

5. gaia: Exekuzio-denborako memoria kudeaketa

5.1 Sarrera eta esleipen estatikoa

5.2 Pila baten bidezko memoriaren esleipen dinamikoa

- » Aldagai ez-lokalen atzipena

- » Parametro-pasatzea

5.3 Memoria esleipen dinamikoa: "Heap allocation"

- » Memoriaren esleipen eta askapen esplizitu eta inplizitua

- » Memoriaren esleipen dinamikorako teknika

5.1 Exekuzio-denborako memoria kudeaketa

- Memoriaren esleipena elementu hauentzat egin behar da:
 - » Objektu-programa
 - » Objektu-programan agertzen diren egiturak, aldagaiak eta konstanteak.
 - » Aldagai iragankorrak, sarrera/irteerako bufferrak, ...

5.1 Memoriaren esleipen estatikoa

- Lengoaia batzuetan konpilazio-denboran erabaki daiteke memoria behar duten objektu guztien tamaina.
 - » Adibidez: FORTRAN
- Memoriaren esleipen estatikoa egiteko prozedura: programaren identifikadore eta konstante bakoitzarekin lotutako espazioa erreserbatzean oinarritzen da.

5.1 Memoriaren esleipen dinamikoa

- Lengoaien ezaugarri hauek memoriaren kudeaketa dinamikoa egitera behartzen dute:
 - » 1. Errekurtsibitatea (C, Ada, Pascal, ...)
 - Bloke habiaratuen egitura (Ada, Pascal, ...)
 - » 2. Memoria esleitu eta askatzeko moduen agerpena (new, alloc, setf, ...)
 - » 3. Luzera aldakorreko datu-egiturak (Lisp, Prolog, ...)

5.2 Pila bidezko memoria esleipena

- Programak kudeatzen duen memoriaren zati bat pila bat bezala erabiliko da: espazioa tontorretik esleitu eta askatuko da
- Azpiprograma baten exekuzioarekin lotutako espazioa bloke batean bilduko da. Bloke honi aktibazio-erregistroa deituko zaio, eta elementu hauetarako espazioa du:
 - parametroak
 - itzultzeko balioak (funtzioak)
 - aldagai lokalak
 - aldagai iragankorrak
 - bueltarako helbidea
 - beste aktibazio-erregistro batzuetarako estekak

Prozedura-deiak

- Dei bat gertatzen denean ondoko ekintzak egin behar dira:
 - deitutako prozeduraren aktibazio-erregistorako espazioaren esleipena
 - dei horretako parametro errealen ebaluazioa eta kokapena aktibazio-erregistroan dagokion tokian
 - prozedurarekiko **orokorrak** diren objektuen atzipena onartuko duten esteken kalkulua
 - **deitzailearen** egoera gorde (erregistroak, ...)
 - itzultzeko helbidea gorde, kontrola bueltatzeko

Prozedura-deiak (II)

- Exekuzioaren bukaera:
 - Azpiprograma funtzioa bada, orduan bueltatu behar den balioa eskuragarri jarri
 - Deitu duen prozeduraren aktibazio-erregistrora bueltatu (**esteka dinamikoa**)
 - Erregistroen egoera berreskuratu eta kontrola bueltatu deiaren hurrengo aginduari (itzultzeko helbidea)
- Hauek egin beharreko ekintzak dira, konpiladoreak sortzen duen kodeak exekutatu beharrekoak. Kode hori azpiprogramaren kodearen barruan edo deian kokatu liteke

Atzipen moduak

a)Memoriaren esleipen estatikoa onartzen duen lengoaia batean atzipen guztiak estatikoki kudeatutako memoriara izango dira

Atzipen moduak (II)

- b) Errekurtsibitatea onartzen bada, baina ez prozeduren habiaraketa (adib. C), prozedura baten kodetik bakarrik honako elementuak atzitu daitezke:
 - » Aldagai orokorrak
 - » Parametroak eta prozeduraren aldagai lokalak
- Exekuzio-denboran honek atzipen zuzenak ekarriko ditu:
 - » Estatikoki kudeatutako memoriara
 - » Pilaren tontorrean dagoen aktibazio-erregistroari

Atzipen moduak (III)

- c) Errekurtsibitatea eta prozeduren habiaraketa onartzen duen lengoaia batean (Pascal, Ada) prozedura baten kodetik honako elementuak atzitu daitezke:
- » Aldagai orokorrak
 - » Parametroak eta prozeduraren aldagai lokalak
 - » Prozedura hori barruan duten prozeduren parametroak eta aldagaiak.

Atzipen moduak (IV)

- Honek exekuzio-denboran honako atzipen zuzenak ekarriko ditu:
 - » Estatikoki kudeatutako memoriara
 - » Pilaren tontorreko aktibazio-erregistrora
- Eta zeharkako atzipenak:
 - » Programa nagusikoa edo lokala ez den aldagai (edo parametro) bat atzitzeko: aldagaia definituta dagoen prozeduraren *azken aktibazio-erregistroa*.

Gogoratzeko

- Prozedura bat errekurtsiboa bada, orduan pila dei errekurtsibo bakoitzeko aktibazio-erregistro bat egongo da pila.
- Une jakin batean, dei horietatik batek bakarrik du kontrola (exekutatzen ari da), eta beste guztiak bere exekuzioaren bukaeraren zain daude.
- Azpiprograma baten barruko prozedura exekutatzen egoteko, beharrezkoa da bere baitan hartzen duen azpiprogramak deitzea.

Gogoratzeko

- Esteka estatikoen kudeaketa beharrezkoa da goi mailako lengoaietan blokeen habiaraketa dagoenean.
- Beste prozedura batean definitutako aldagaien atzipenek dakarte (exekuzio-denboran) esteka estatikoen katearen korritzea.
- Konpiladoreak badauka kodea sortzeko informazio nahikoa, badakielako zein den aldagaiaren erazagupena eta bere erabileraren mailen arteko diferentzia (sinboloen taularen irismenaren informazioa).

Adibidea

Agindu berdina era desberdinetan itzuliko da prozedura batean edo bestean agertzen bada.

```
procedure P1...
```

```
var a...
```

```
    Procedure P11...
```

```
        Procedure P111...
```

```
        begin
```

```
            ...
```

```
            a:=
```

```
            ...
```

```
        end
```

```
    begin
```

```
        ...
```

```
        a:=
```

```
        ...
```

```
    end
```

```
begin
```

```
    ...
```

```
    a:=
```

```
    ...
```

```
end
```

Display

- Aldagai baten memoria posizioa atzitzeko kostua aldagaiaren erazagupena eta erabileraren arteko mailen diferentziaren araberakoa izango da.
- Aurreko problema konpontzeko beste aukera bat DISPLAY izeneko egitura erabiltzea da. DISPLAY egitura taula bat da (normalean makinaren erregistroetan gordeta, atzipen azkarragoa izateko), exekuzioaren une jakin batean atzigarri dauden aktibazio-erregistro guztien erakusleak gordetzeko.

Display

- DISPLAYaren kudeaketa konpiladoreak sortutako kodeak egingo du, honako portaerarekin:
 - » Azpiprograma “sakonagoa” deitzen denean DISPLAYaren tamaina handitu egingo da
 - » tamaina mantendu egingo da maila bereko azpiprograma deitzen bada
 - » tamaina gutxitu egingo da maila orokorreko prozedura deituz gero

Adibidea (I)

```
program Adibide;
```

```
...
```

```
  procedure P1;
```

```
  ...
```

```
  procedure P3;
```

```
  ...
```

```
    procedure P31;
```

```
    ...
```

```
      procedure P311;
```

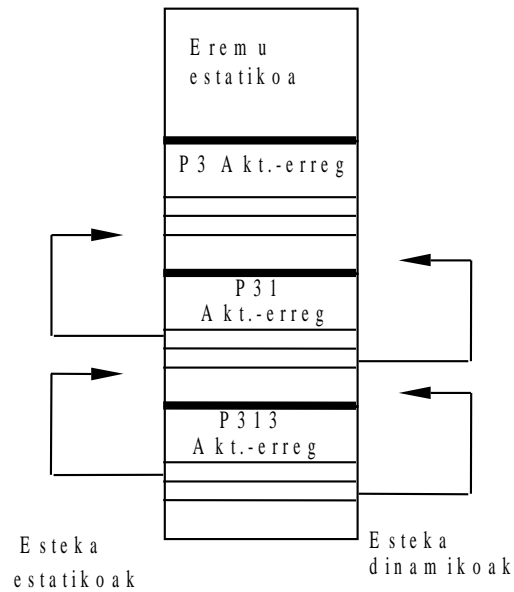
```
      ...
```

```
      procedure P313;
```

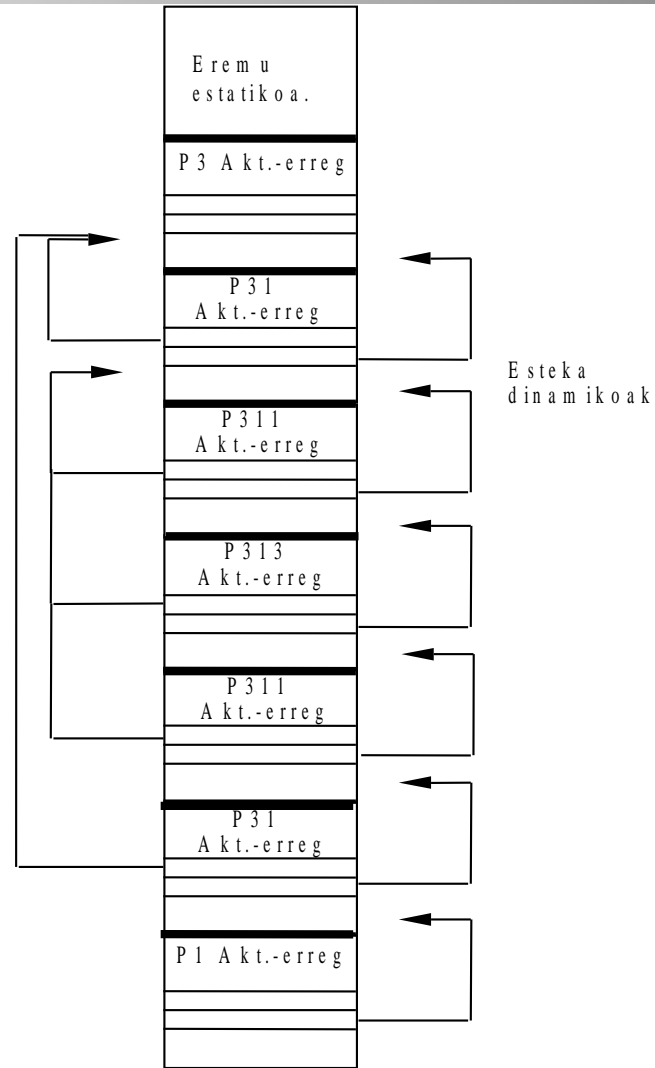
```
      ...
```

- adibidez: Adibide -> P3 -> P31 -> P313 -> P311 -> P311 -> P31 -> P1

Adibidea (II)



a irudia



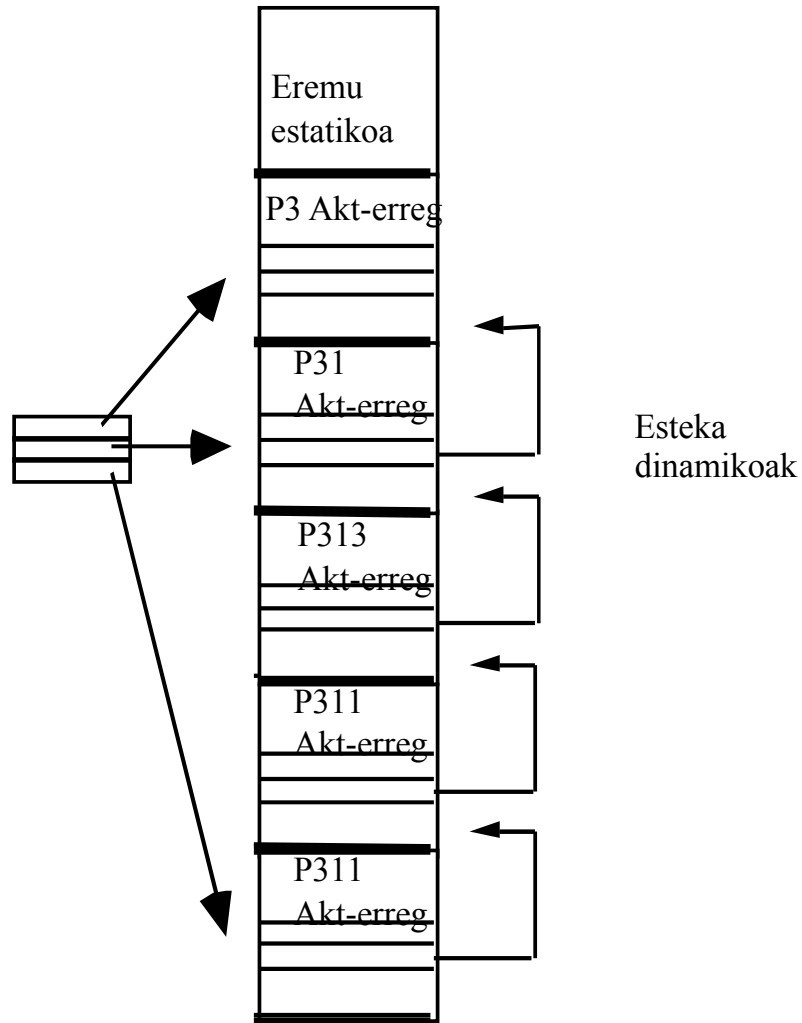
b irudia

Adibidea (III)

```
program Adibide;  
  ...  
  procedure P1;  
  ...  
  procedure P3;  
    ...  
    procedure P31;  
      ...  
      procedure P311;  
      ...  
      procedure P313;  
      ...
```

- adibidez: Adibide -> P3 -> P31 -> P313 -
> P311 -> P311

Adibidea (IV)



Parametro-pasatzea

- balioa (kopia): sarrerako parametroa
- erreferentzia: sarrera/irteerako ala irteerako parametroa
- kopia-berrezarpena: sarrera/irteerako ala irteerako parametroa

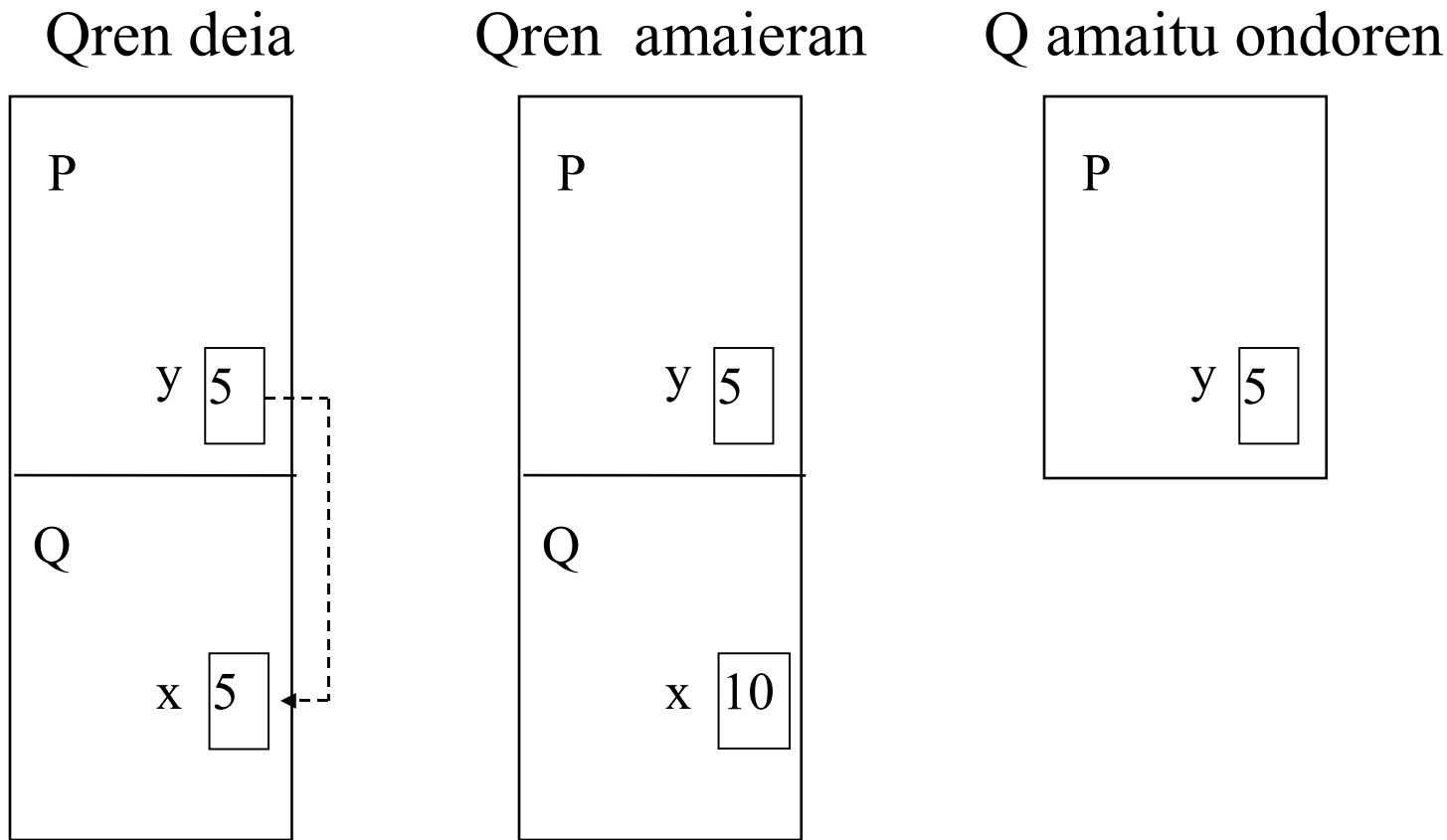
Adibidea

```
procedure P
y: Integer;
  procedure Q(x: integer);
  begin ...
    write(y);
    x := 10;
    write(x);
  end; (* Q *)
begin ...
  y := 5;
  Q(y);
  write(y);
  ...
end; (* P *)
```

Balioz (kopia)

Deitutako azpiprogramak parametro errealairen kopia bat erabiltzen du.

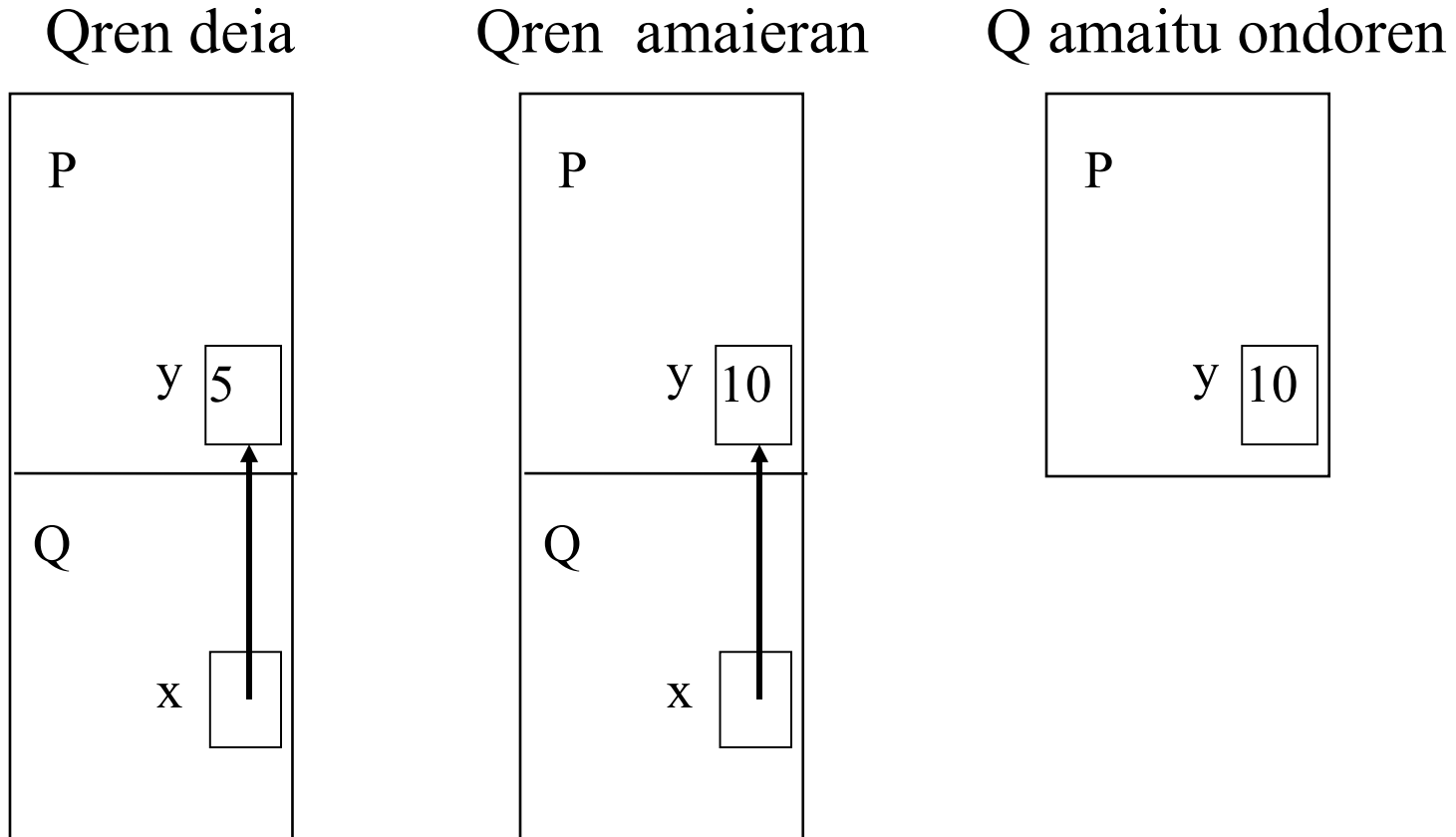
Aktibazio erregistroen pila:



Erreferentziaz

Deitutako azpiprogramak parametroaren memoria helbidea erabiltzen du.

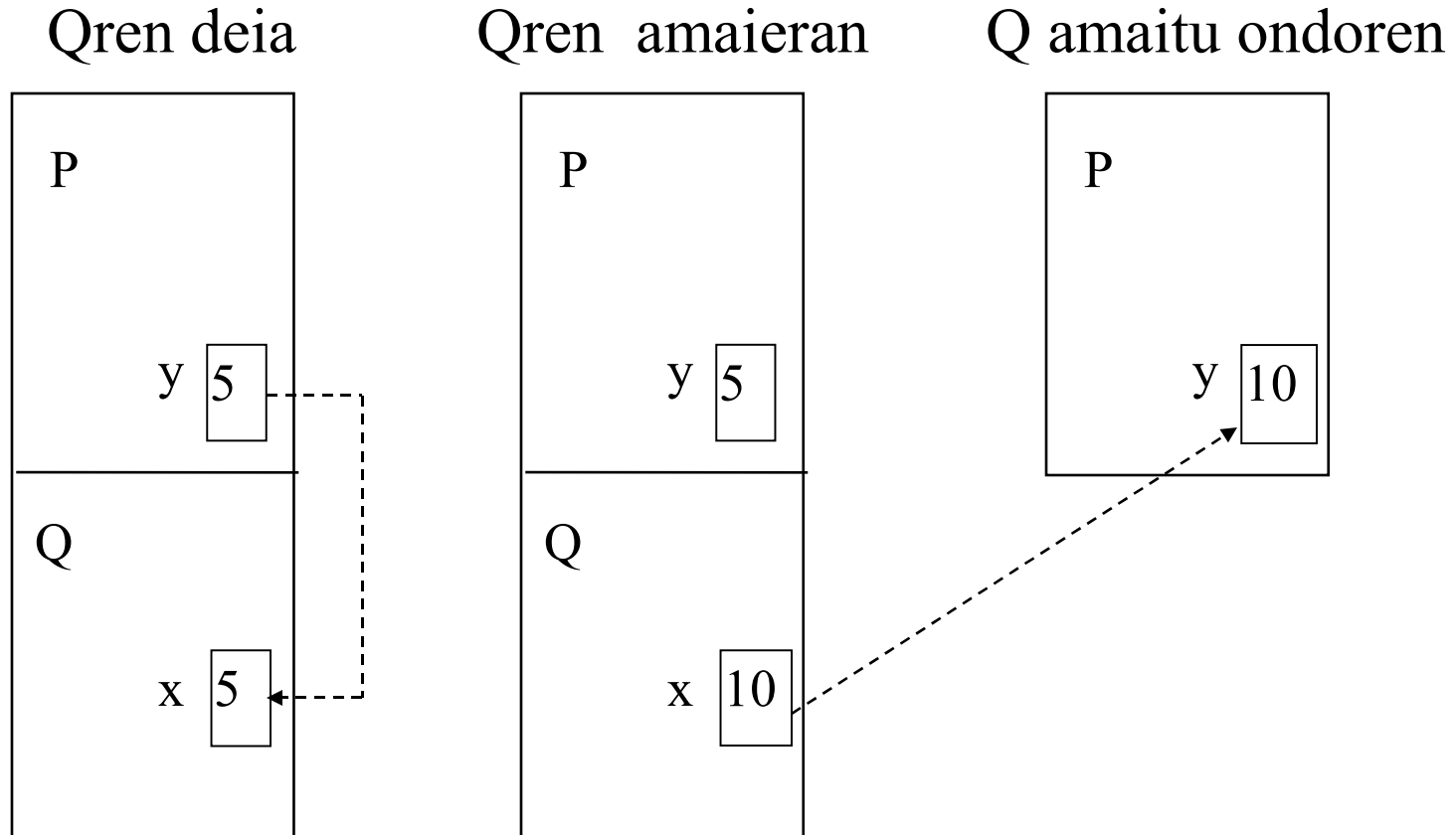
Aktibazio erregistroen pila:



Kopia-berrezarpenaz

Deitutako azpiprogramak parametro errearen kopia bat erabiltzen du eta prozeduraren egikaritzapena amaitzean, emaitza parametro errealean kopiatzen da.

Aktibazio erregistroen pila:



5.3 Heap allocation

Heap: memoria posizioen "piloa". Memoriaren area honetatik behar denean memoria posizioak har daitezke.

Pilaren bidezko esleipenak balio ez duenean erabiltzen da.

Adibidea: erakusleak. Memoria hartu eta askatzen da edozein ordenatan. Eragiketak:

- *new(p)*: p motako objektu bat gordetzeko memoria zatia hartzen du
- *dispose(p)*: p objektuak erabiltzen duen memoria zatia askatzen du

Heap allocation (II)

Memoriaren antolaketa exekuzio-denboran:

Area estatikoa	Kodea + datu orokorrak
Aktibazio erregistroen pila	
Heap	
	Erabilia
	Erabilia
	Erabilia
	Erabilia

Heap allocation (III)

"Heap" arearen kudeaketarako inplementazio posible bat: posizio hutsen lista bat (eta posizio okupatuen beste lista bat, baina ez da beharrezkoa).

- `new(p)`: "p"ri posizio hutsen listako lehen posizioa esleitu. Posizio hori hutsen listatik kendu.
- `dispose(p)`: "p"k erabilitako memoria hutsen zerrendan erantsi.

Heap allocation (IV)

Blokeak tamaina desberdinekoak izan daitezkeenean ("p"ren tamainaren arabera) aukera desberdinak daude:

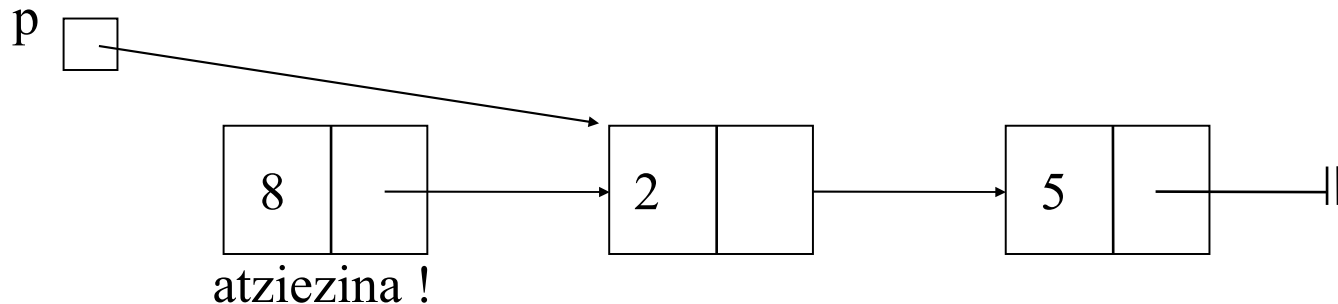
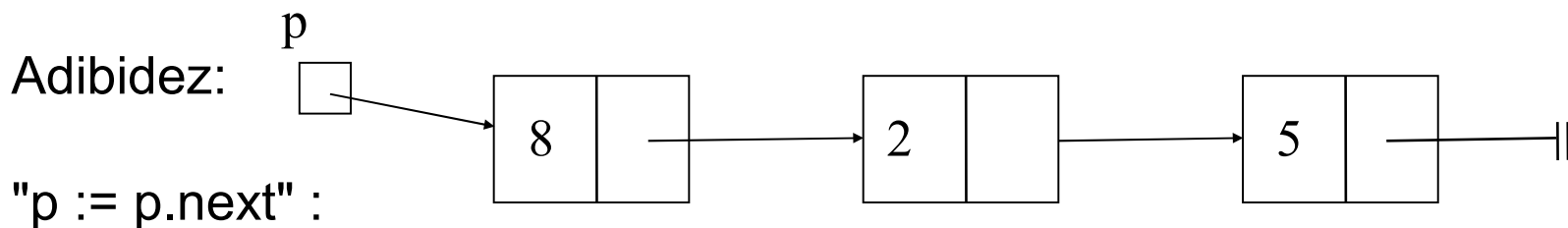
- "p" sartzen den lehen posizio hutsen multzoa erabili. Arazoa: memoriaren zatiketa handia. Tokia badago baina bloke handiak ezin dira sartu.
- "p"ren tamainaren antzekoena duen posizio hutsen multzoa erabili.
- Luzera desberdinetako bloke hutsen listak mantendu.

Heap allocation (V)

Askapen inplizitua

Lengoaia askotan ez dago memoria askatzeko eragiketak (dispose),
Ada eta Lisp adibide.

Horrelakoetan, tarteka zabor bilketa egin behar da (garbage collection):
atziezinak diren baina okupatu gisa agertzen diren memoria blokeak
berreskuratu.



Heap allocation (VI)

Askapen inplizituaren abantailak:

- Abstrakzio maila handiagoa: sistema arduratzen da lan horretaz eta ez programatzailea.
- Segurtasun handiagoa: ez dago "dispose" oraindik erabiliko den posizio batean nahi gabe egiteko aukerarik.

Desabantailak:

- Sistemak "garbage collection" programaren egikaritzapenaren erdian egikaritzen du, hau da, programa motelduko du.