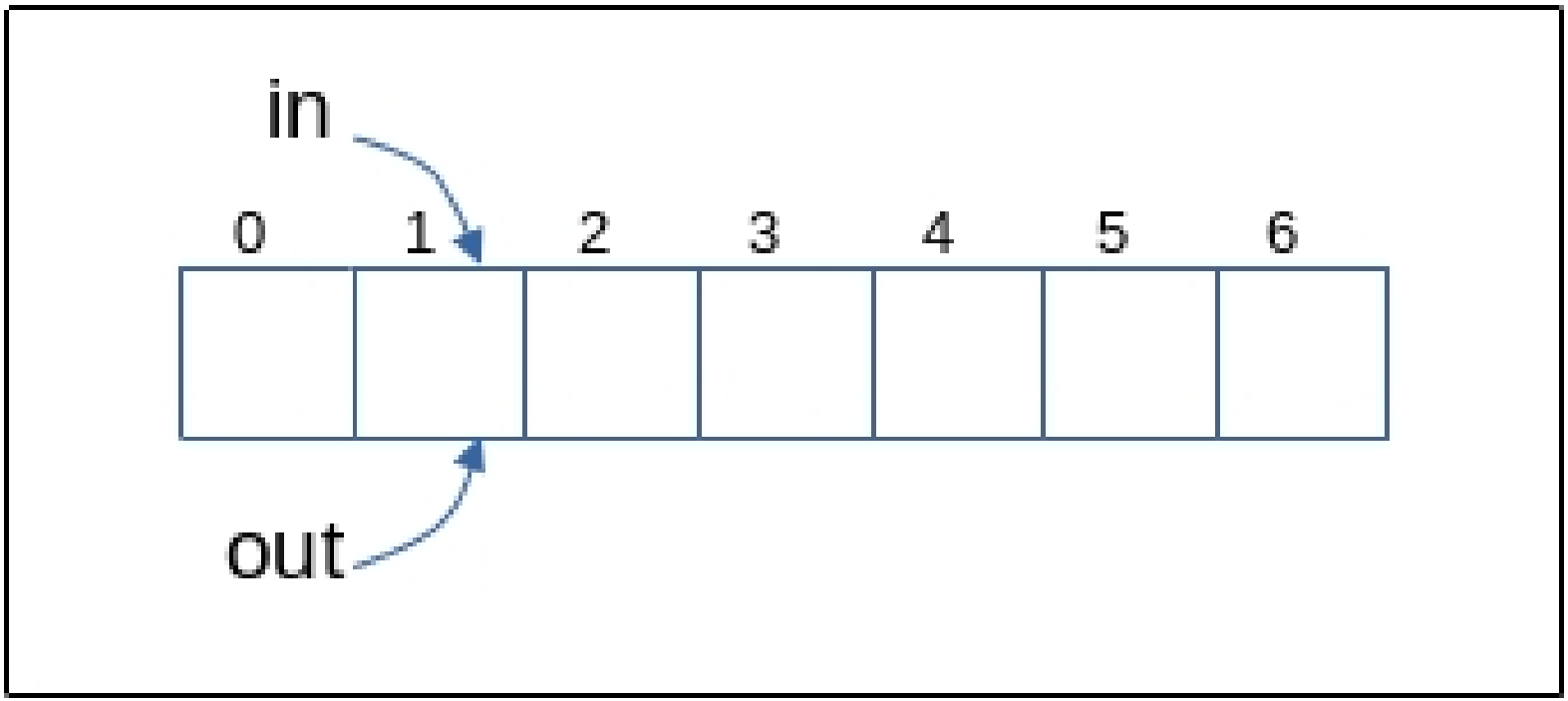
// Producteur	// Consommateur

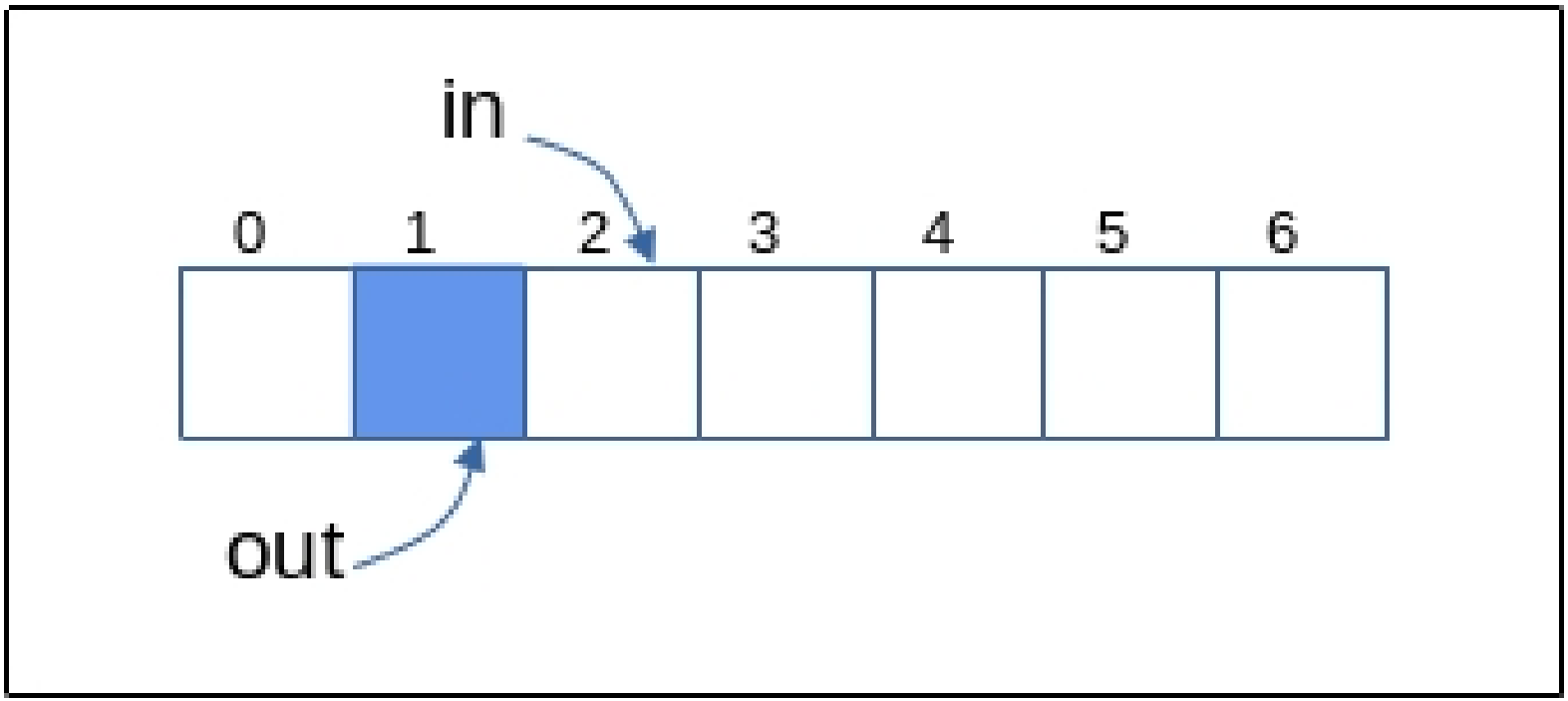
while (true)
  ...
  produire un item
  ...
  while((in+1 %n)=out);
  tampon[in] := nextp;
  in := in + 1 mod n;
	While (true)
	  while(in=out) ;
	  nextc := tampon[out]
	  out := out+1 mod n;
	  ...
	  traiter un item
	  ...

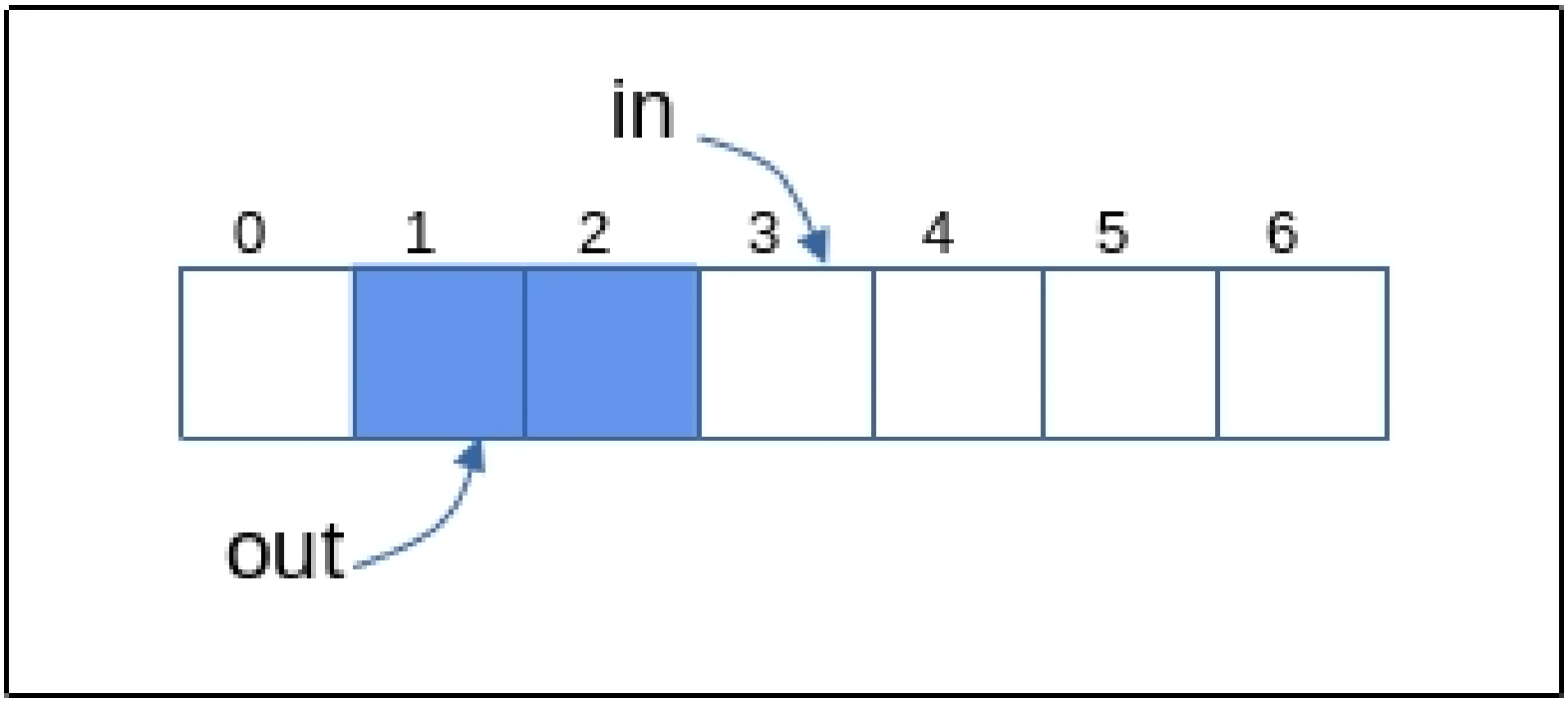
```
parend
```

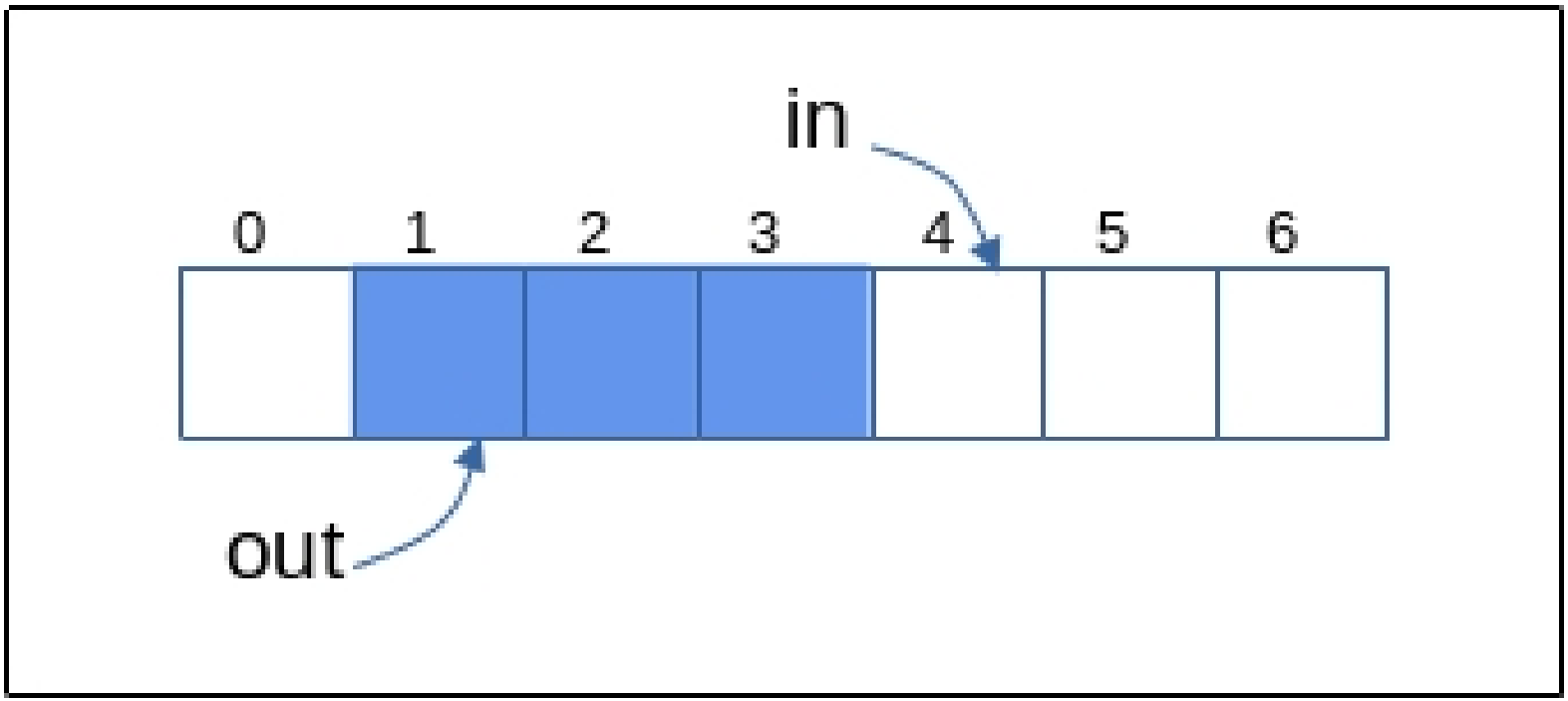

in
0
1
2
3
4
5
6
out

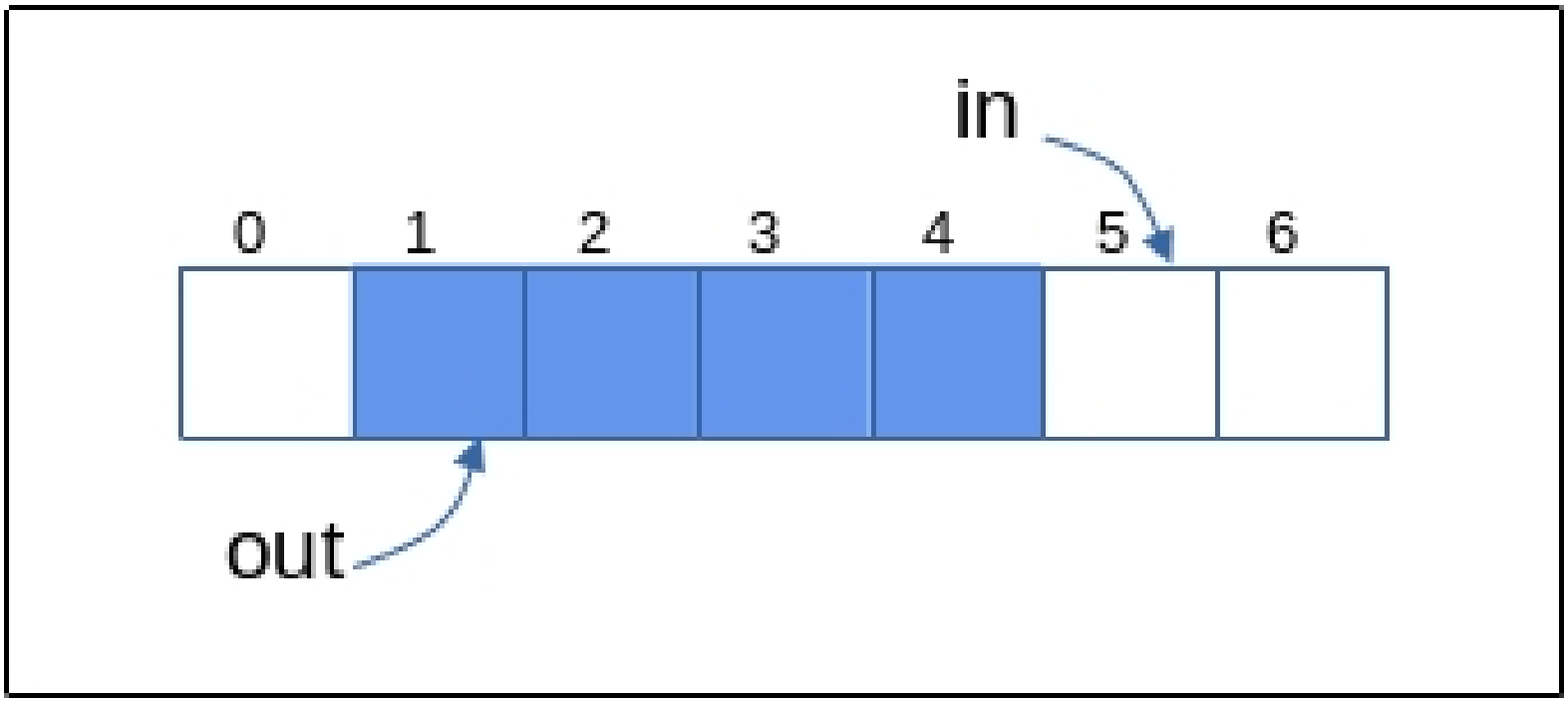

in
0
1
2
3
4
5
6
out

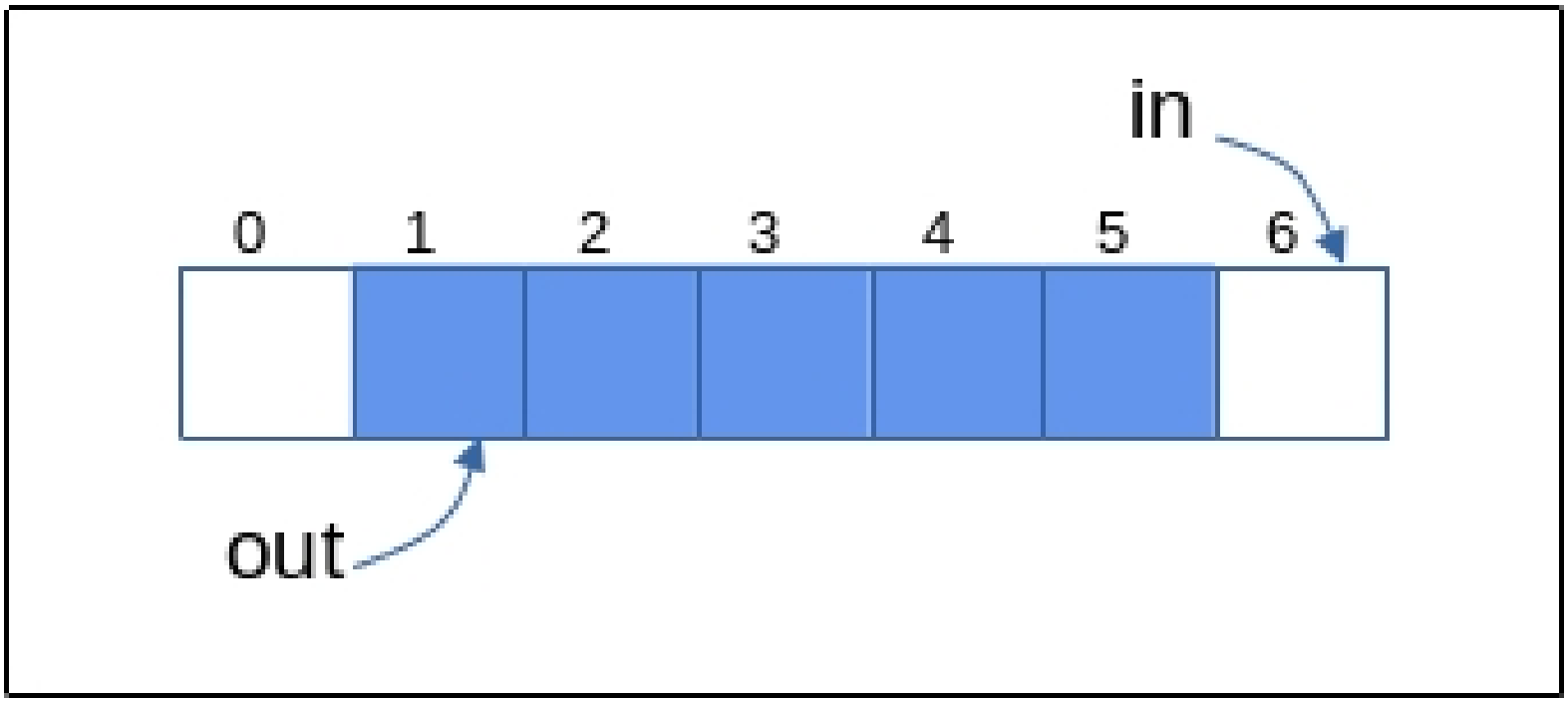

in
0
1
2
3
4
5
6
out

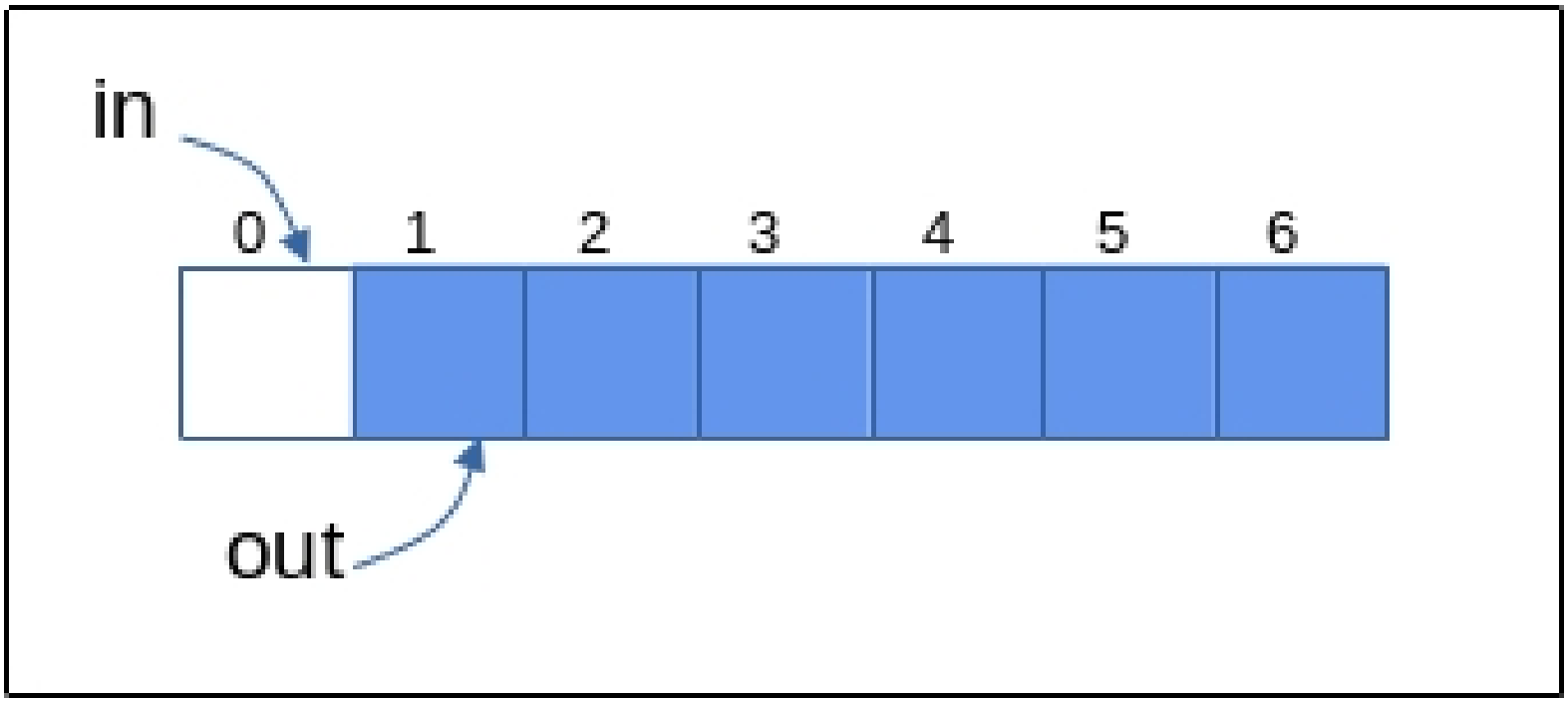

in
0
1
2
3
4
5
6
out


in
0
1
2
3
4
5
6
out


in
0
1
2
3
4
5
6
out


in
0
1
2
3
4
5
6
out


in
0
1
2
3
4
5
6
out


in
0
1
2
3
4
5
6
out

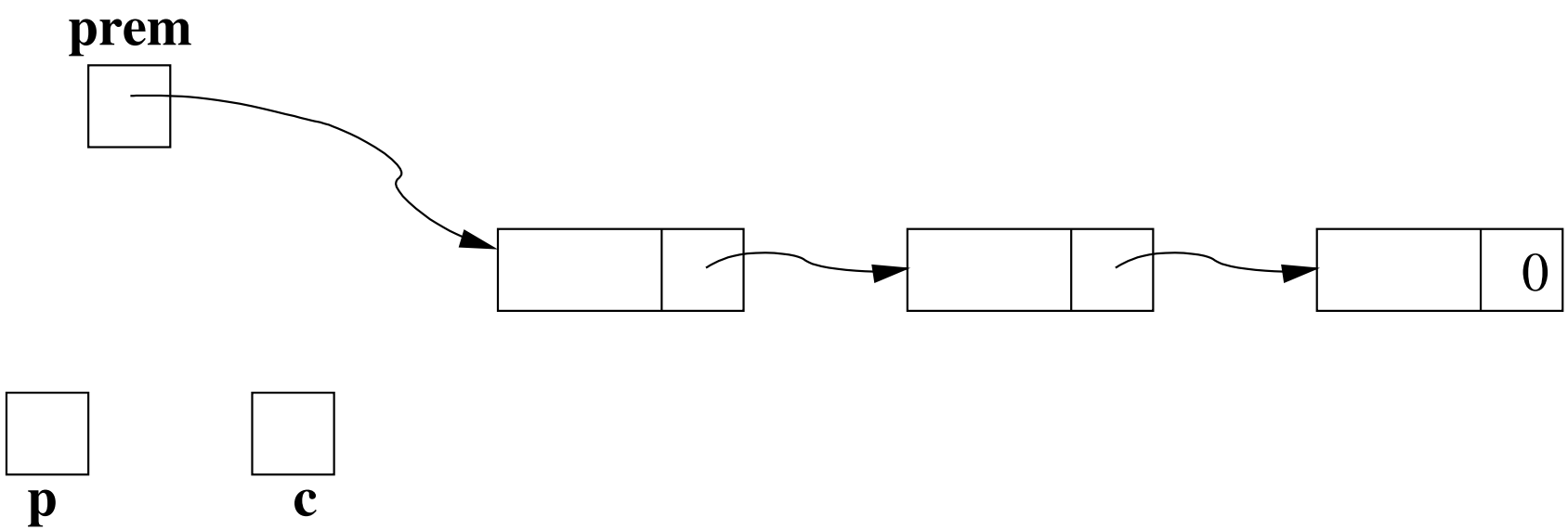
Chapitre 3 - Synchronisation

```
type    item = ...
type    tampon : record     element : item ;
                            suiv : pointer to tampon ;
                 end ;
var     prem, p, c : pointer to tampon ;
nextp, nextc : item ;
prem := nil ;
parbegin
```

```
//      Producteur

while (true)
      ...
      produire item nextp
      ...
      new(p);              // 2
      p.elem := nextp;// 3
      p.suiv := prem; // 4
      prem := p;           // 5
```

```
//      Consommateur

while (true)
      while (prem=nil);
      c := prem;           // 1
      prem:=prem.suiv;// 6
      nextc := c.elem;// 7
      dispose(c);          // 8
      ...
      traiter item nextc
      ...
```

```
parend
```

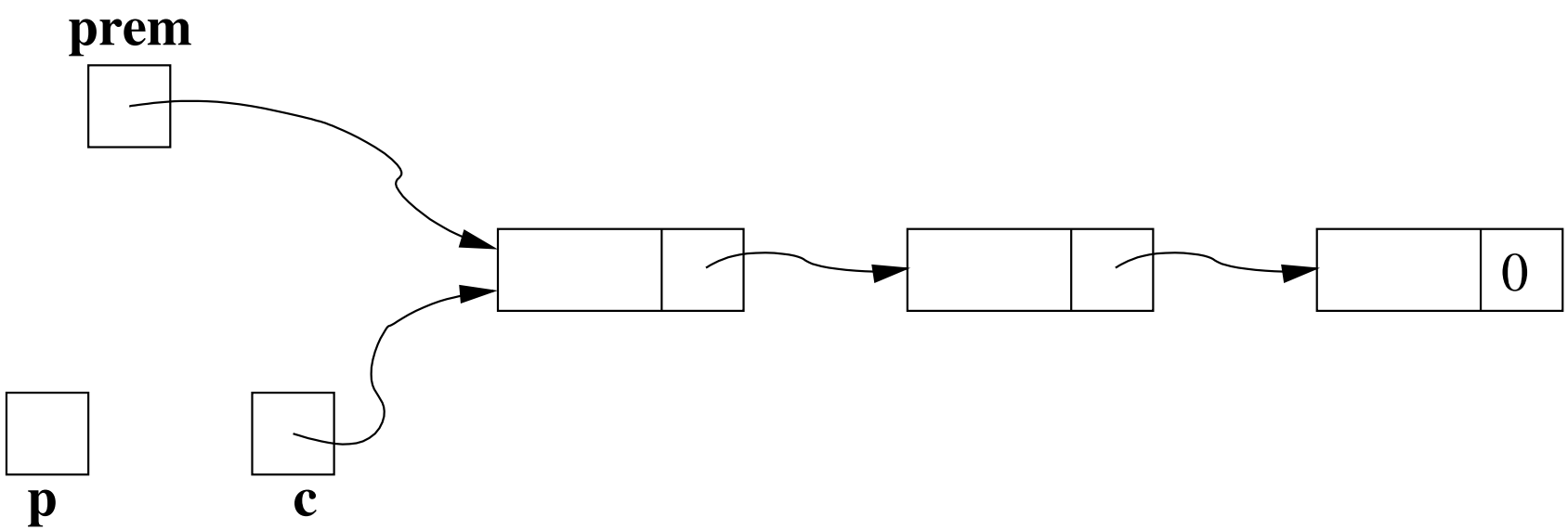

prem
0
p
c

```
type    item = ...
type    tampon : record    element : item ;
                           suiv : pointer to tampon ;
                 end ;
var     prem, p, c : pointer to tampon ;
nextp, nextc : item ;
prem := nil ;
parbegin
```

```
//      Producteur

while (true)
      ...
      produire item nextp
      ...
      new(p);              // 2
      p.elem := nextp;// 3
      p.suiv := prem; // 4
      prem := p;           // 5
```

```
//      Consommateur

while (true)
      while (prem=nil);
      c := prem;           // 1
      prem:=prem.suiv;// 6
      nextc := c.elem;// 7
      dispose(c);          // 8
      ...
      traiter item nextc
      ...
```

```
parend
```

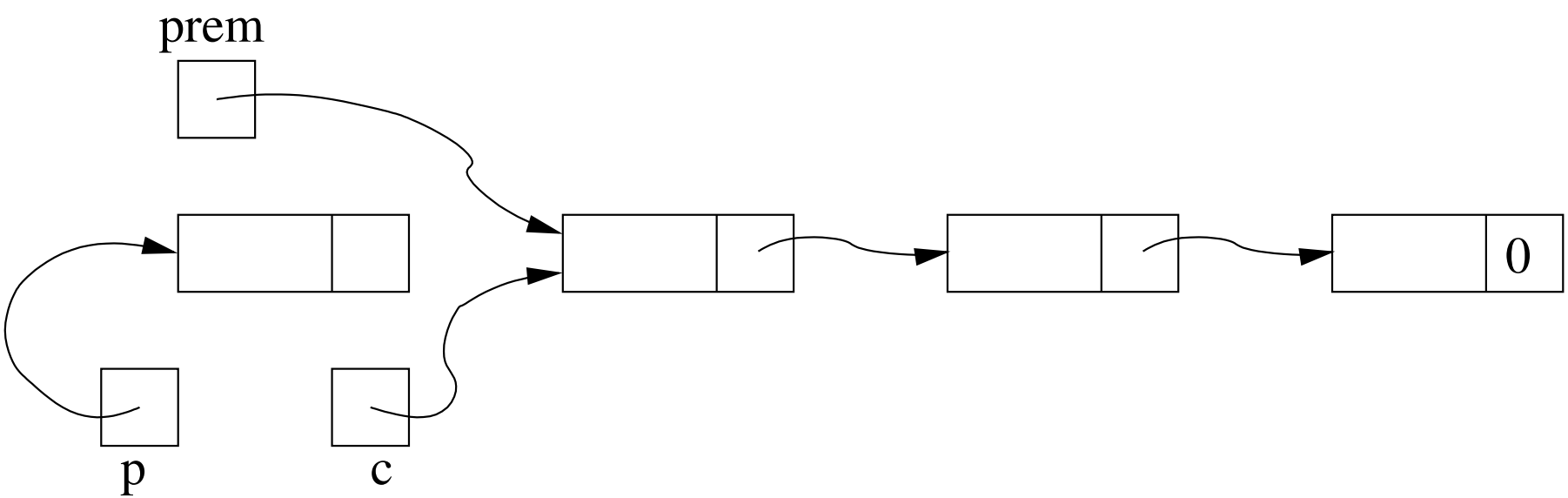

prem
0
p
c

```
type    item = ...
type    tampon : record     element : item ;
                            suiv : pointer to tampon ;
                 end ;
var     prem, p, c : pointer to tampon ;
nextp, nextc : item ;
prem := nil ;
parbegin
```

```
//      Producteur

while (true)
      ...
      produire item nextp
      ...
      new(p);              // 2
      p.elem := nextp;// 3
      p.suiv := prem; // 4
      prem := p;           // 5
```

```
//      Consommateur

while (true)
      while (prem=nil);
      c := prem;           // 1
      prem:=prem.suiv;// 6
      nextc := c.elem;// 7
      dispose(c);          // 8
      ...
      traiter item nextc
      ...
```

```
parend
```

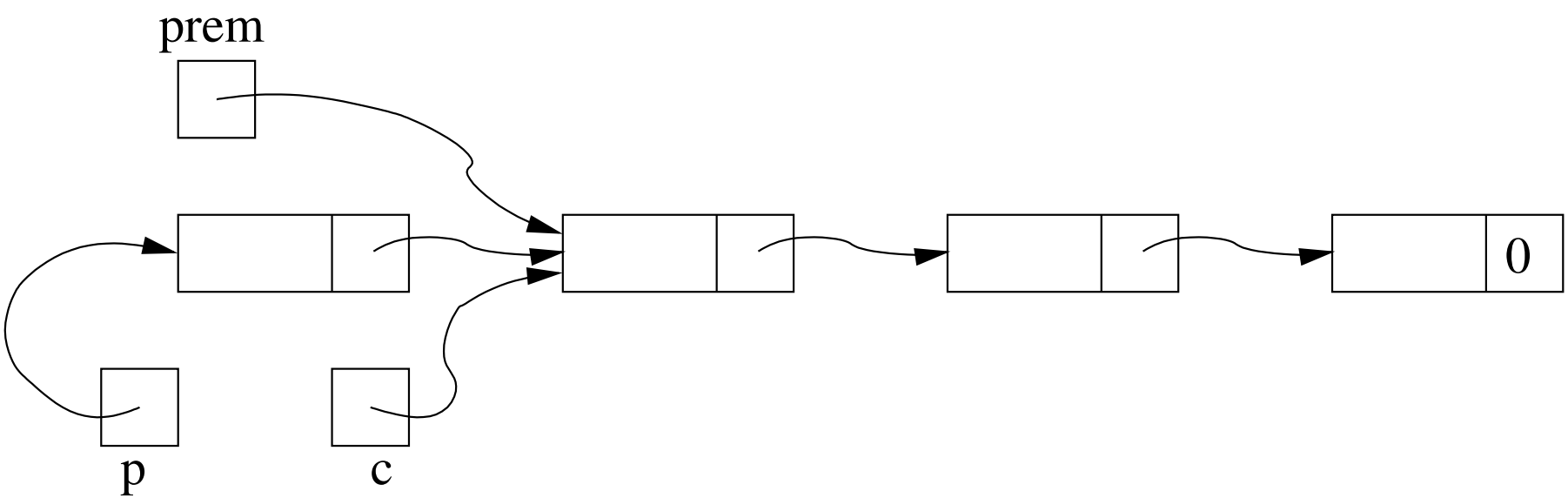

prem
0
p
c

```
type    item = ...
type    tampon : record    element : item ;
                           suiv : pointer to tampon ;
                 end ;
var     prem, p, c : pointer to tampon ;
nextp, nextc : item ;
prem := nil ;
parbegin
```

```
//      Producteur

while (true)
      ...
      produire item nextp
      ...
      new(p);              // 2
      p.elem := nextp;// 3
      p.suiv := prem; // 4
      prem := p;           // 5
```

```
//      Consommateur

while (true)
      while (prem=nil);
      c := prem;           // 1
      prem:=prem.suiv;// 6
      nextc := c.elem;// 7
      dispose(c);          // 8
      ...
      traiter item nextc
      ...
```

```
parend
```

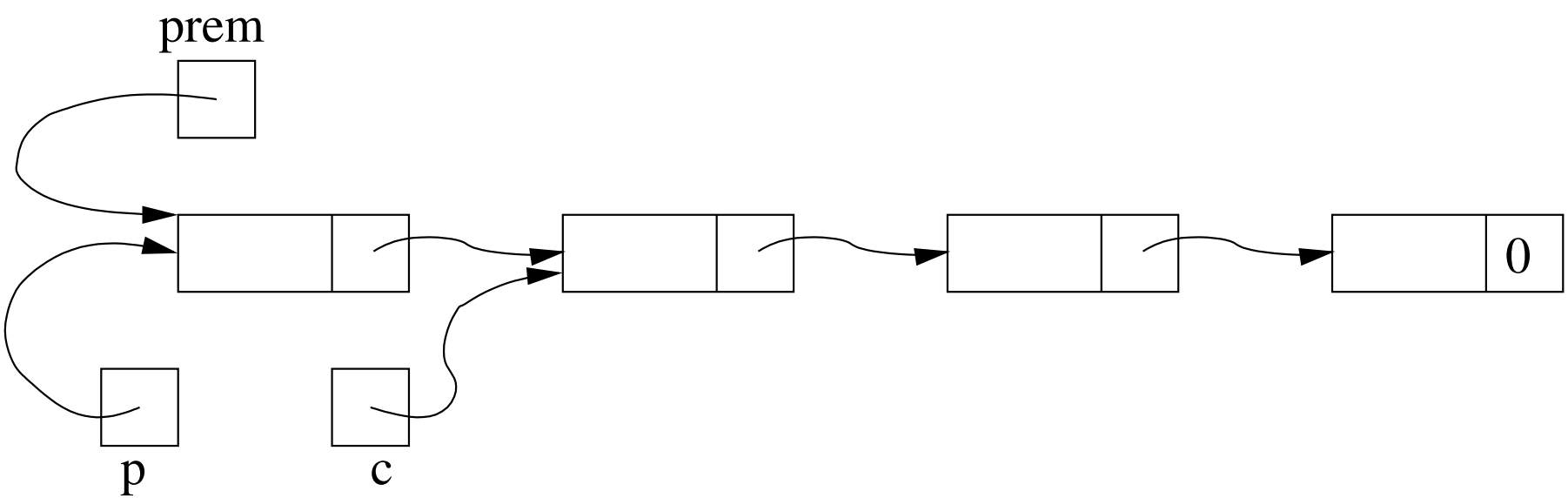

prem
0
p
c

```
type    item = ...
type    tampon : record    element : item ;
                           suiv : pointer to tampon ;
                 end ;
var     prem, p, c : pointer to tampon ;
nextp, nextc : item ;
prem := nil ;
parbegin
```

```
//      Producteur

while (true)
      ...
      produire item nextp
      ...
      new(p);              // 2
      p.elem := nextp;// 3
      p.suiv := prem; // 4
      prem := p;           // 5
```

```
//      Consommateur

while (true)
      while (prem=nil);
      c := prem;           // 1
      prem:=prem.suiv;// 6
      nextc := c.elem;// 7
      dispose(c);          // 8
      ...
      traiter item nextc
      ...
```

```
parend
```

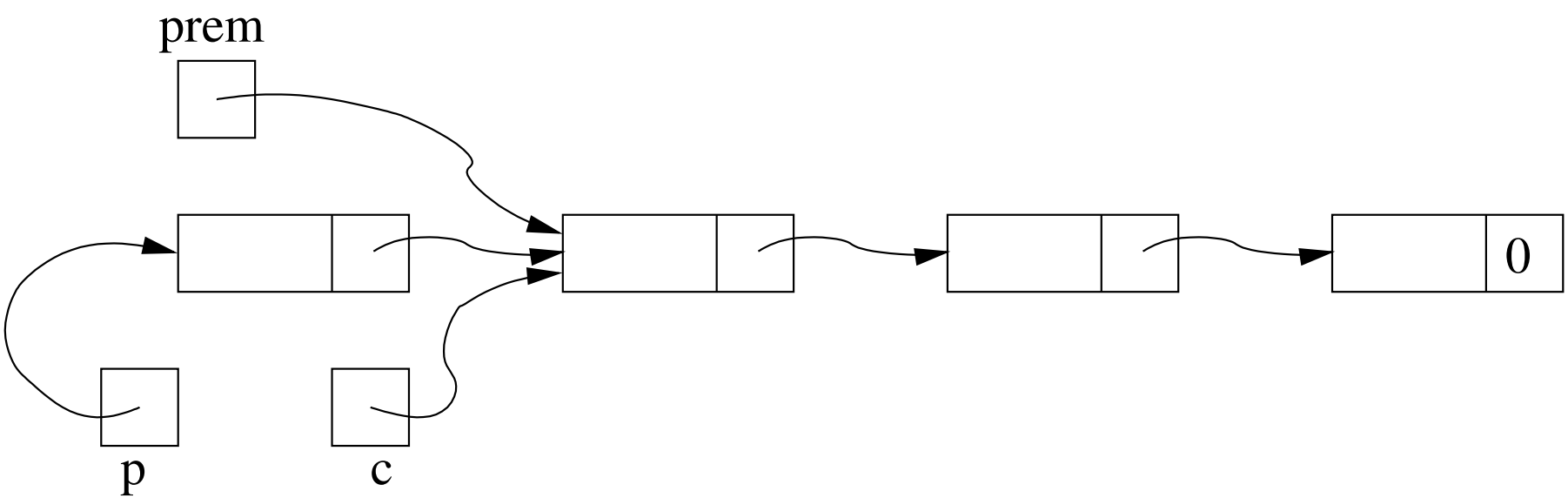

prem
0
p
c

```
type    item = ...
type    tampon : record     element : item ;
                            suiv : pointer to tampon ;
                 end ;
var     prem, p, c : pointer to tampon ;
nextp, nextc : item ;
prem := nil ;
parbegin
```

```
//      Producteur

while (true)
      ...
      produire item nextp
      ...
      new(p);              // 2
      p.elem := nextp;// 3
      p.suiv := prem; // 4
      prem := p;           // 5
```

```
//      Consommateur

while (true)
      while (prem=nil);
      c := prem;           // 1
      prem:=prem.suiv;// 6
      nextc := c.elem;// 7
      dispose(c);          // 8
      ...
      traiter item nextc
      ...
```

```
parend
```

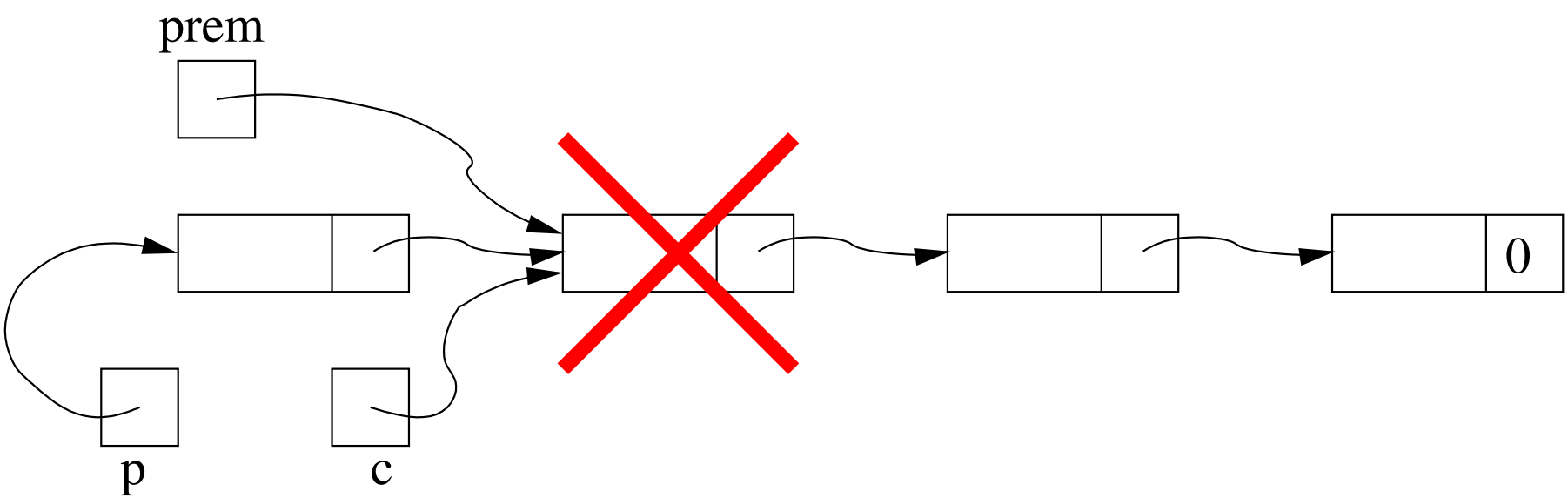

prem
0
p
c

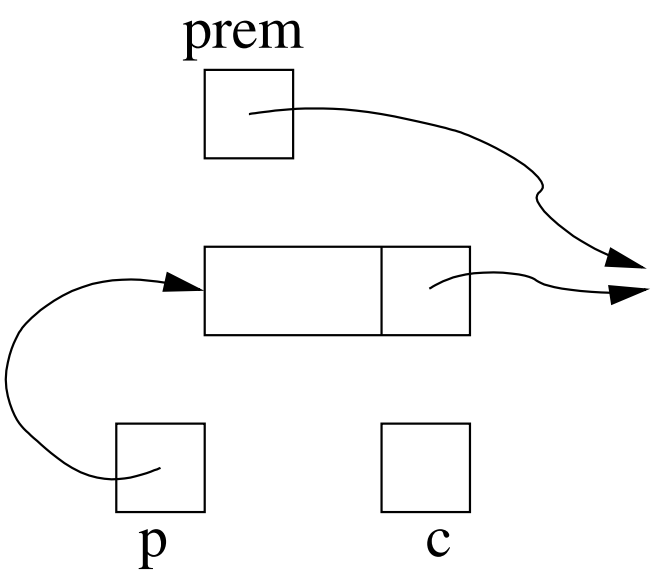
```
type    item = ...
type    tampon : record     element : item ;
                            suiv : pointer to tampon ;
                 end ;
var     prem, p, c : pointer to tampon ;
nextp, nextc : item ;
prem := nil ;
parbegin
```

```
//      Producteur

while (true)
      ...
      produire item nextp
      ...
      new(p);              // 2
      p.elem := nextp;// 3
      p.suiv := prem; // 4
      prem := p;           // 5
```

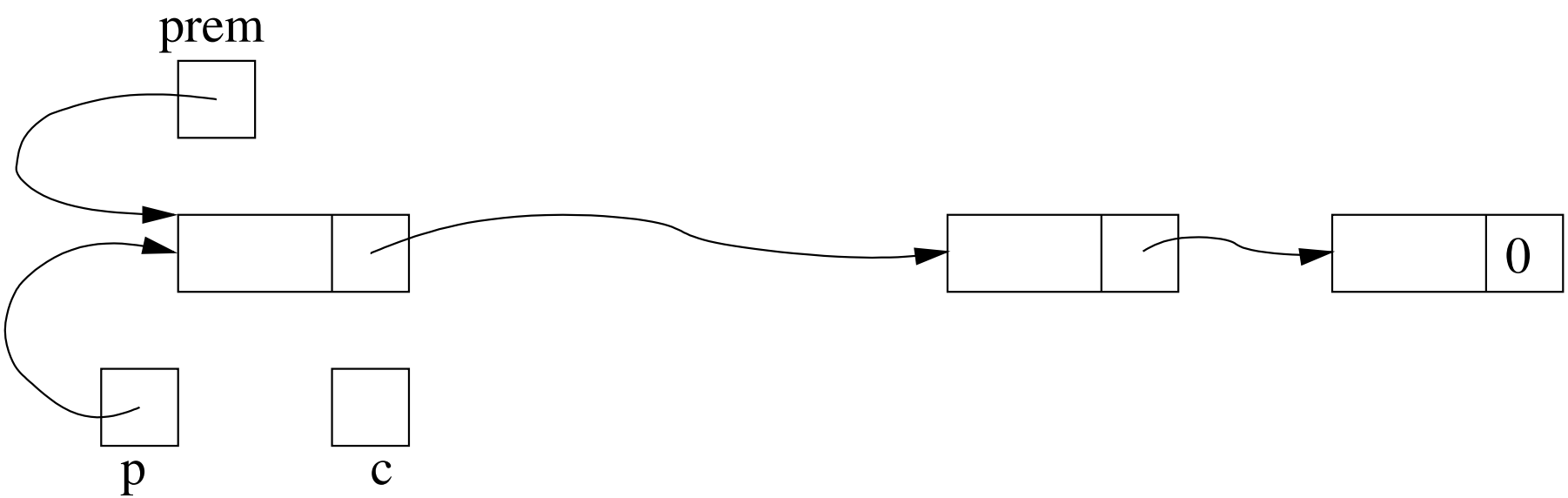
```
//      Consommateur

while (true)
      while (prem=nil);
      c := prem;           // 1
      prem:=prem.suiv;// 6
      nextc := c.elem;// 7
      dispose(c);          // 8
      ...
      traiter item nextc
      ...
```

```
parend
```


prem
0
p
c


prem
p
c


0


prem
0
p
c