Tipo de dato Pila.

Tipo de dato Pila.

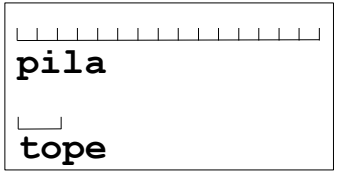
Implementación estática

Implementación dinámica

Tipos de Datos Abstractos

Tipo de dato Pila.

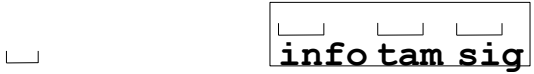
Implementación estática



tPila

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
14  #define      TAM_PILA          340
15
16  typedef struct
17  {
18      char          pila[TAM_PILA];
19      unsigned      tope;
20  } tPila;
```

Implementación dinámica

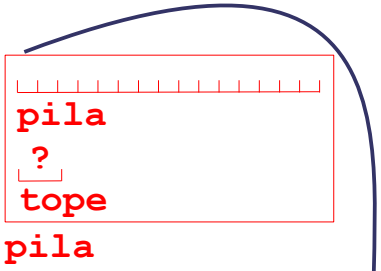


tPila tNodo

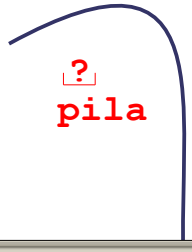
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
13  typedef struct sNodo
14  {
15      void            *info;
16      unsigned         tamInfo;
17      struct sNodo    *sig;
18  } tNodo;
19  typedef tNodo *tPila;
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica

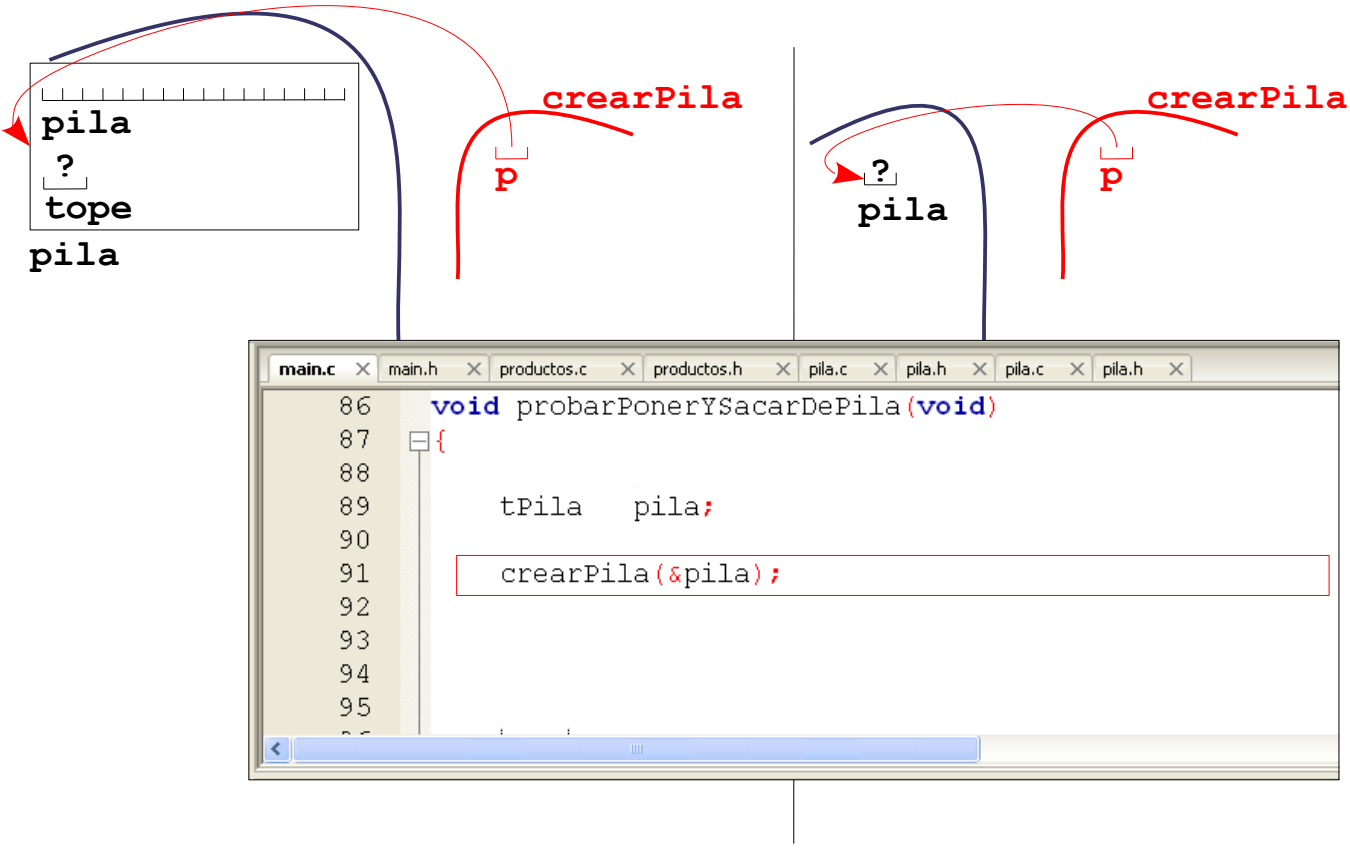



```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
86  void probarPonerYSacarDePila(void)
87  {
88
89      tPila   pila;
90
91
92
93
94
95
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica



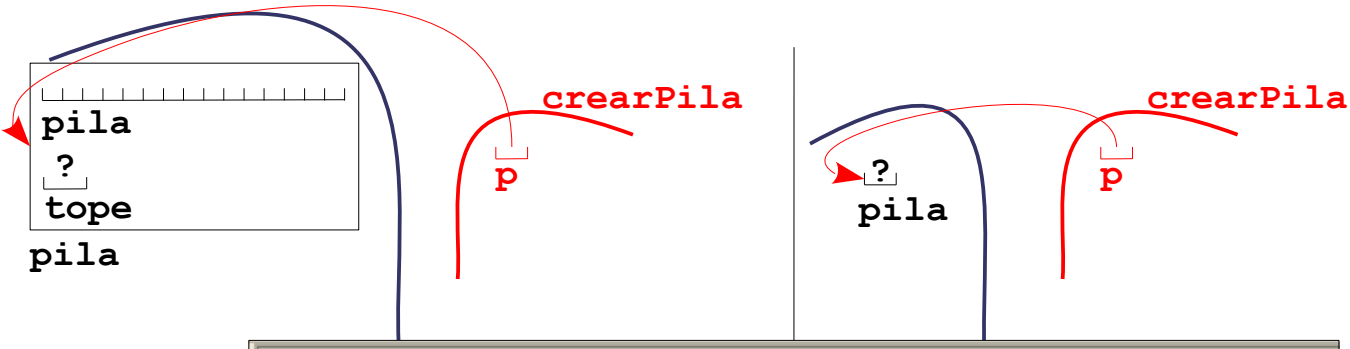

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88
89      tPila   pila;
90
91      crearPila(&pila);
92
93
94
95
```

Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
86  void probarPonerYSacarDePila(void)
87  {
88
89      tPila   pila;
90
91      crearPila(&pila);
92
```

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
 9  void crearPila(tPila *p)
10  {
11      p->tope = TAM_PILA;
12  }
```

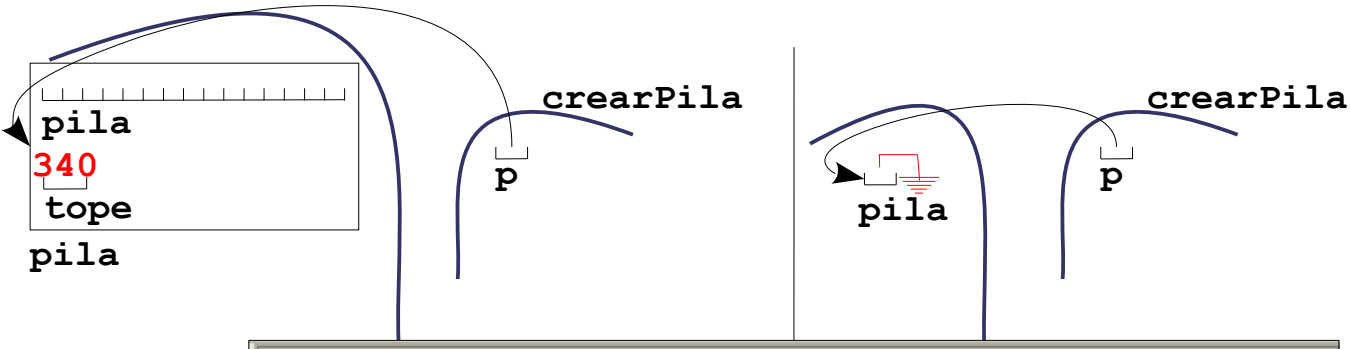
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
 9  void crearPila(tPila *p)
10  {
11      *p = NULL;
12  }
```

Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88
89      tPila   pila;
90
91      crearPila(&pila);
92
```

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h

 9  void crearPila(tPila *p)
10  {
11      p->tope = TAM_PILA;
12  }
```

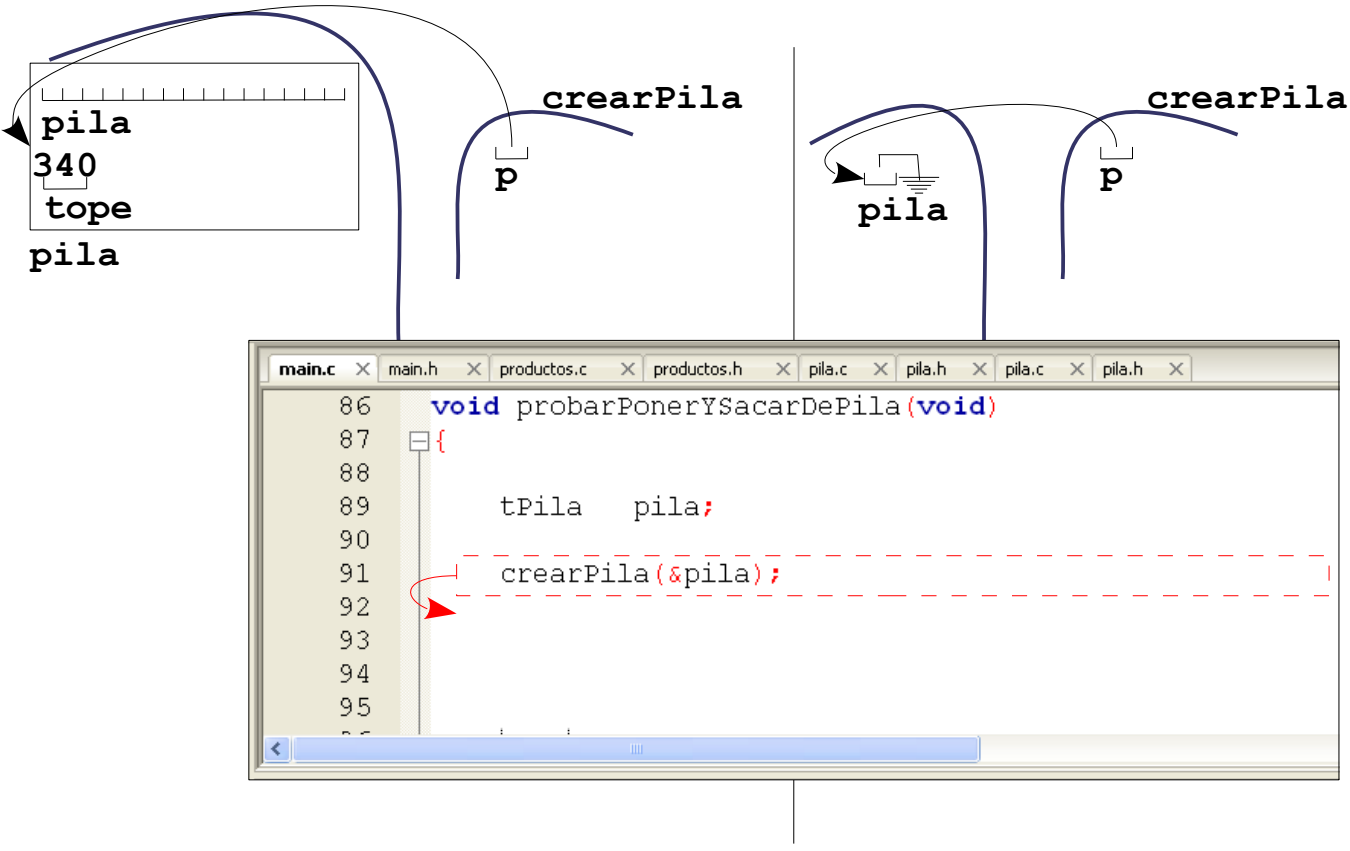
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h

 9  void crearPila(tPila *p)
10  {
11      *p = NULL;
12  }
```

Tipos de Datos Abstractos

Tipo de dato Pila.

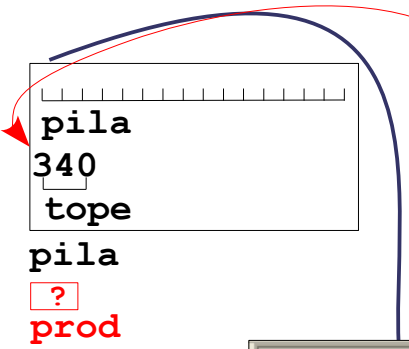
Implementación estática

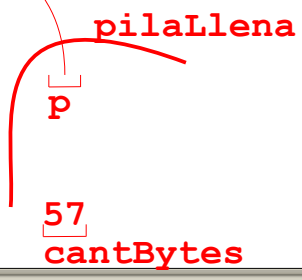
Implementación dinámica



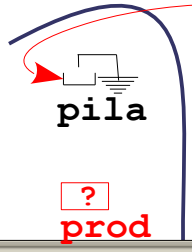
Tipo de dato Pila.

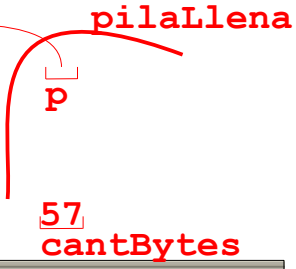
Implementación estática





Implementación dinámica





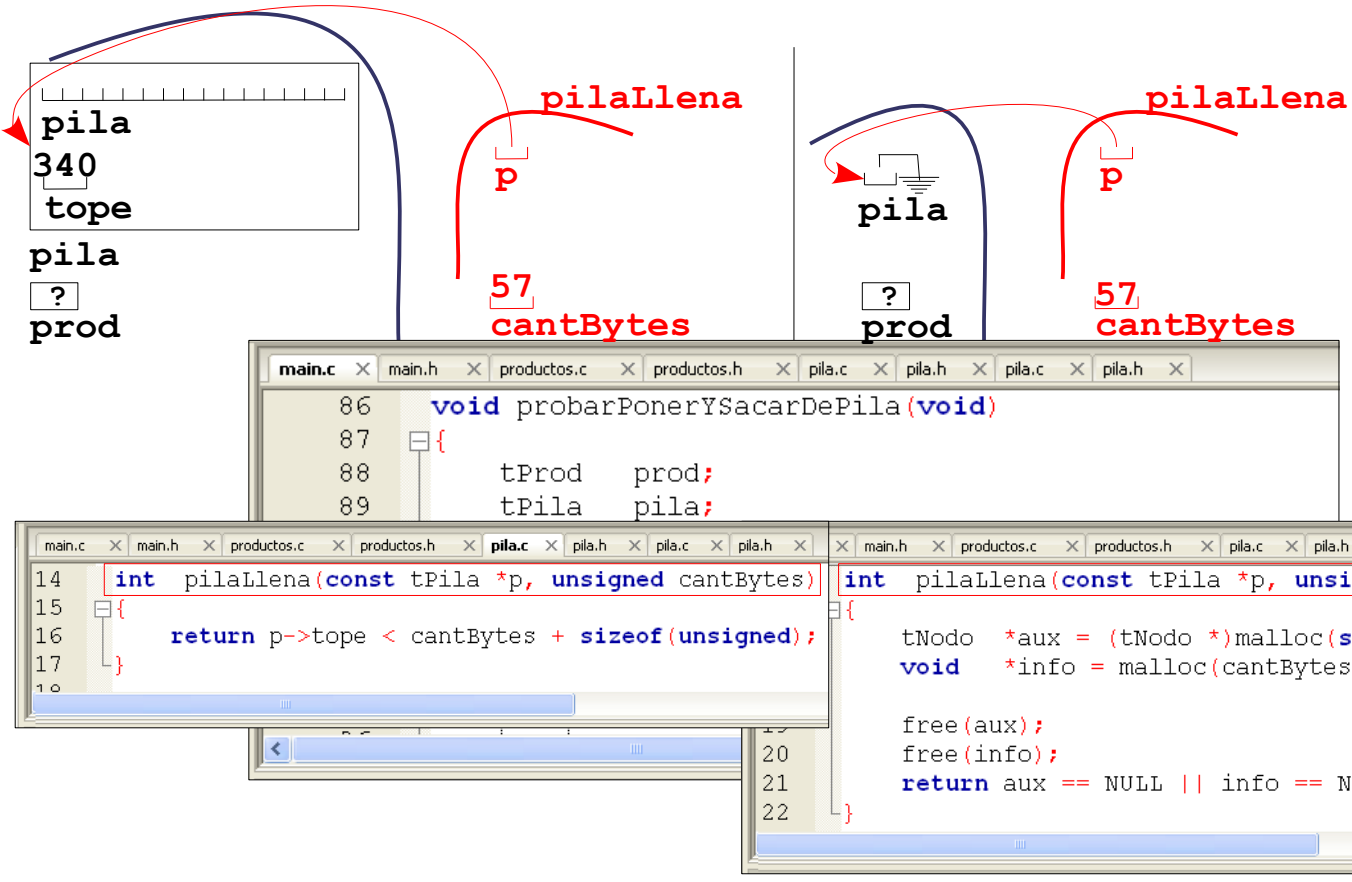

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93
94
95
```

Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

Implementación dinámica

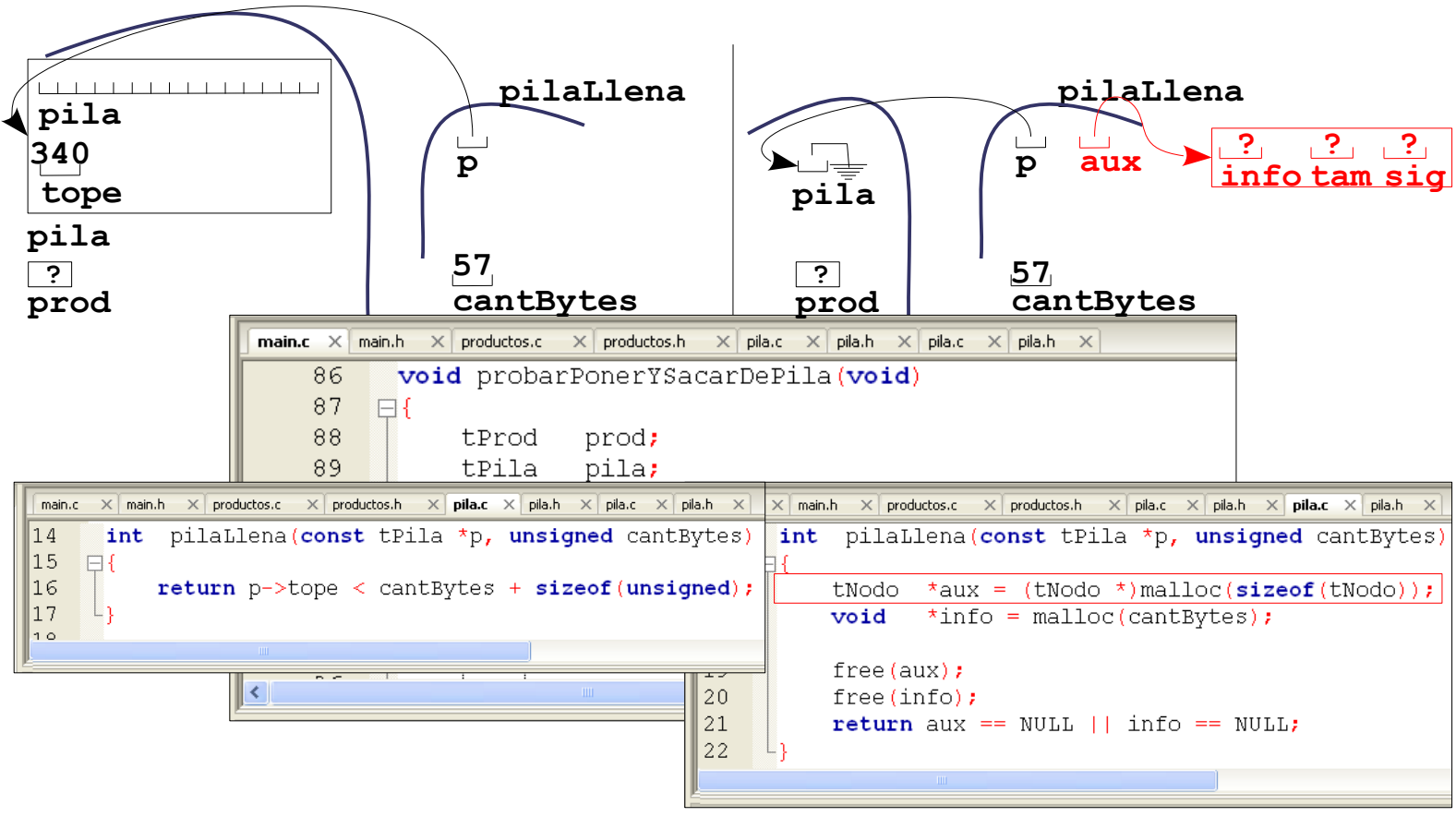


Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

Implementación dinámica

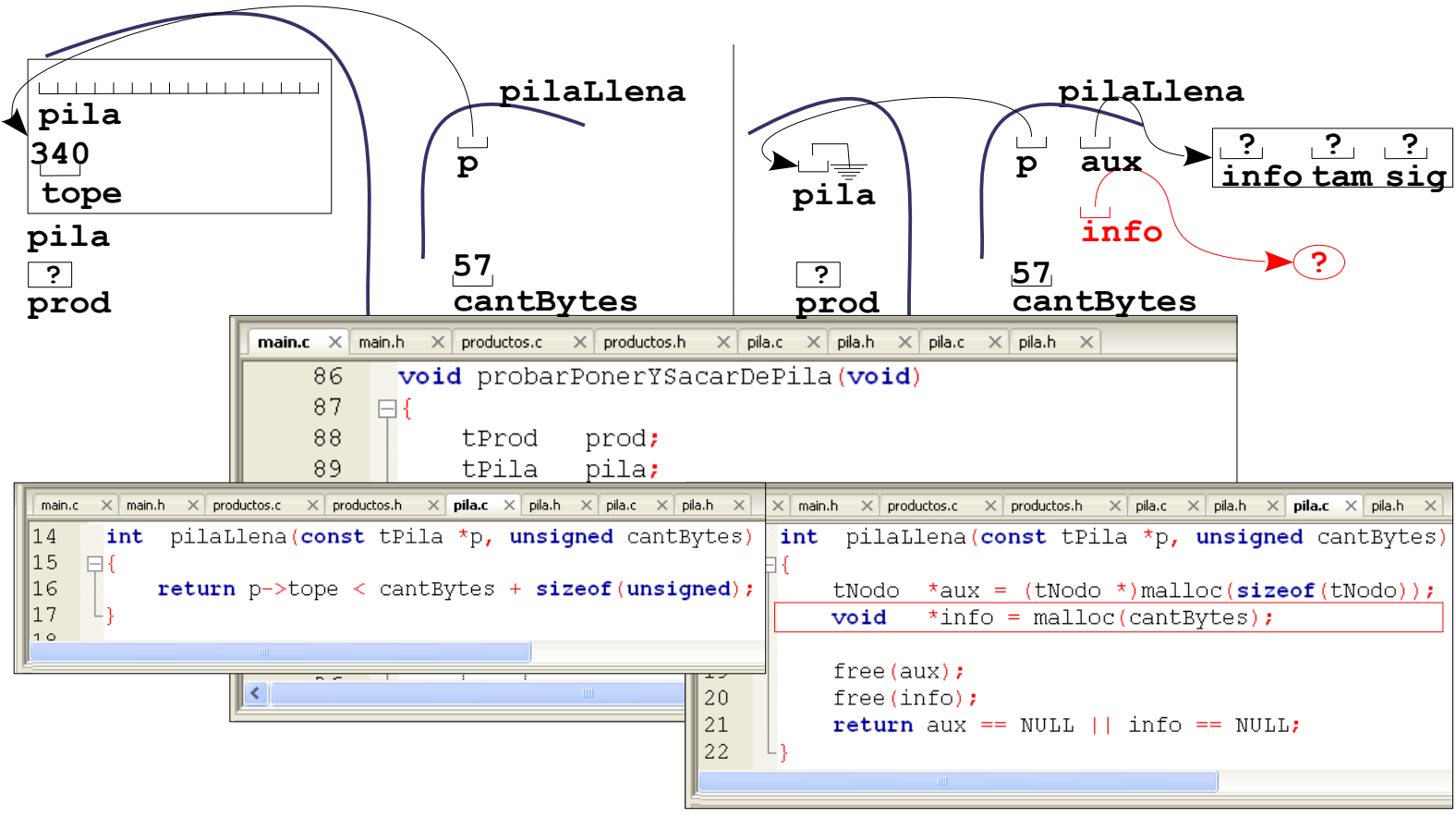


Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

Implementación dinámica

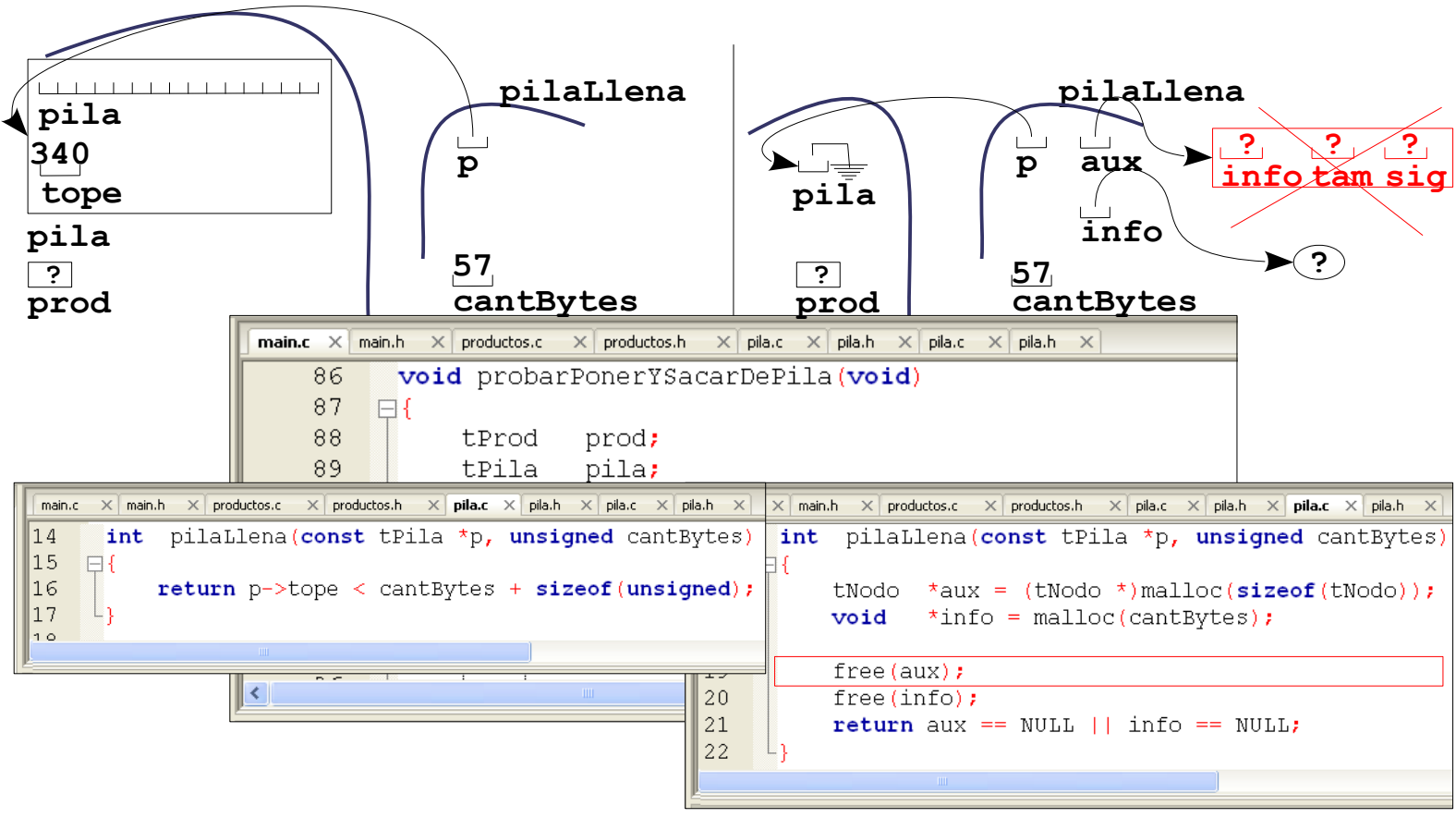


Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

Implementación dinámica

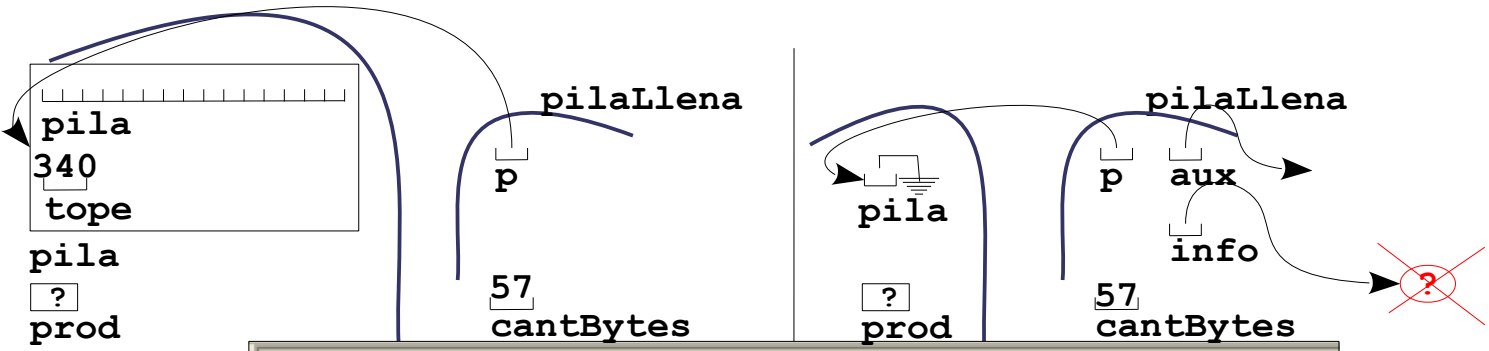


Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

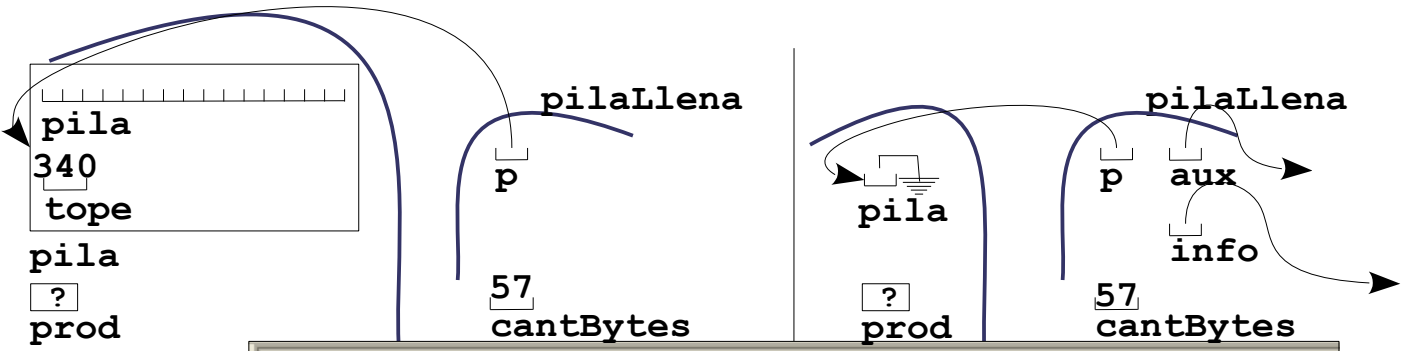
```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

Implementación dinámica



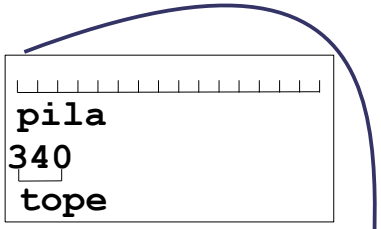

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
20  return aux == NULL || info == NULL;
22  }
```

Tipo de dato Pila.

Implementación estática

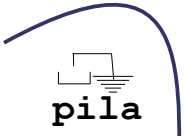


pila

?

prod

Implementación dinámica



?

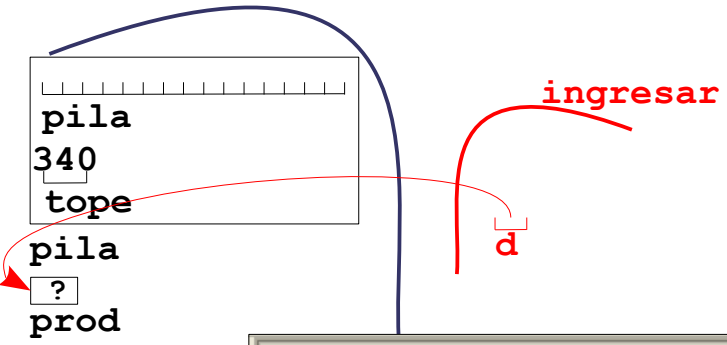
prod

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

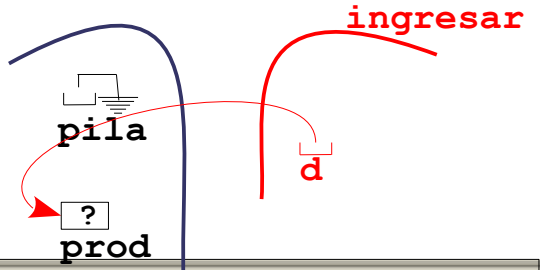
```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94
95
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica

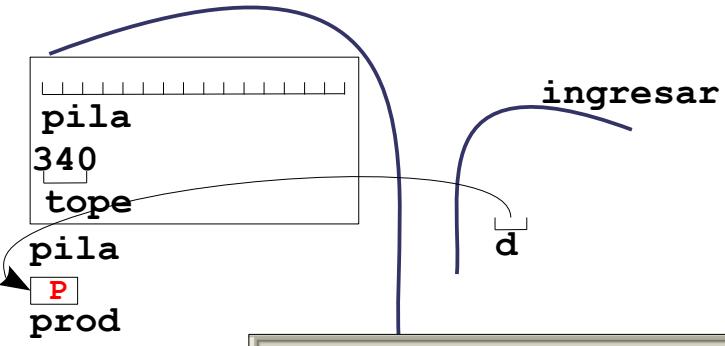


main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

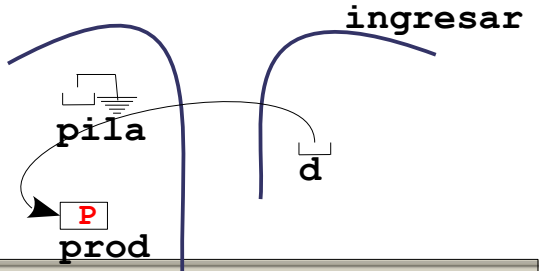
```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94
95
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica

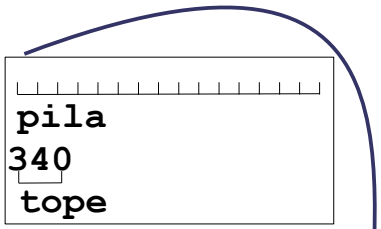


main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94
95
```

Tipo de dato Pila.

Implementación estática



pila

P

prod

Implementación dinámica

P

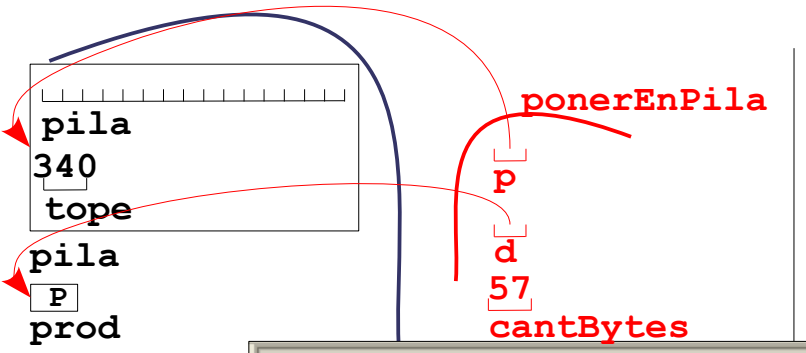
prod

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

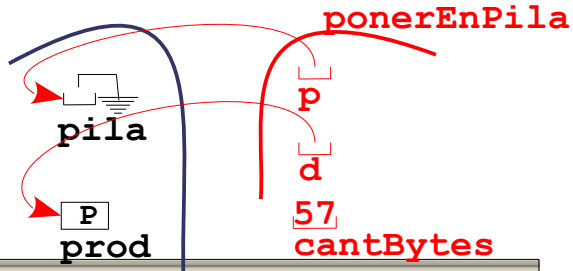
```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica



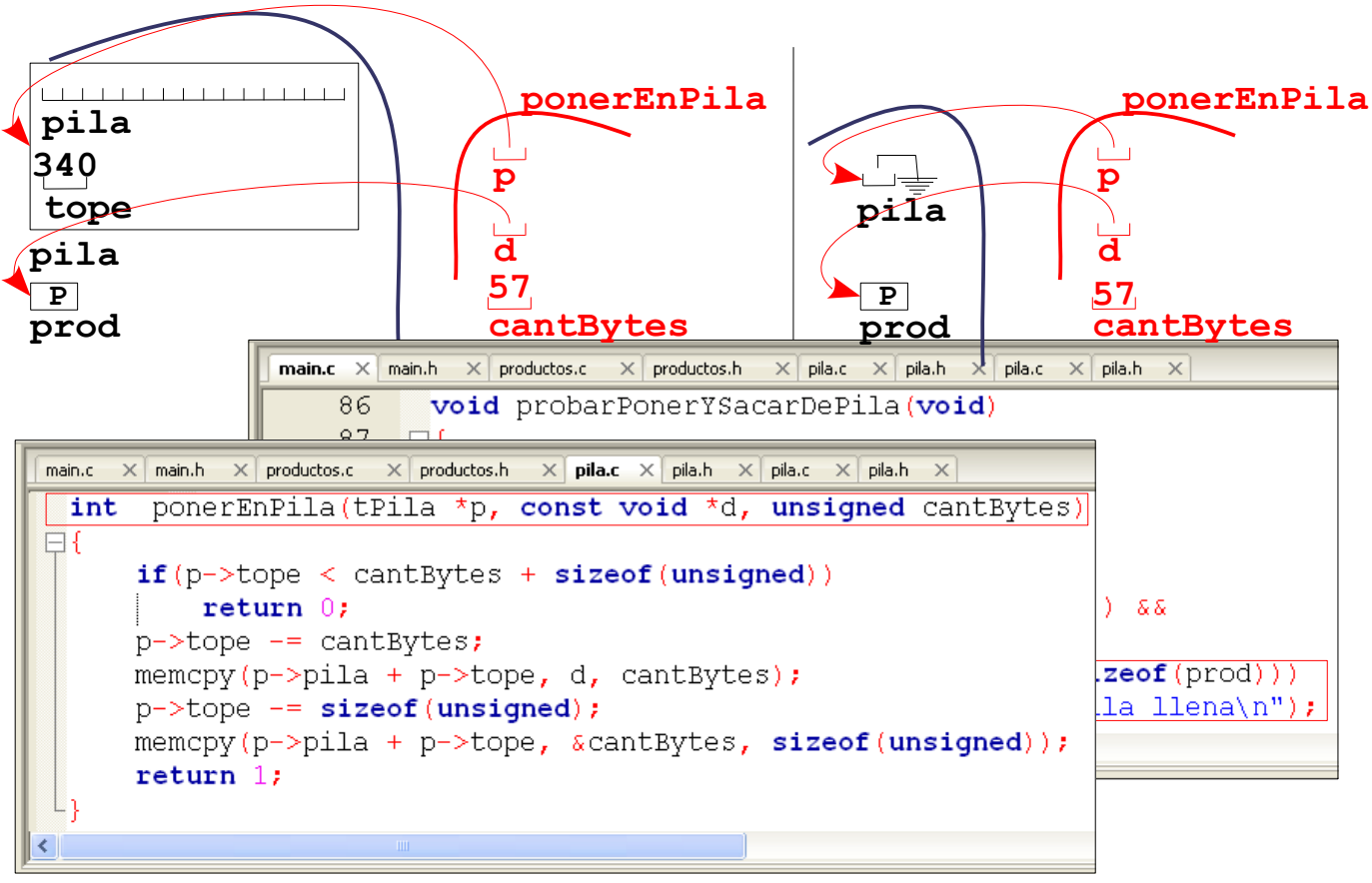
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

Tipo de dato Pila.

Implementación estática

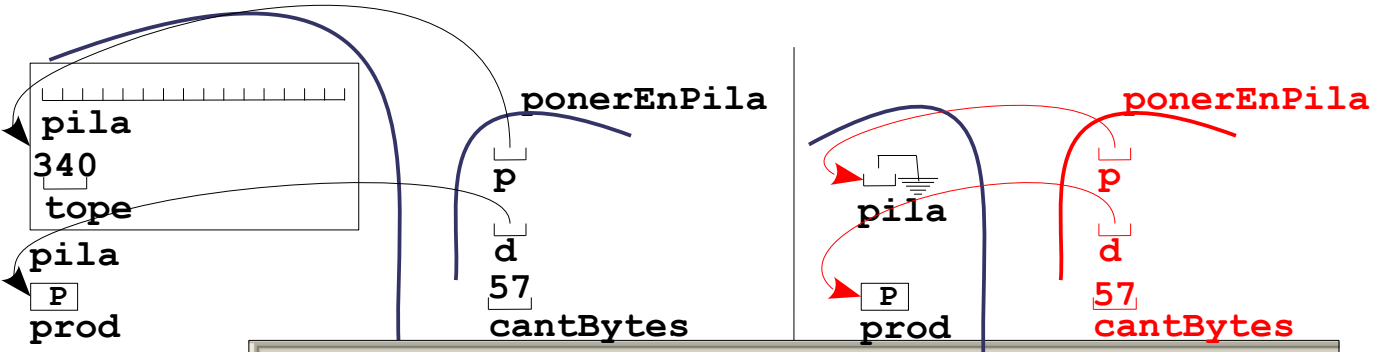
Implementación dinámica



Tipo de dato Pila.

Implementación estática

Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
```

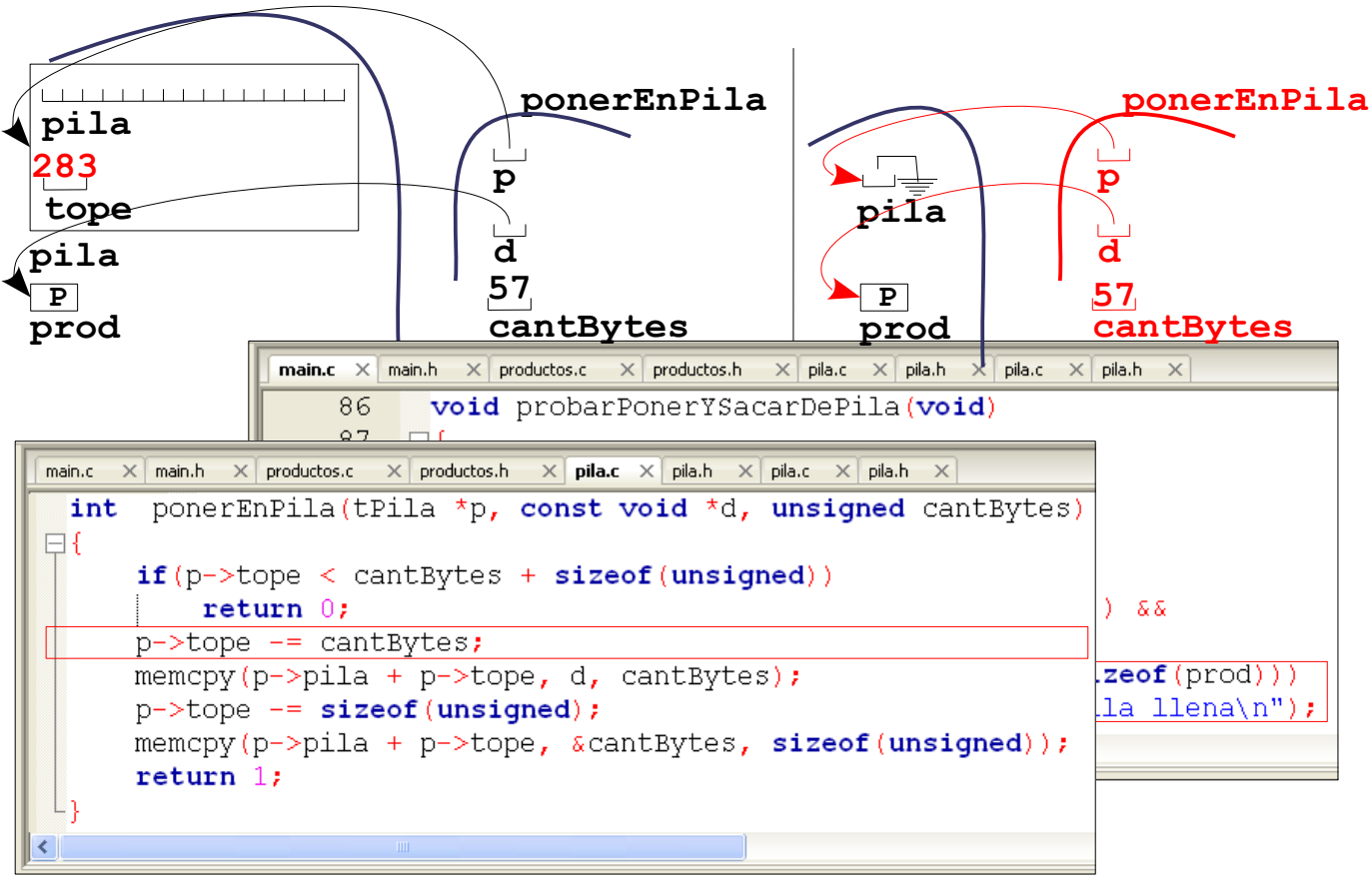
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

Tipo de dato Pila.

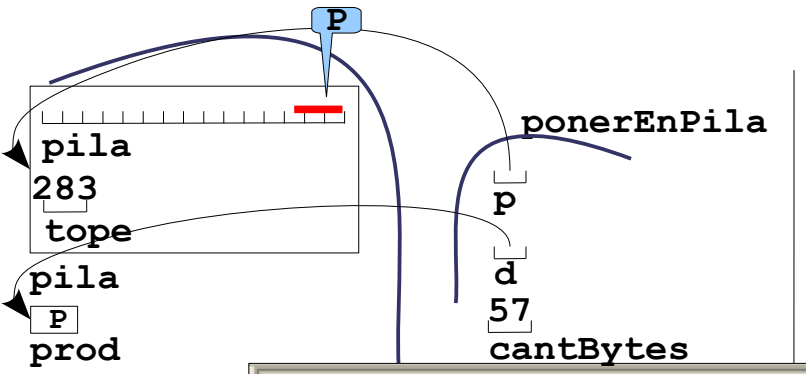
Implementación estática

Implementación dinámica

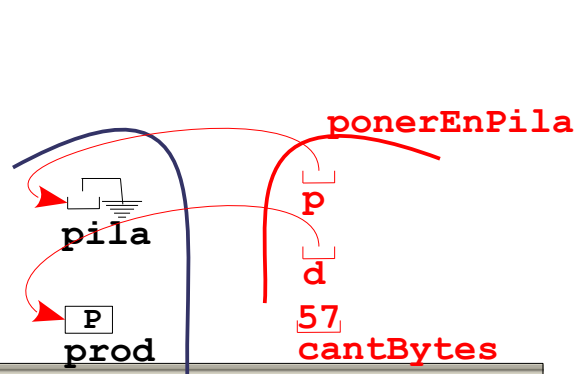


Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
```

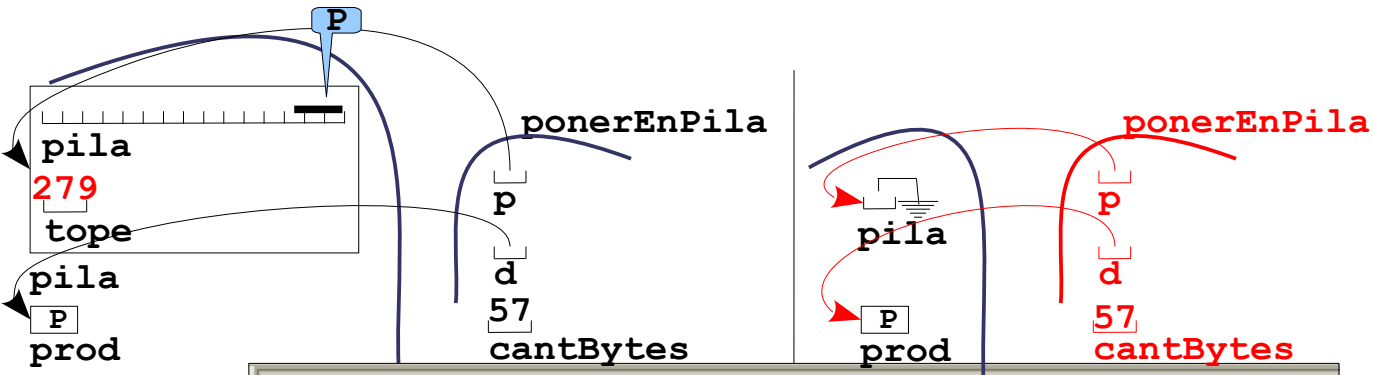
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica



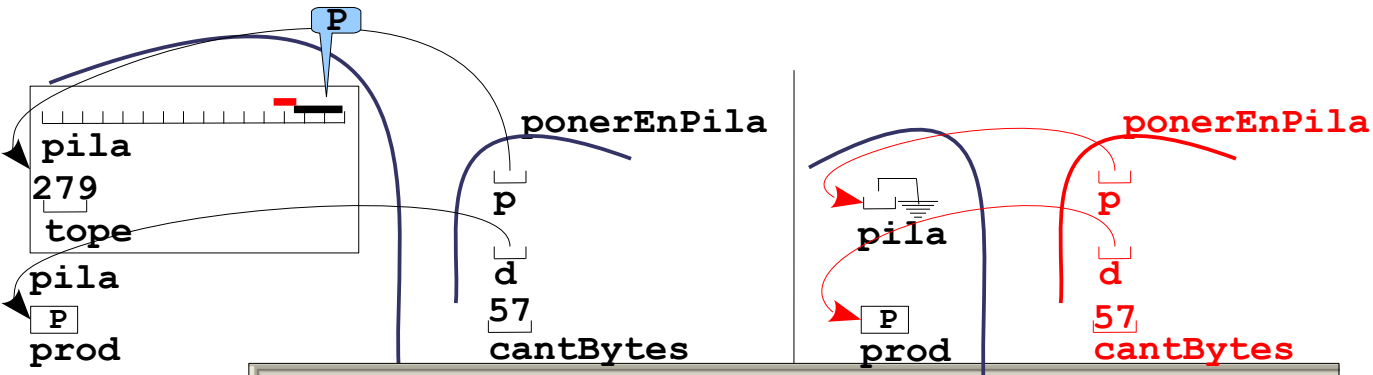

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
86  void probarPonerYSacarDePila(void)
```

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
```

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

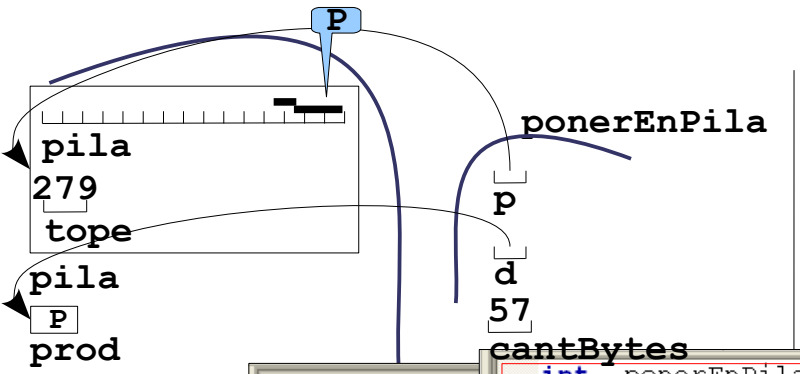
```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

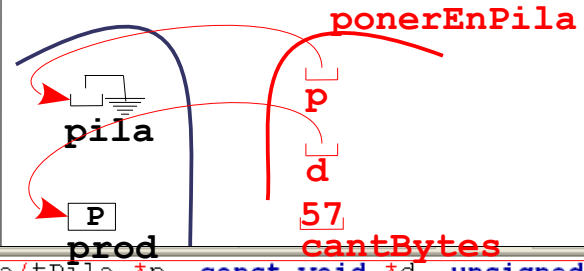
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

Implementación dinámica





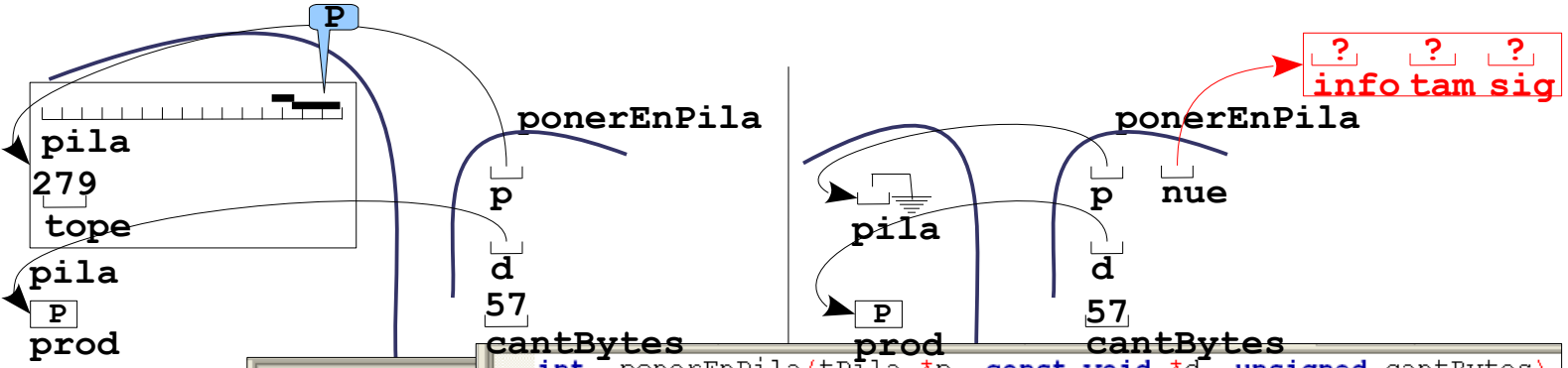

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

Implementación dinámica



main.c | main.h | productos.c

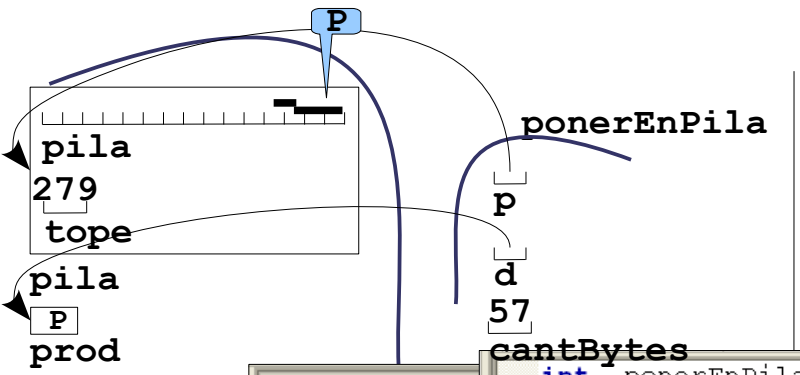
```
int  ponerEnPila(tPila *p
{
    if(p->tope < cantByte
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->t
    p->tope -= sizeof(uns
    memcpy(p->pila + p->t
    return 1;
}
```

main.c | main.h

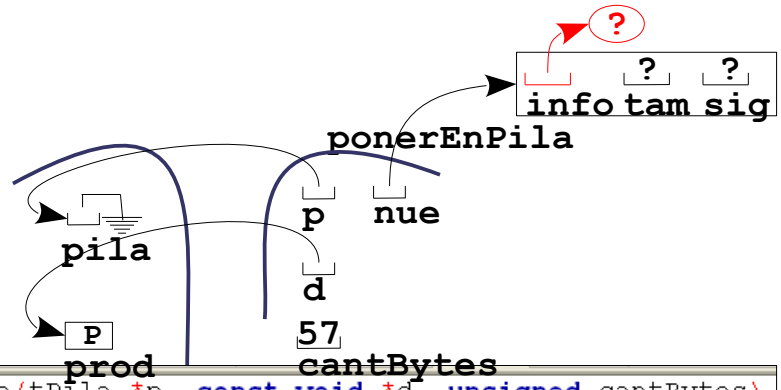
```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica

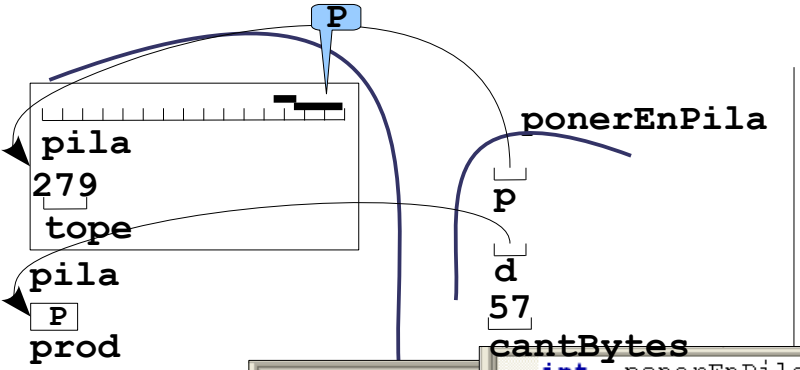



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

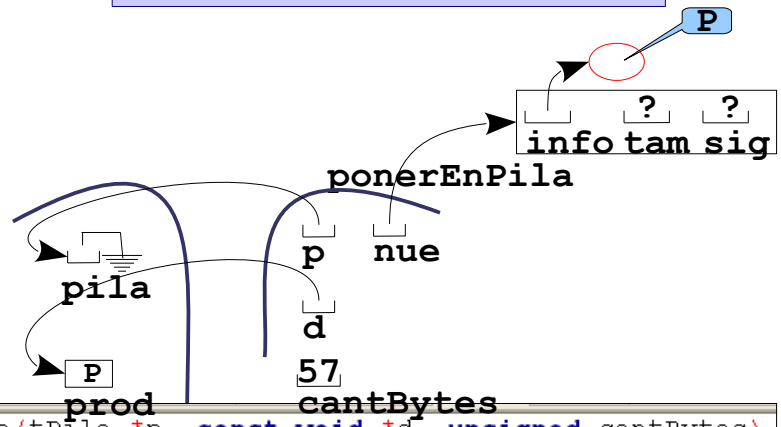
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c

```
int  ponerEnPila(tPila *p
{
    if(p->tope < cantByte
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->t
    p->tope -= sizeof(uns
    memcpy(p->pila + p->t
    return 1;
}
```

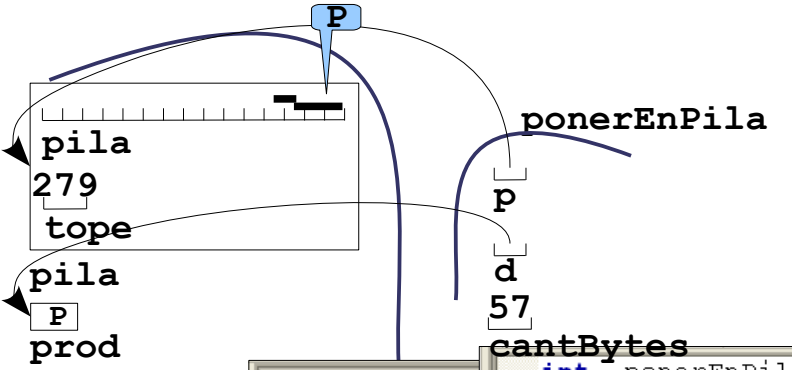
main.c | main.h

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

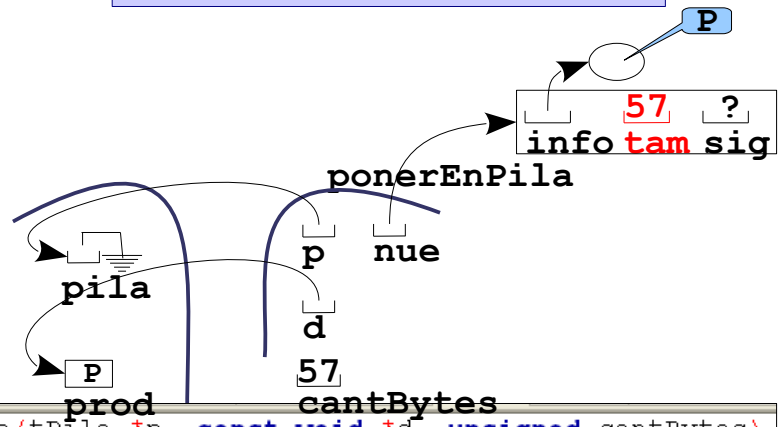
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



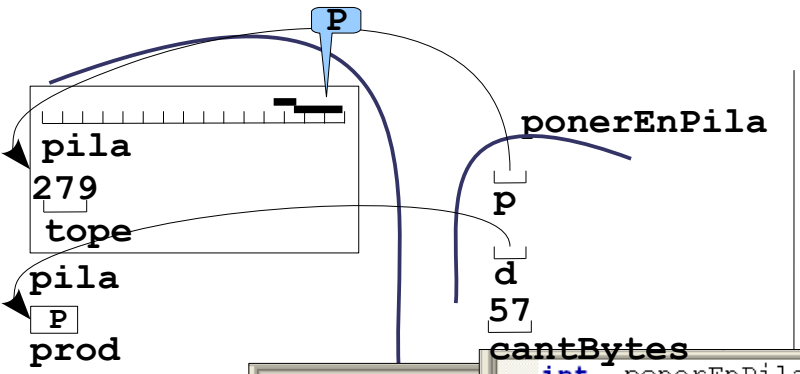
Implementación dinámica



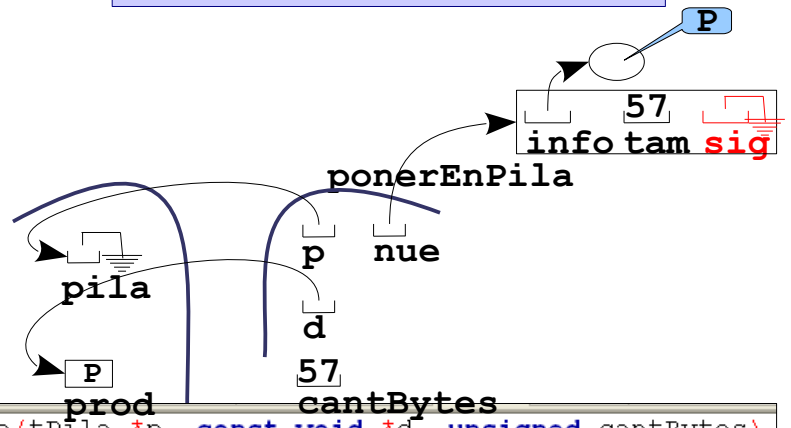

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica

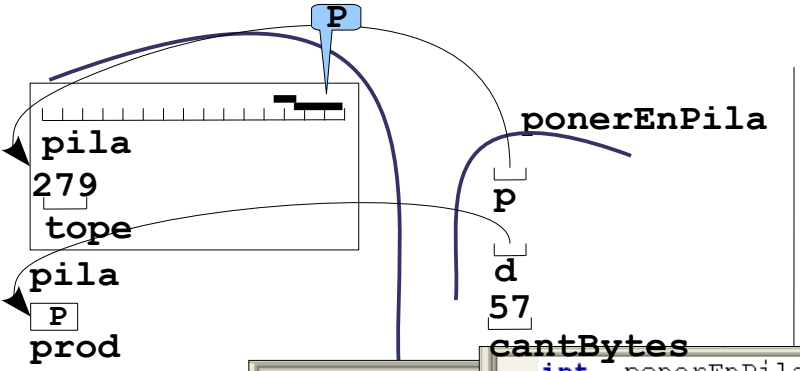



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

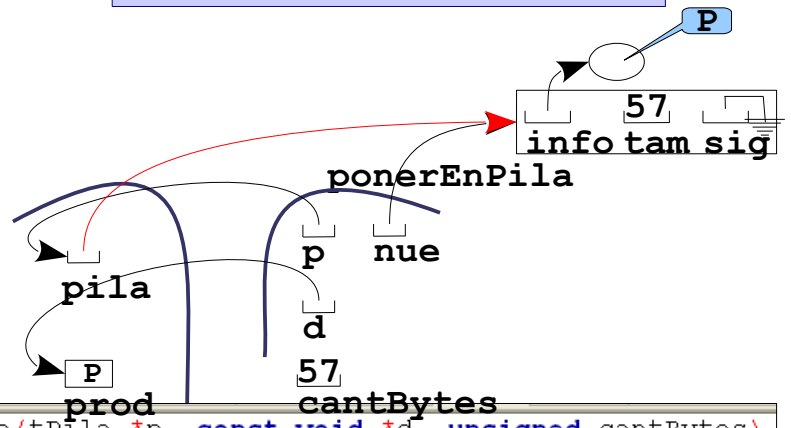
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

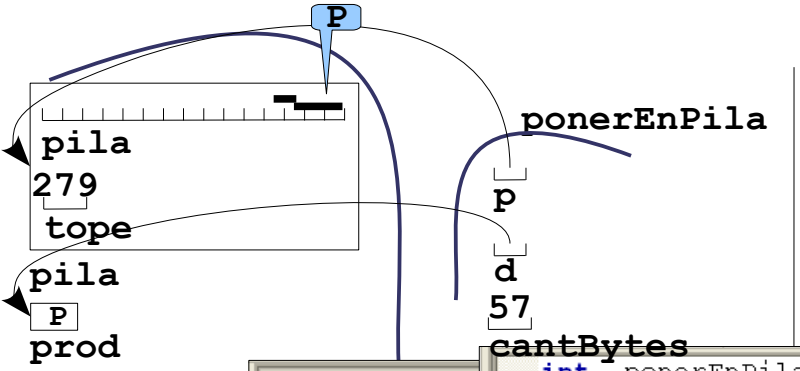



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

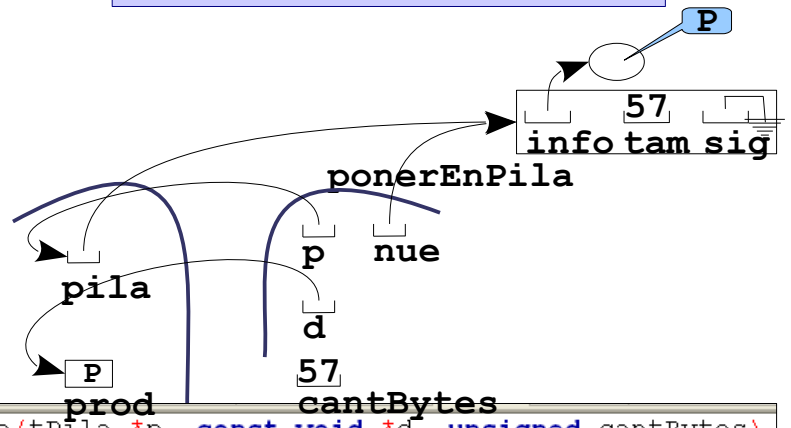
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

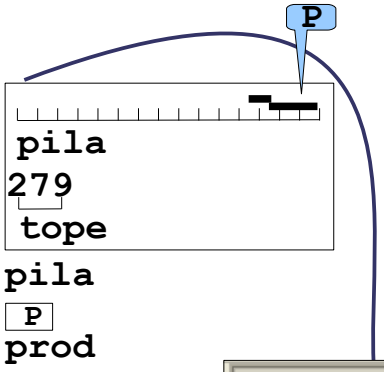



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

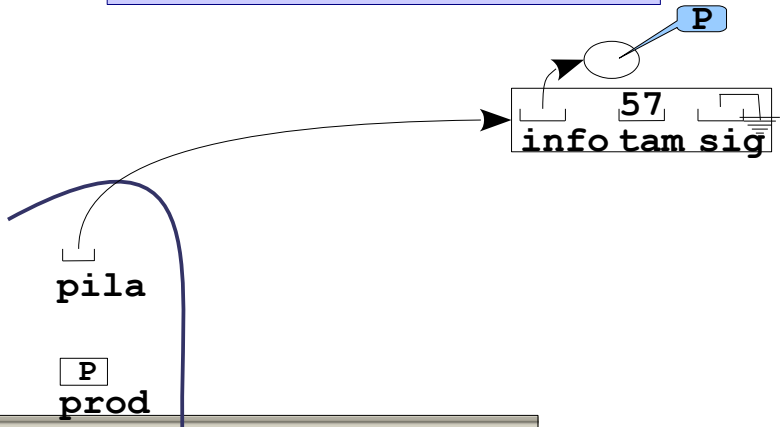
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



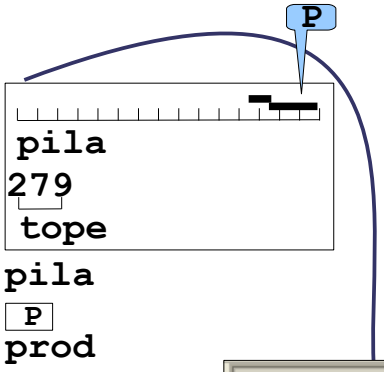
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

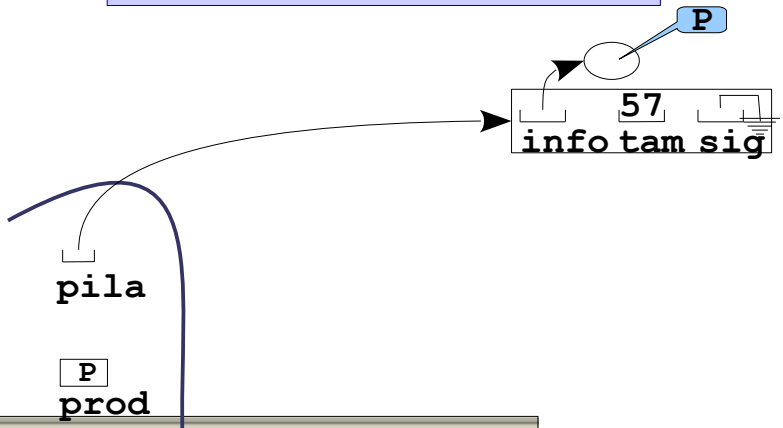
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



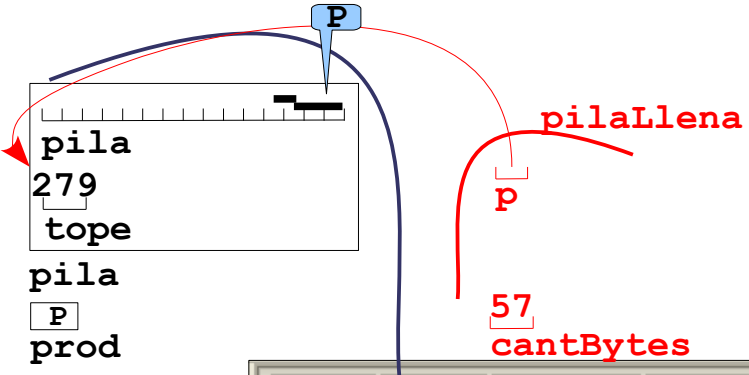
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

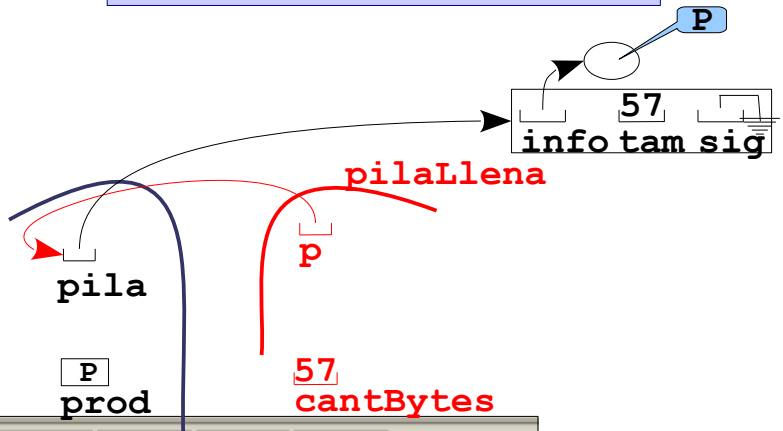
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



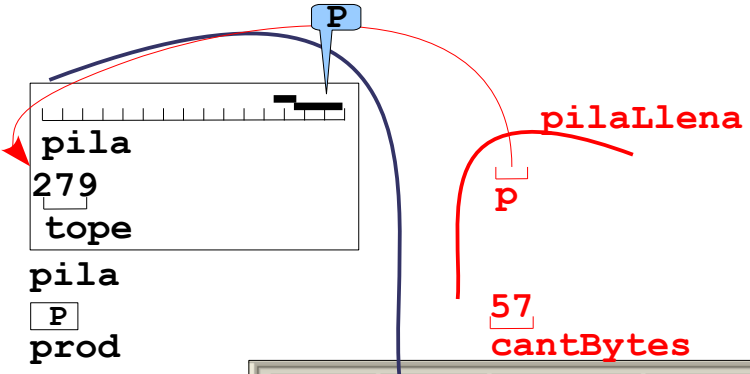
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

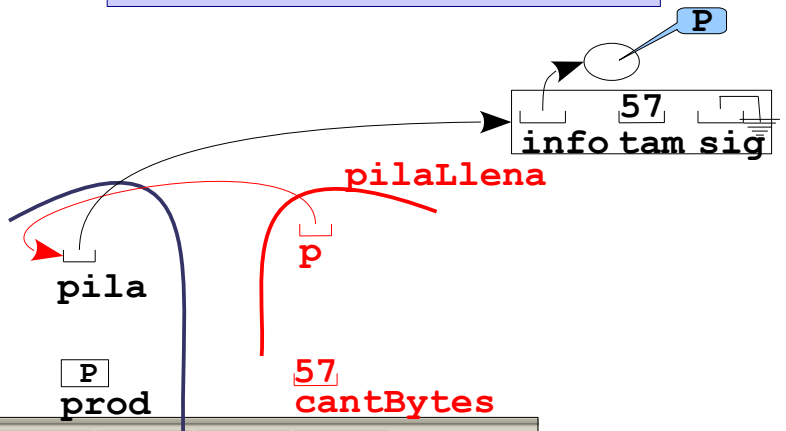
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

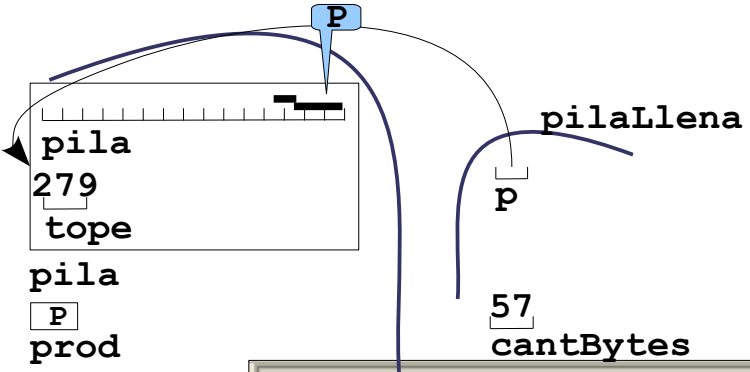
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

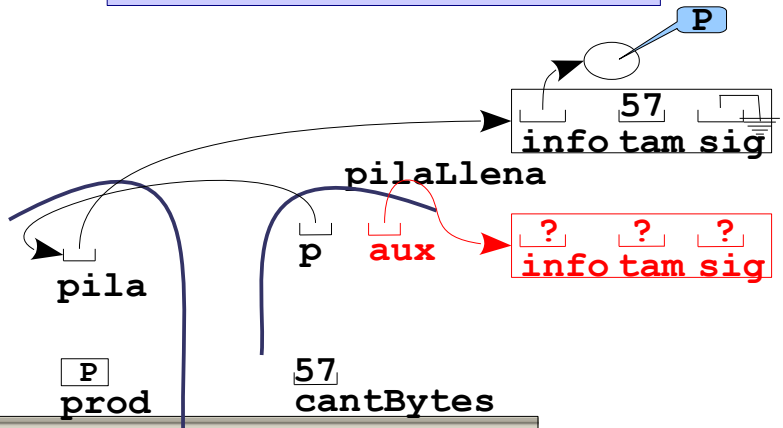
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

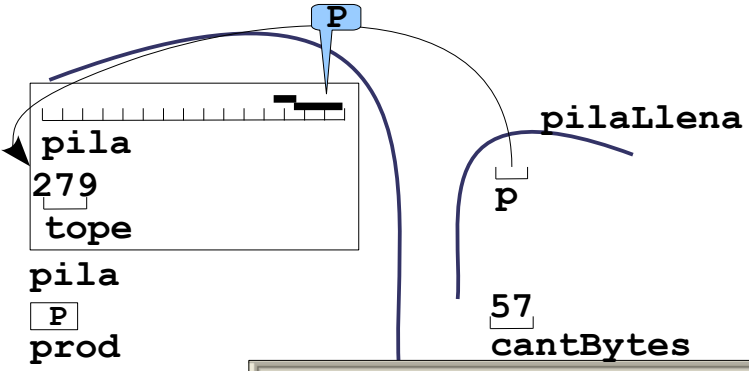
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

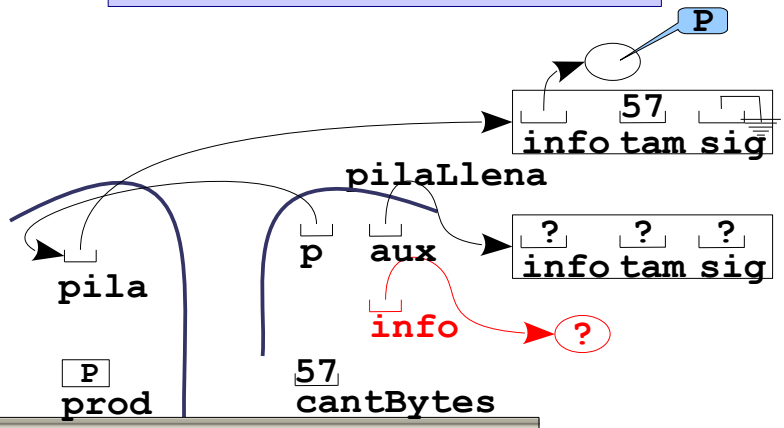
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

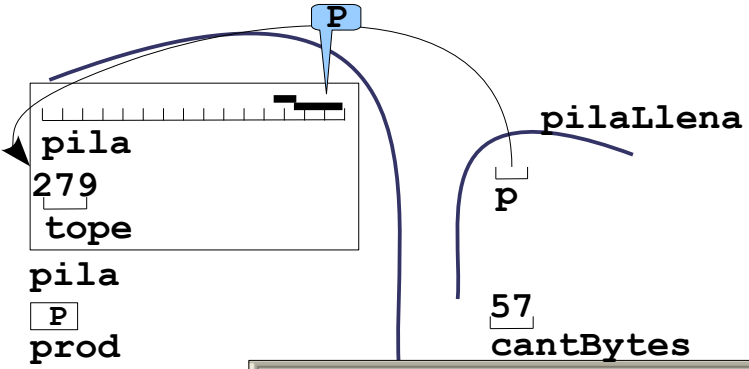
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

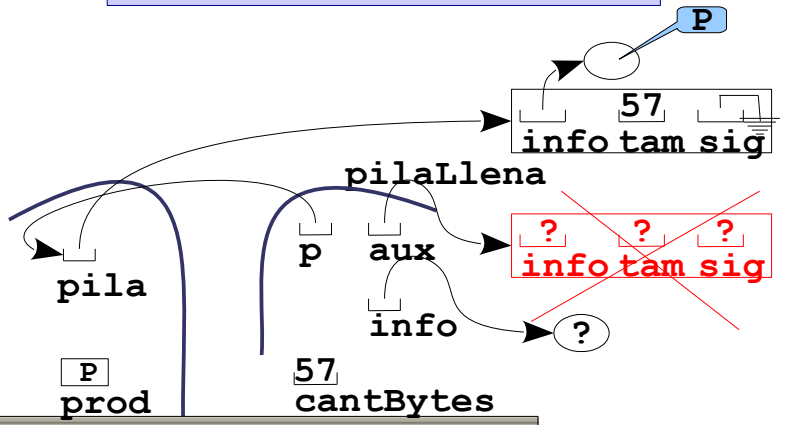
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

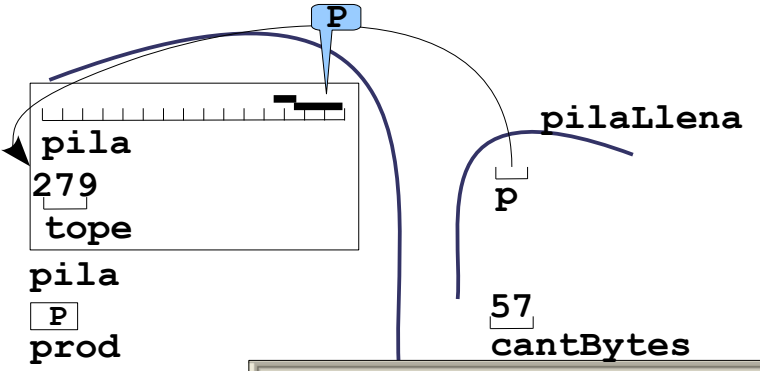
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

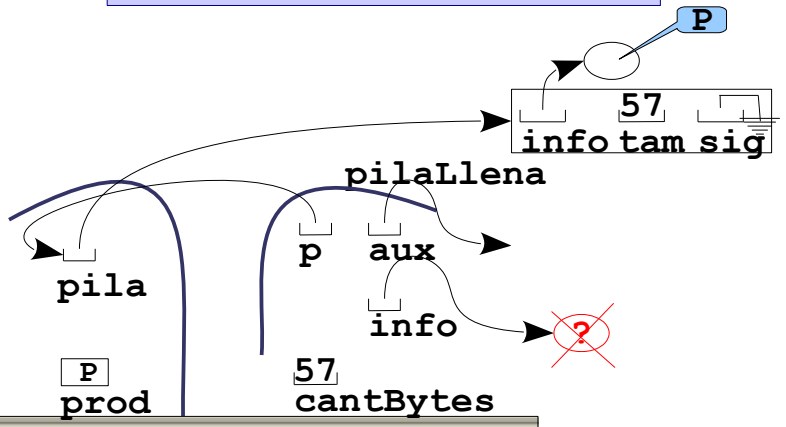
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

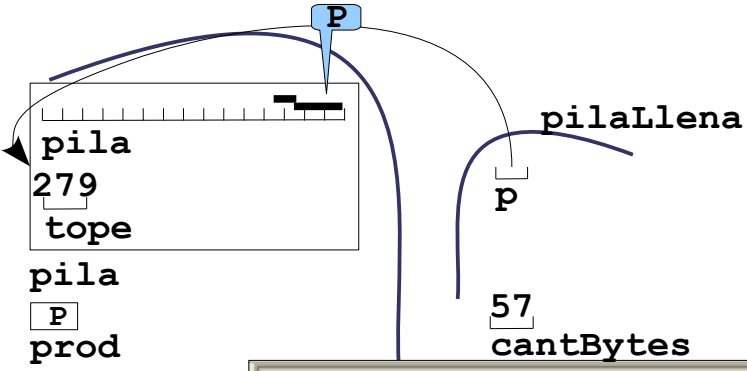
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

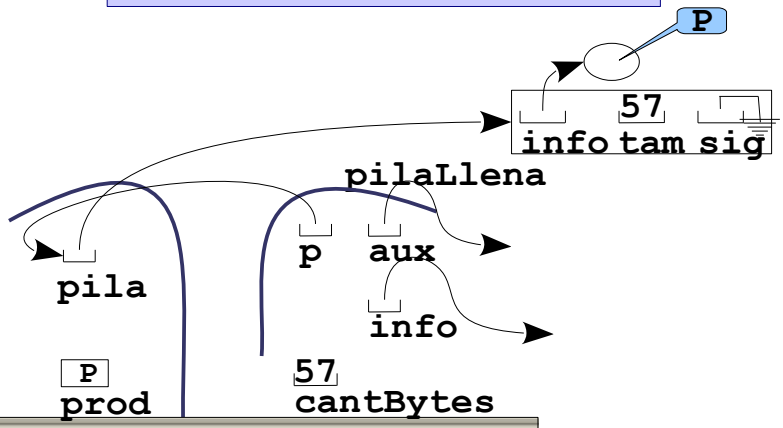
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

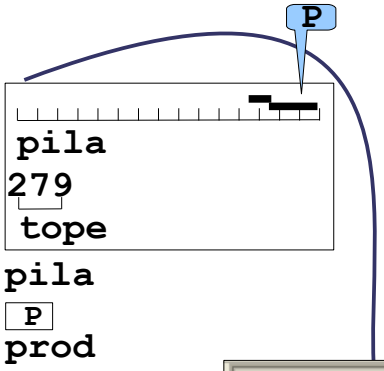
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
20  return aux == NULL || info == NULL;
22  }
```

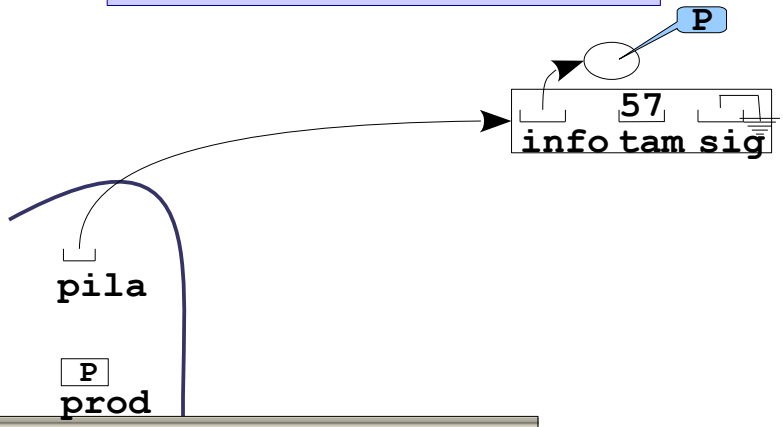
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



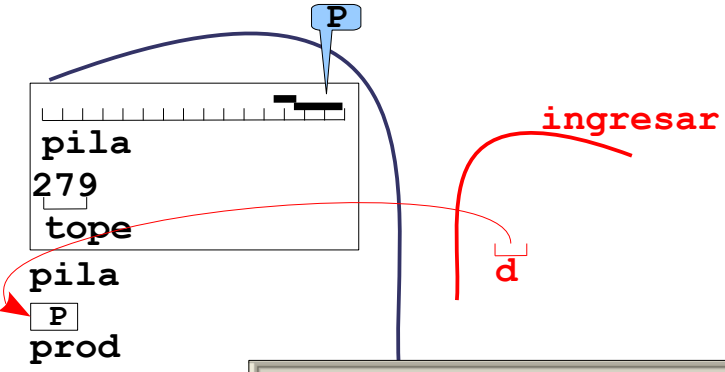
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

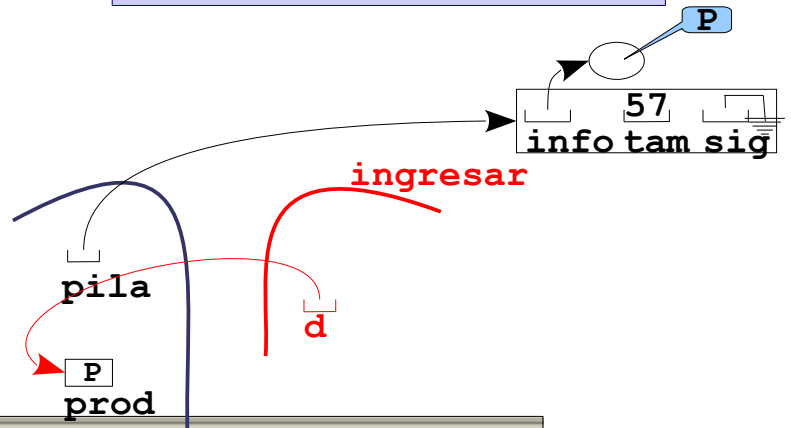
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



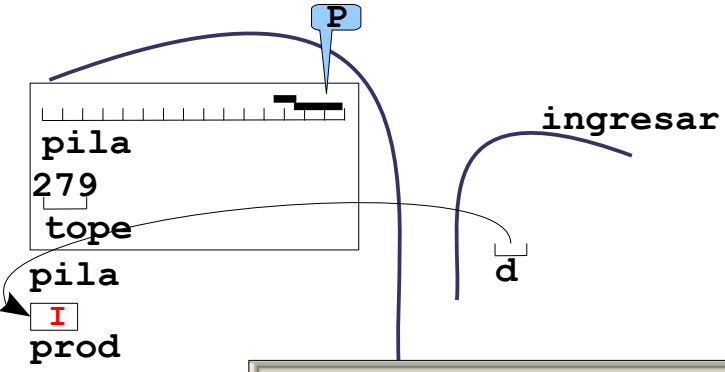
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

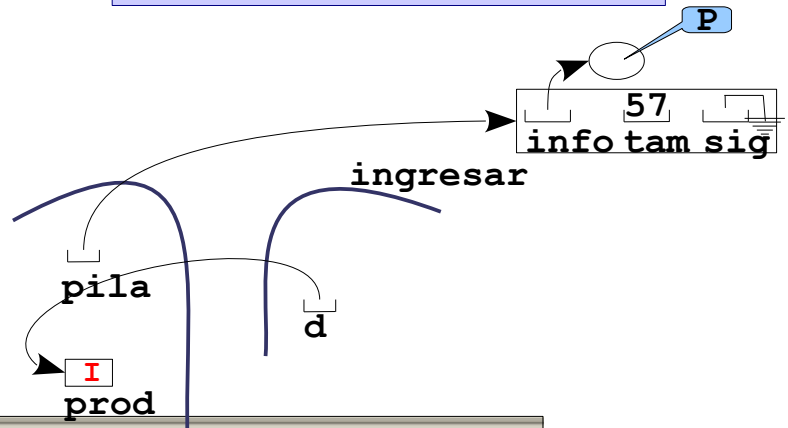
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



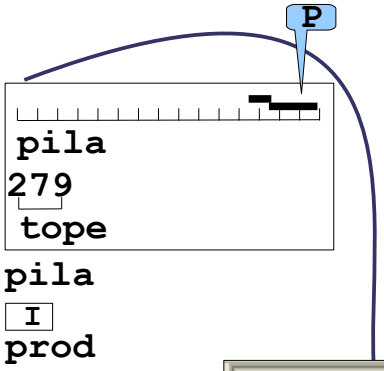
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

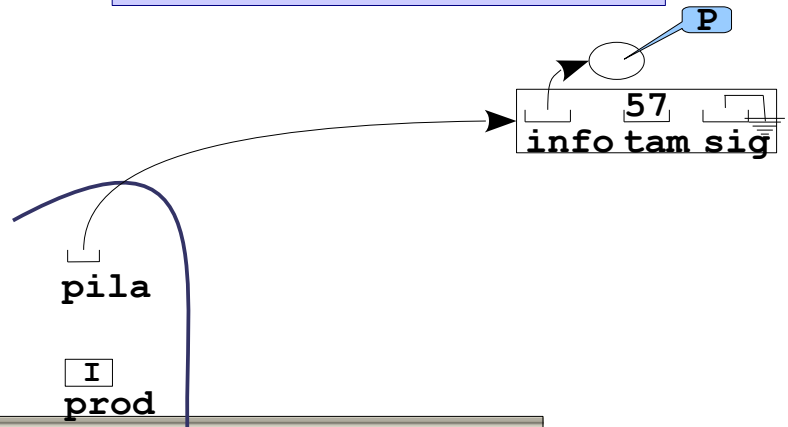
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

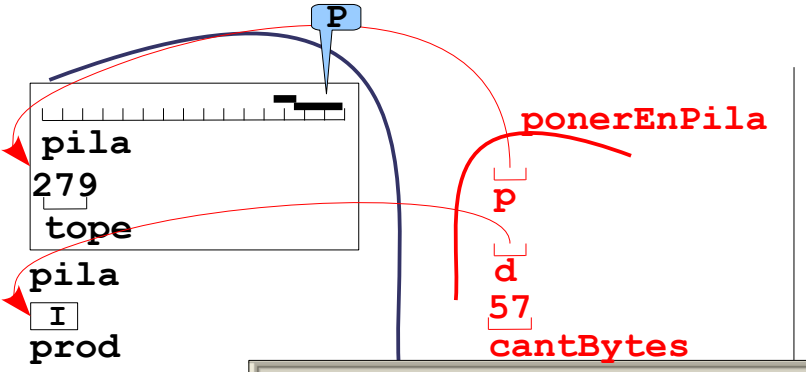



```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

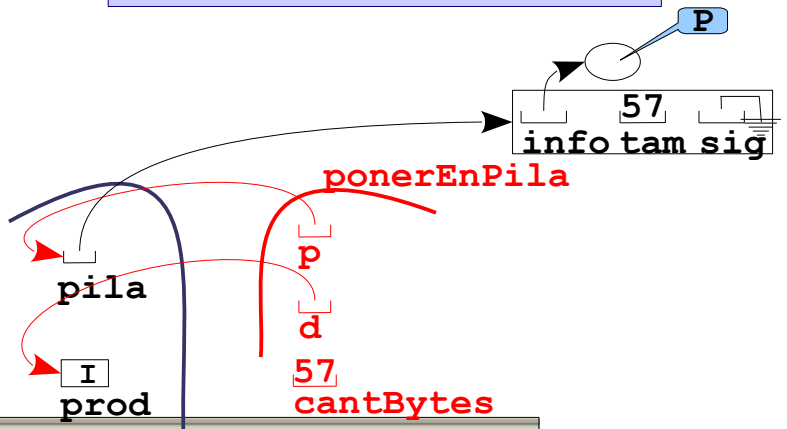
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



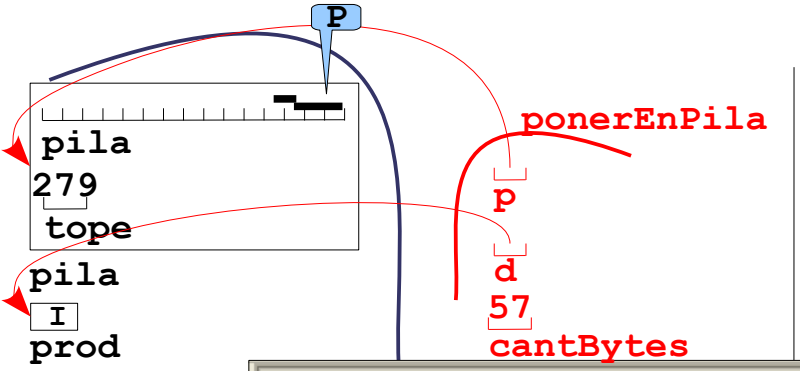
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

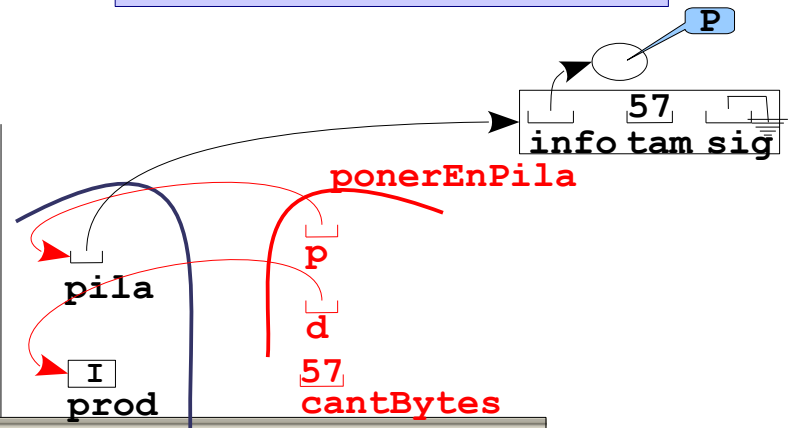
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
```

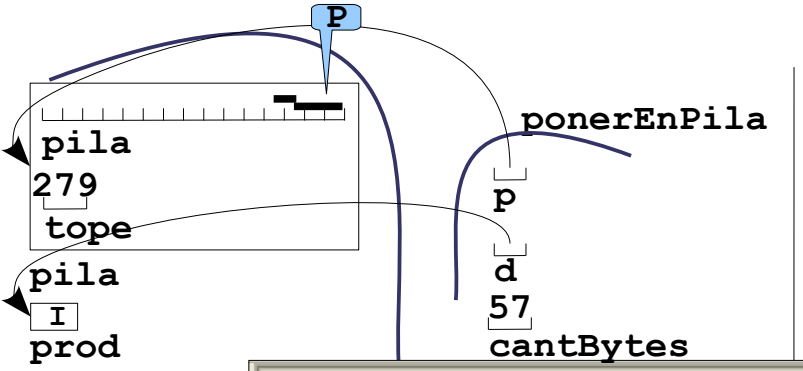
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

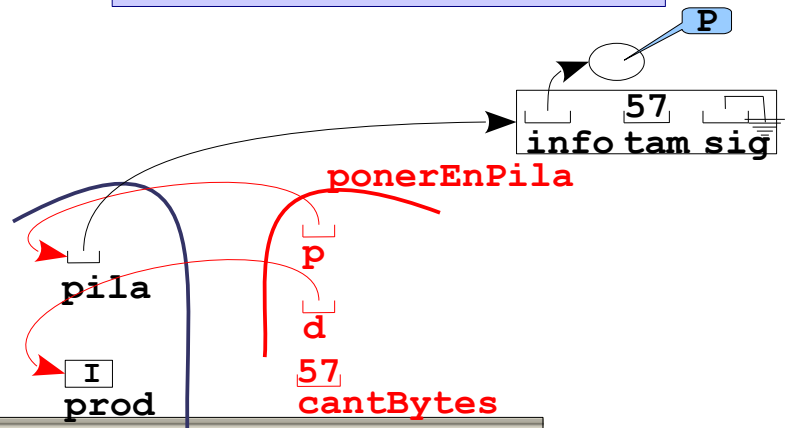
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



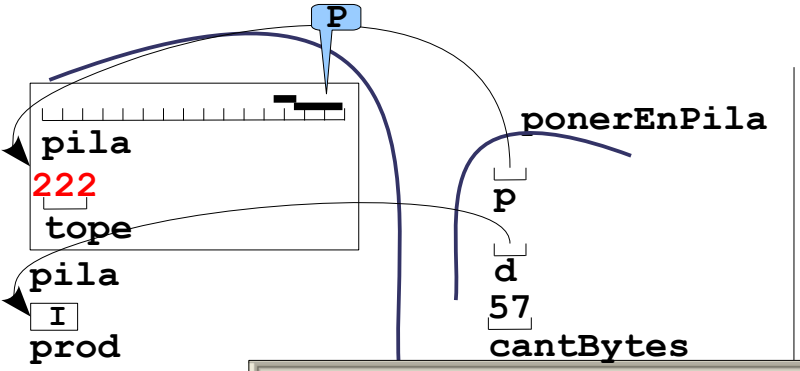

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
 86  void probarPonerYSacarDePila(void)
```

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

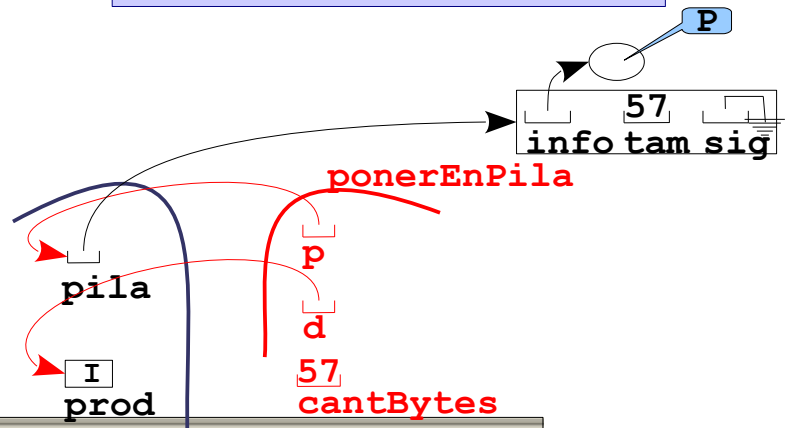
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
```

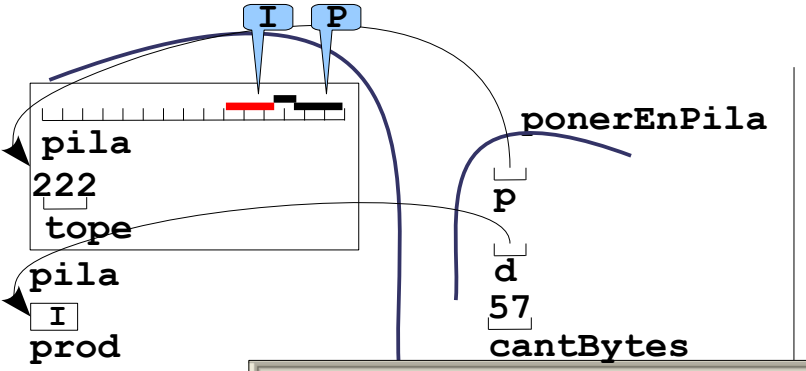
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

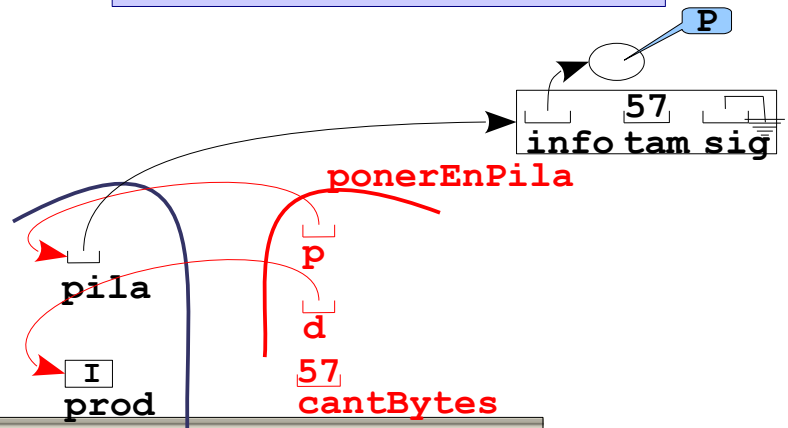
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
```

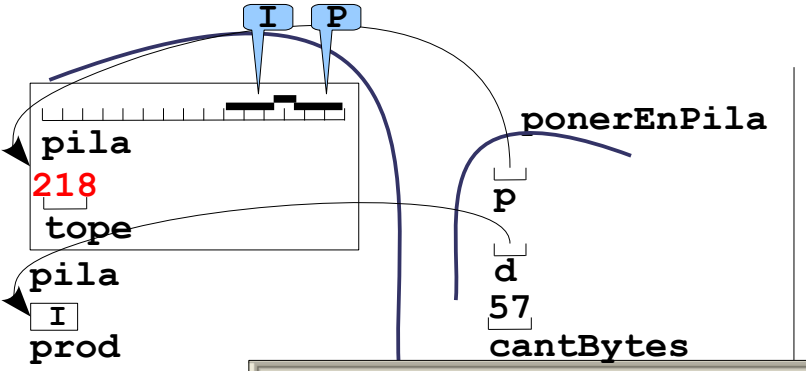
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

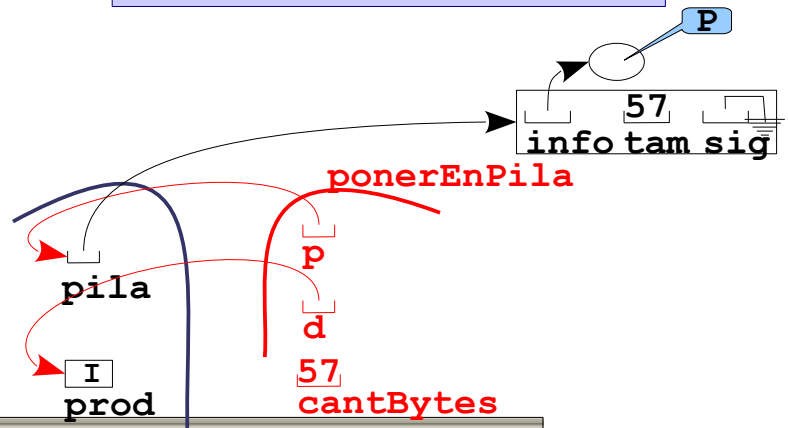
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
```

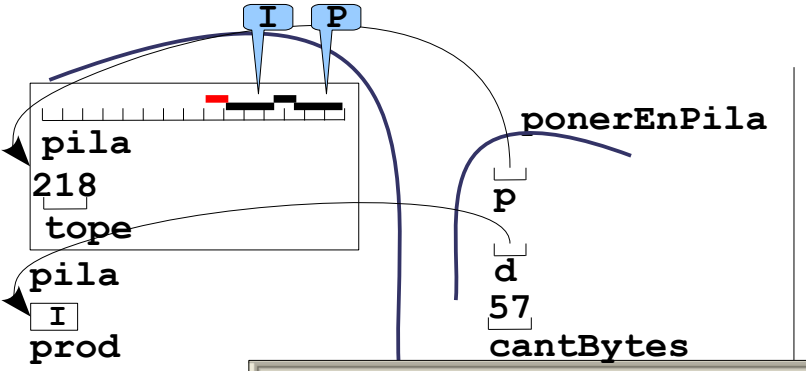
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

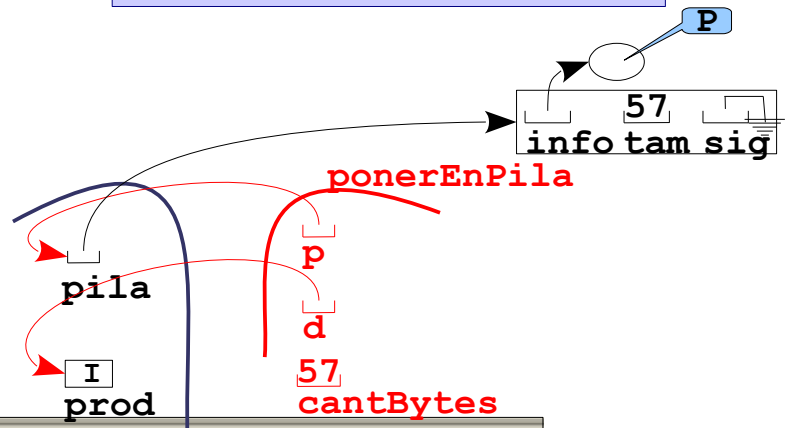
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
```

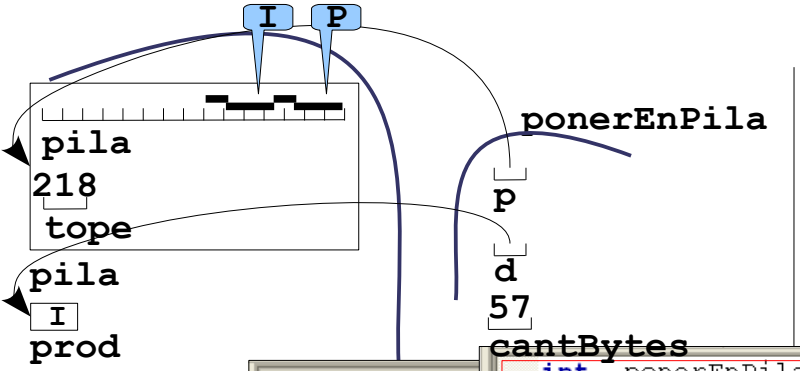
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

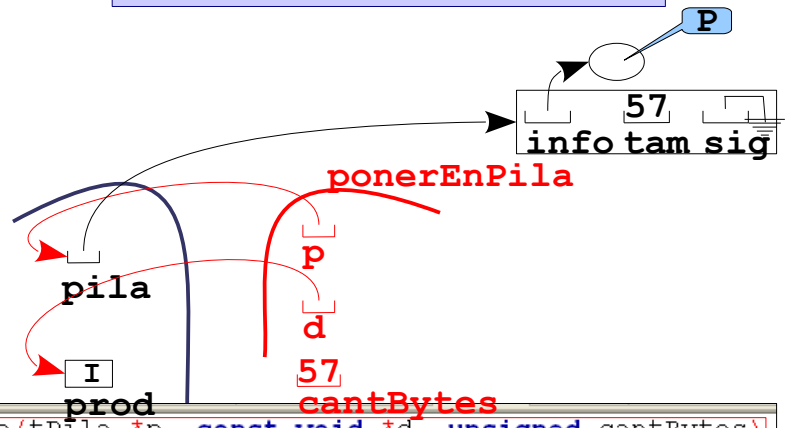
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

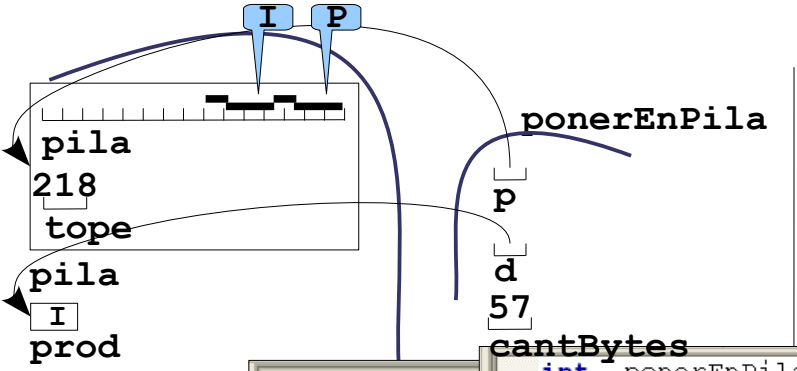



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

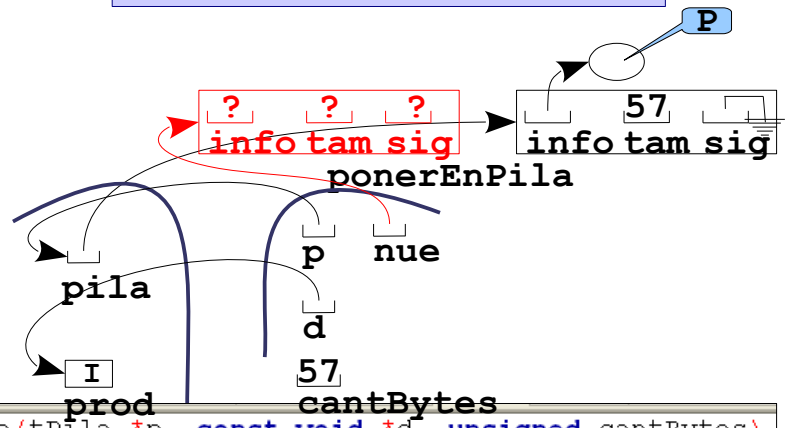
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

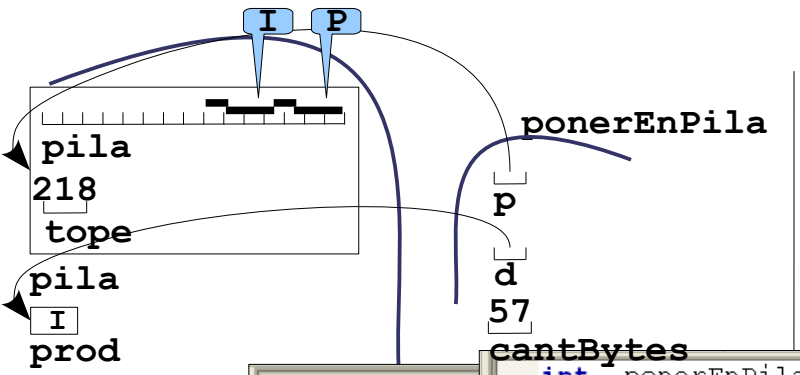



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

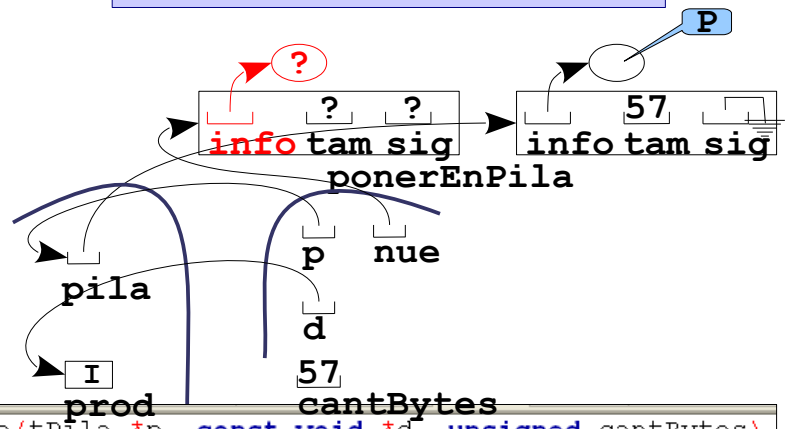
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



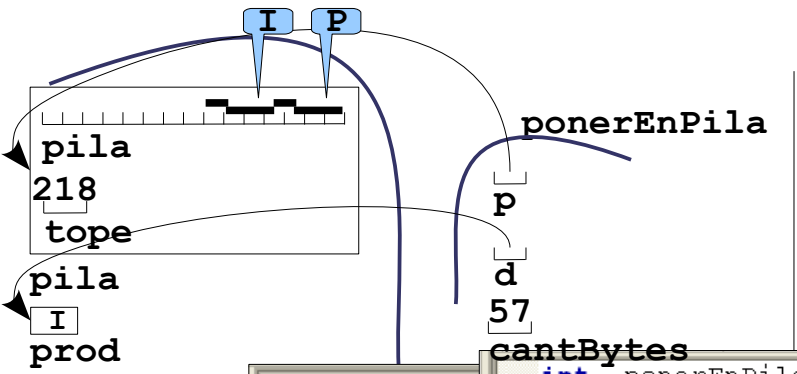
Implementación dinámica



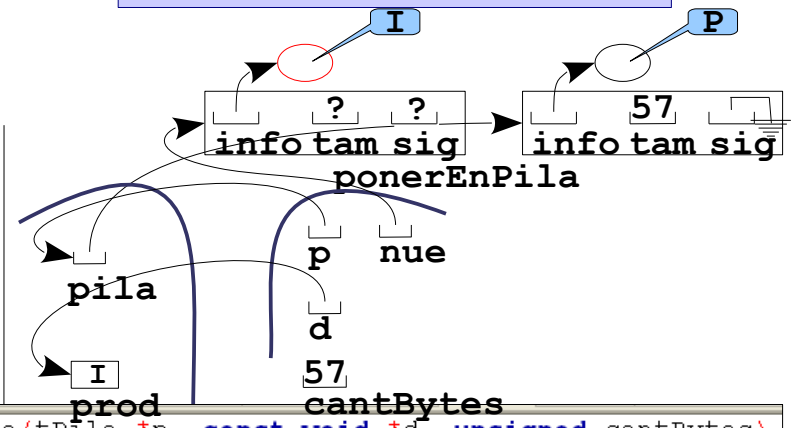

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica

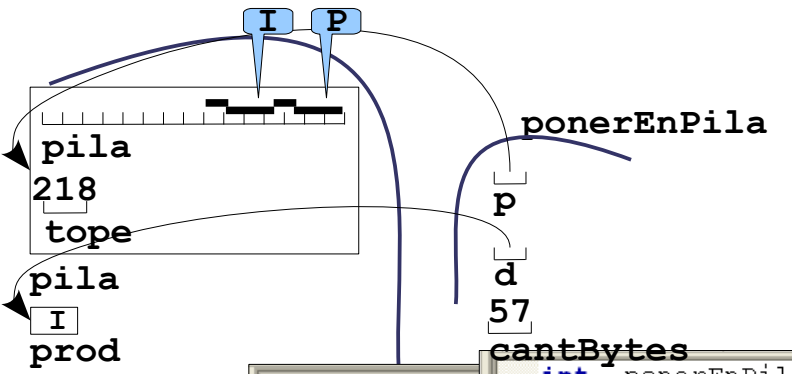



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

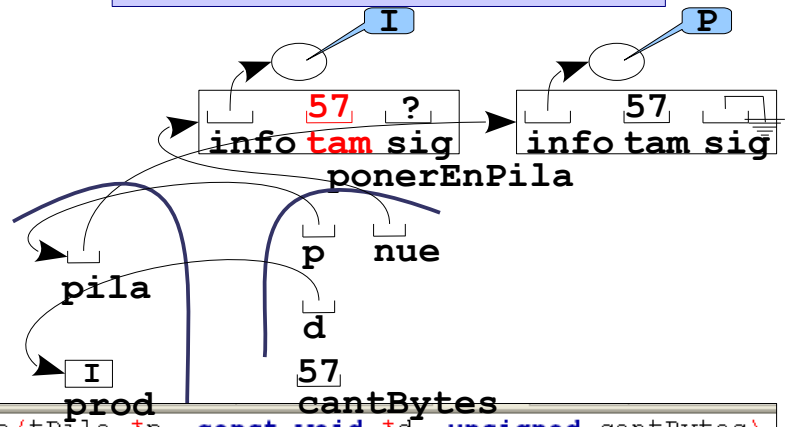
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

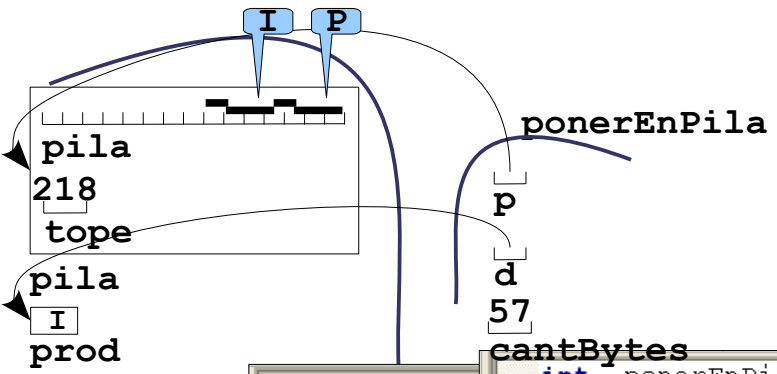



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

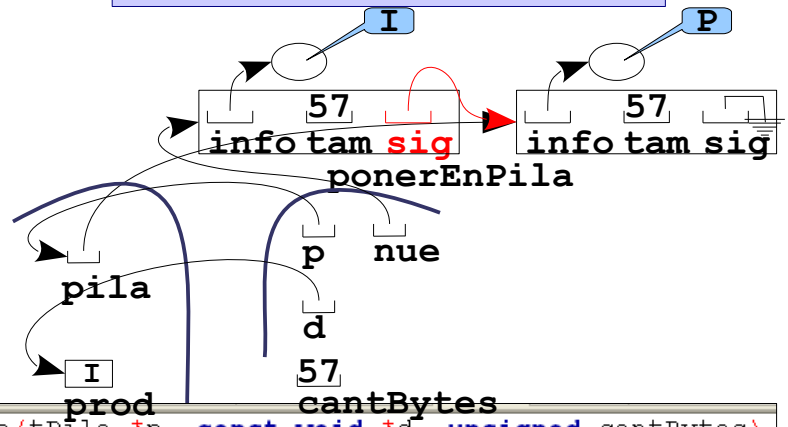
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c

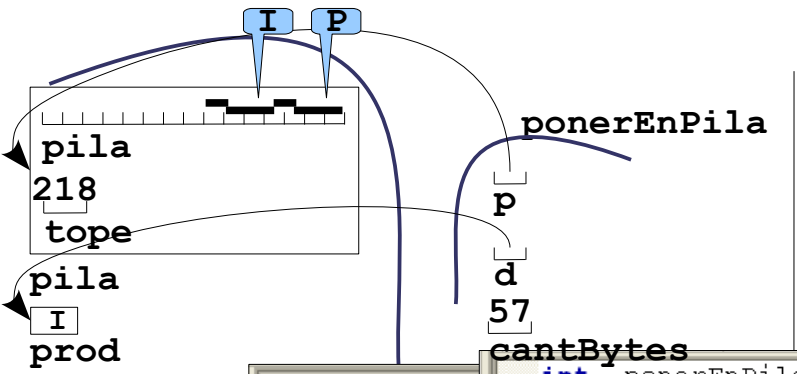
```
int  ponerEnPila(tPila *p
{
    if(p->tope < cantByte
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->t
    p->tope -= sizeof(uns
    memcpy(p->pila + p->t
    return 1;
}
```

main.c | main.h

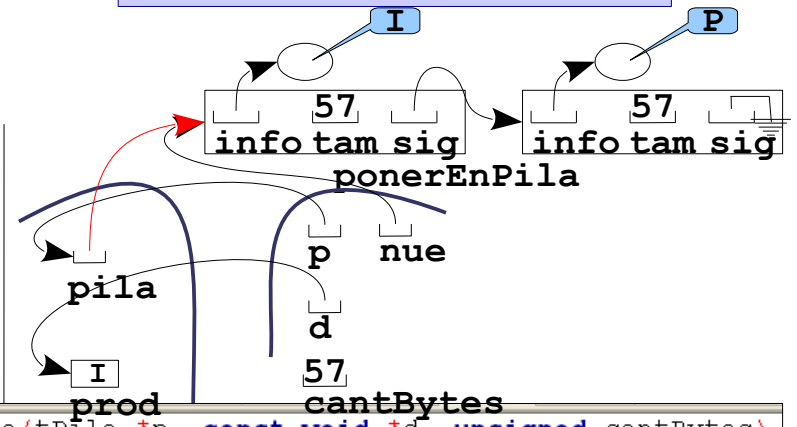
```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c

```
int  ponerEnPila(tPila *p
{
    if(p->tope < cantByte
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->t
    p->tope -= sizeof(uns
    memcpy(p->pila + p->t
    return 1;
}
```

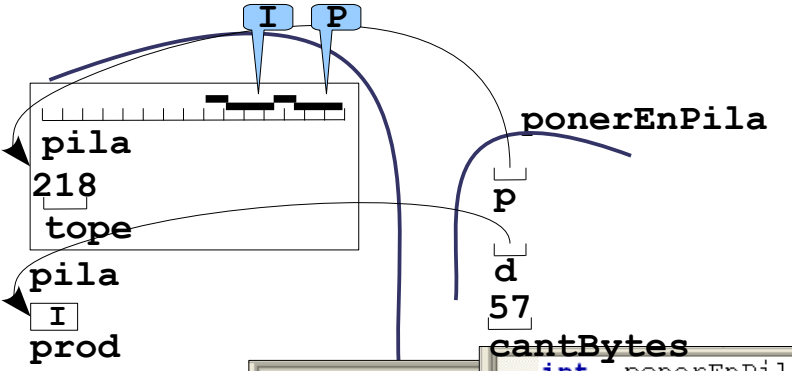
main.c | main.h

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

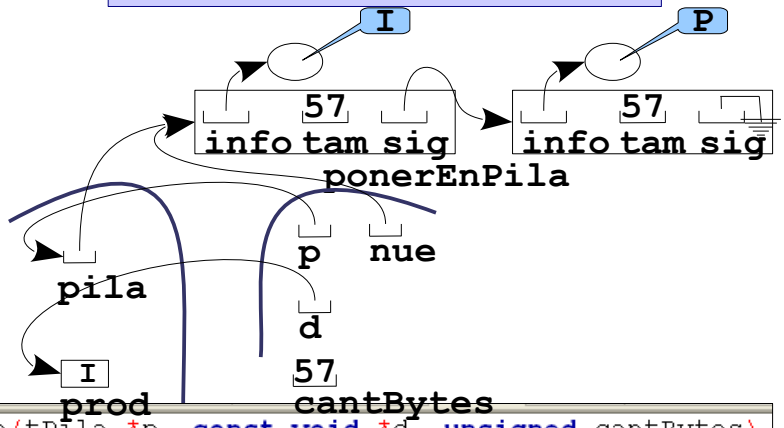
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



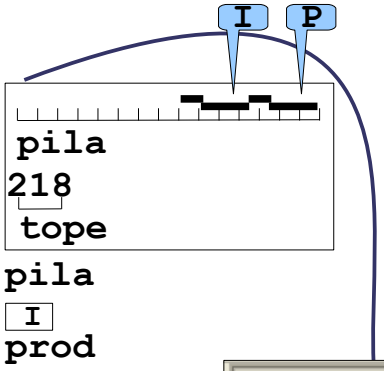

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

```
int  ponerEnPila(tPila *p
{
    if(p->tope < cantByte
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->t
    p->tope -= sizeof(uns
    memcpy(p->pila + p->t
    return 1;
}
```

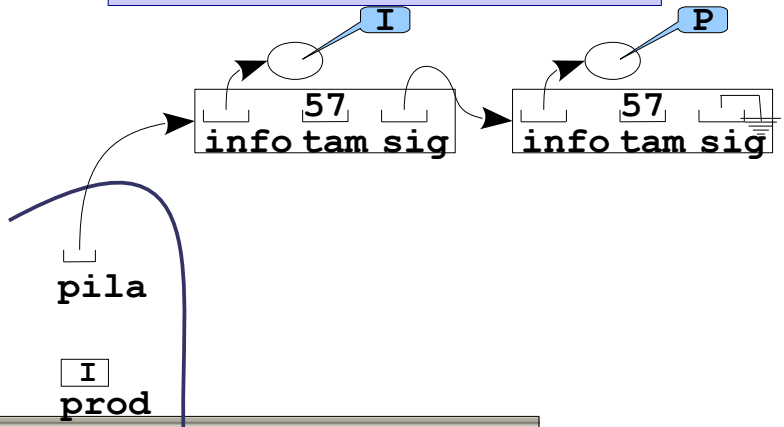
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



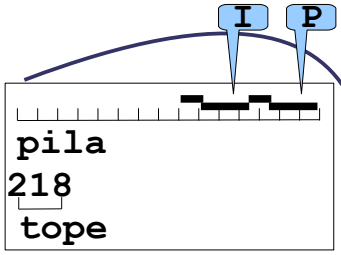
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

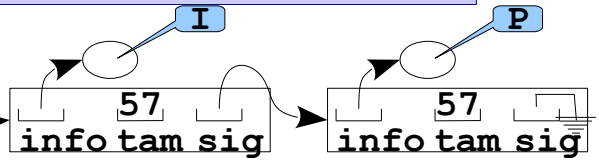


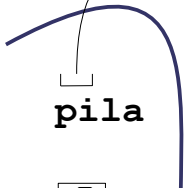
pila

I

prod

Implementación dinámica





I

prod

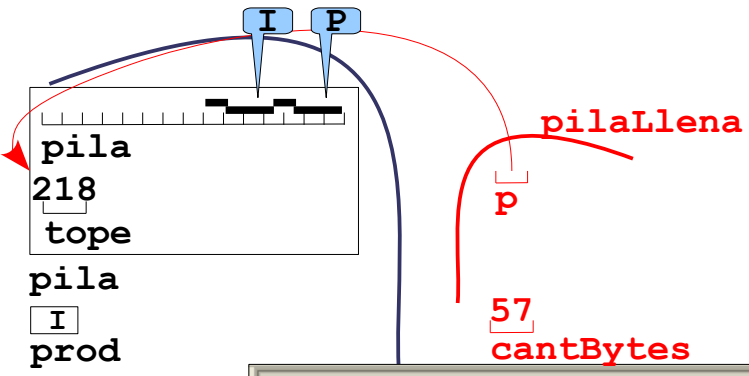
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

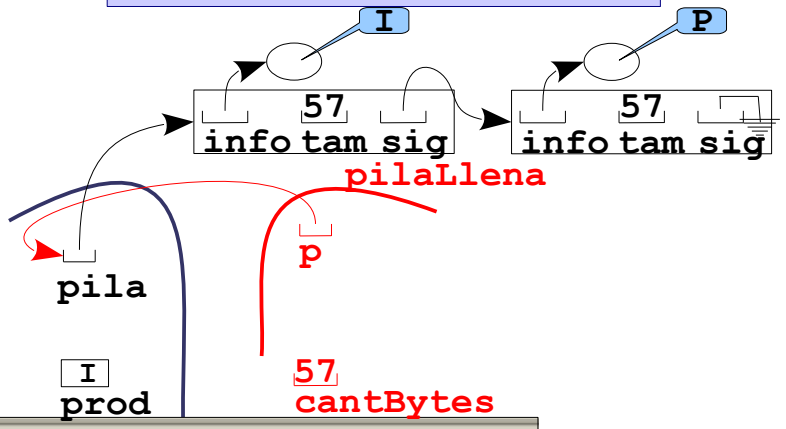
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



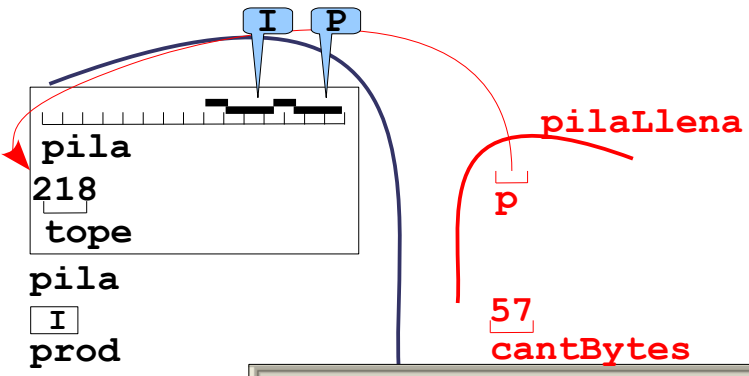
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

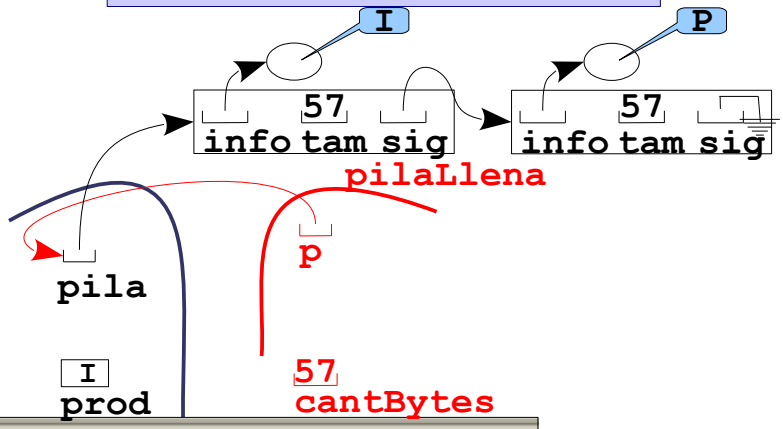
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

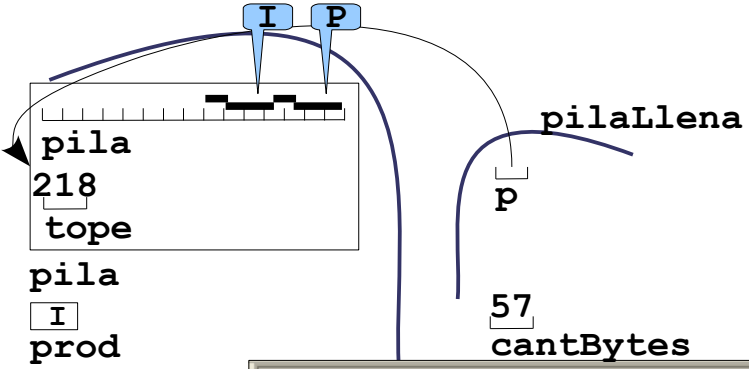
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

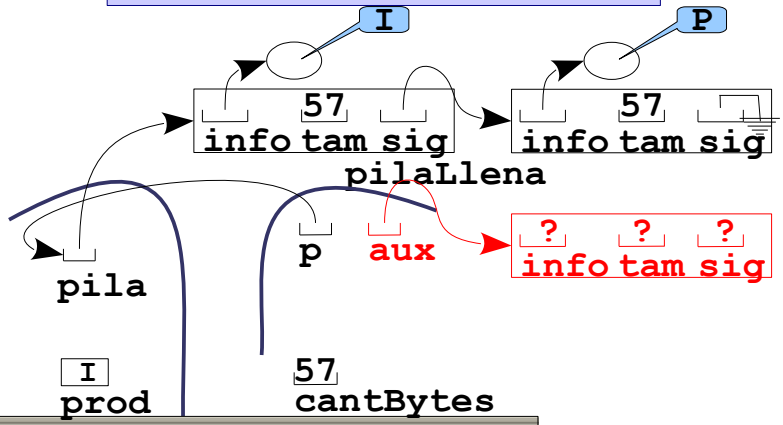
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

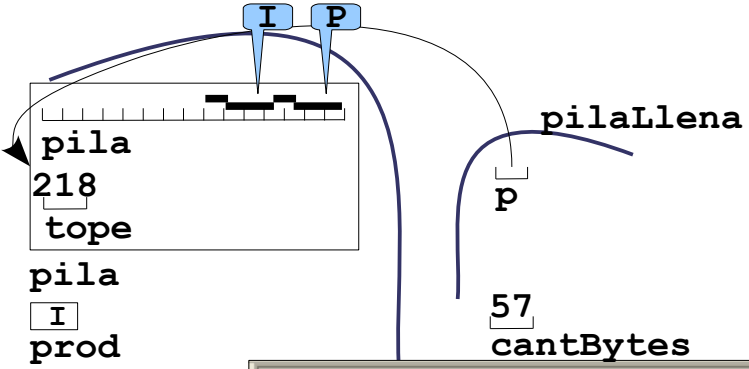
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

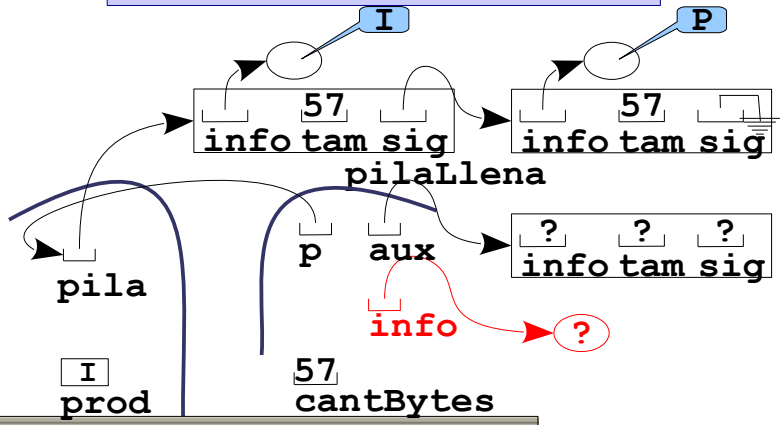
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

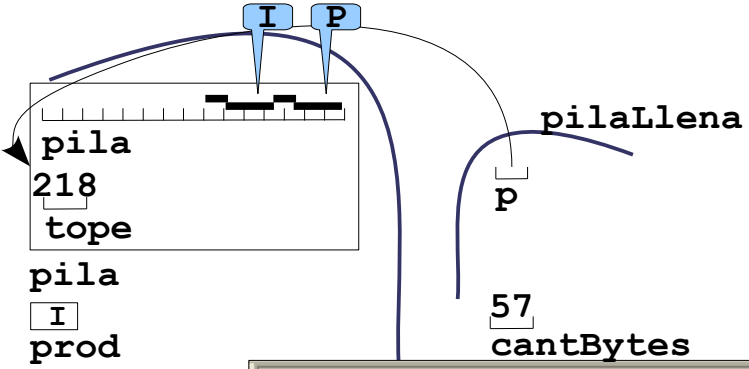
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

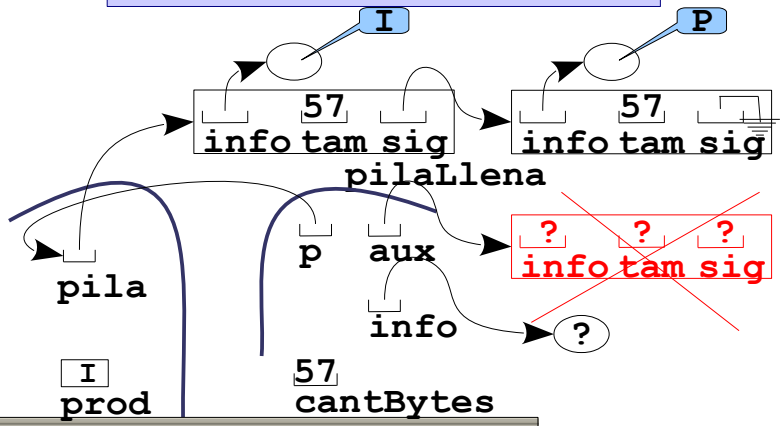
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

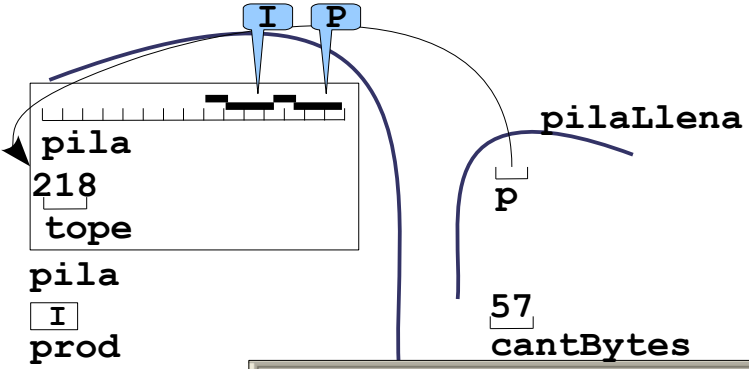
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

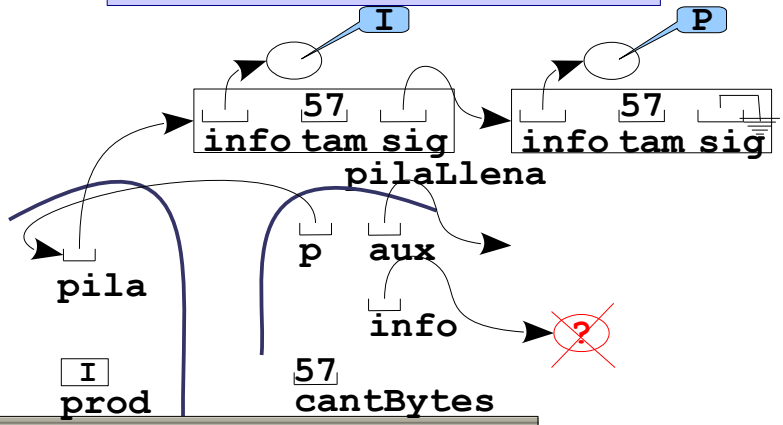
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

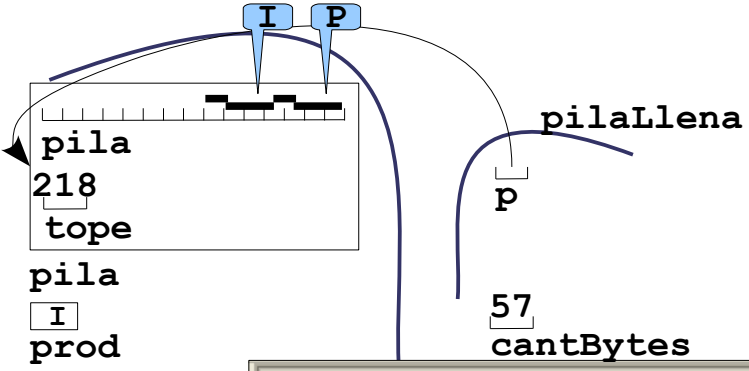
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

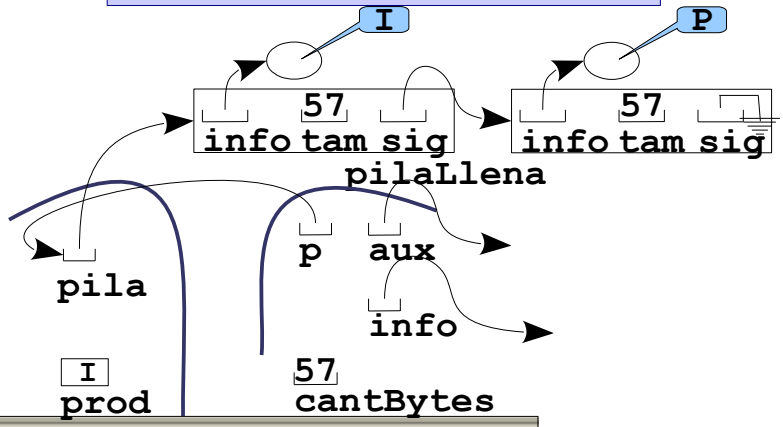
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
```

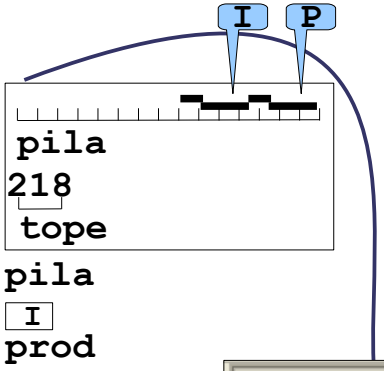
```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

14  int  pilaLlena(const tPila *p, unsigned cantBytes)
15  {
16      return p->tope < cantBytes + sizeof(unsigned);
17  }
```

```
main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

int  pilaLlena(const tPila *p, unsigned cantBytes)
{
    tNodo  *aux = (tNodo *)malloc(sizeof(tNodo));
    void   *info = malloc(cantBytes);

    free(aux);
    free(info);
20  return aux == NULL || info == NULL;
21  }
22
```

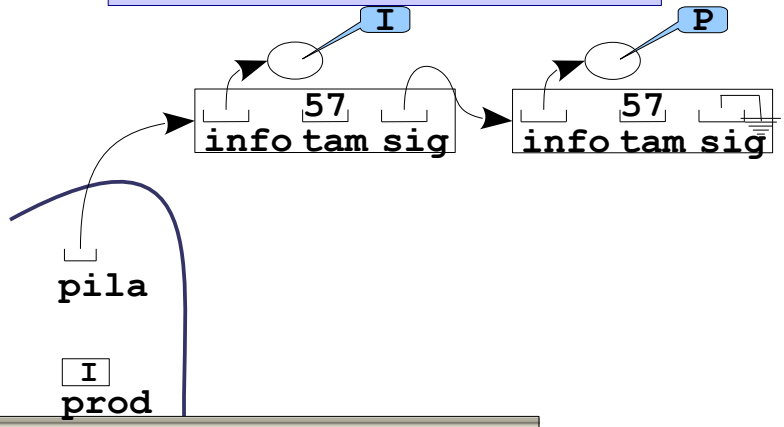
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



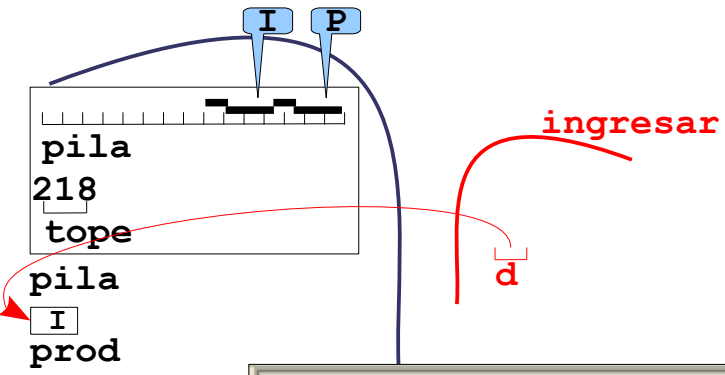
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

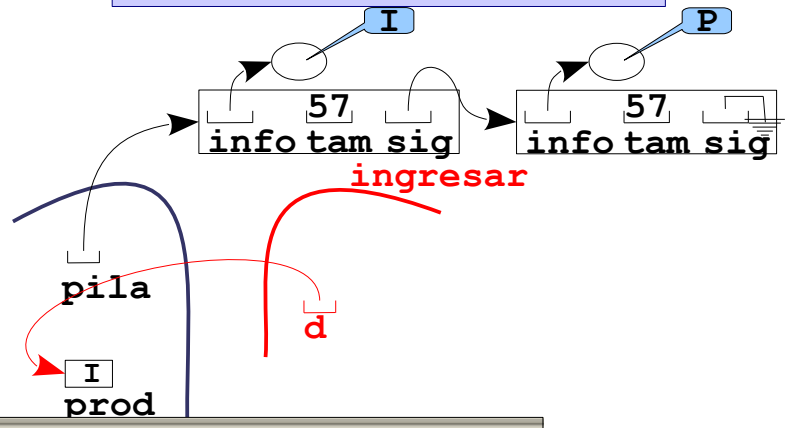
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



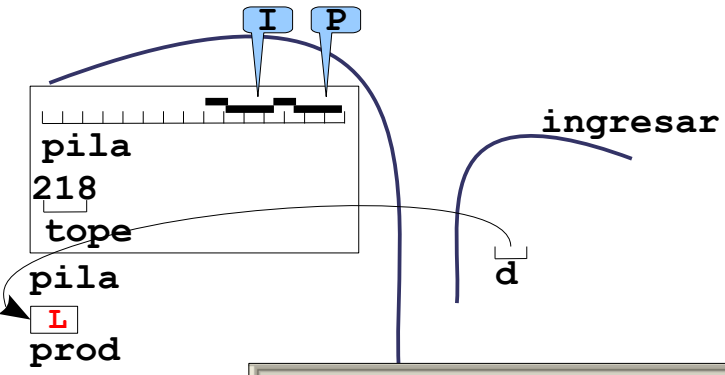
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

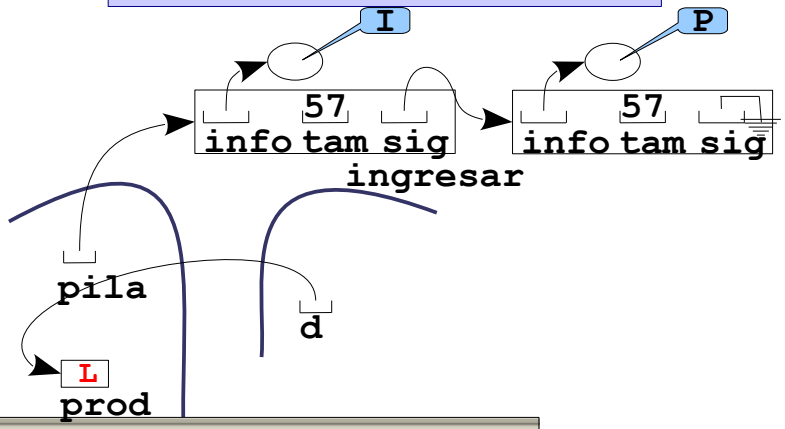
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



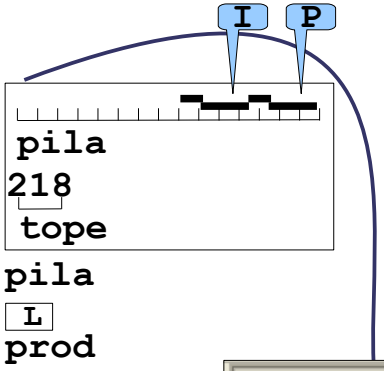
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

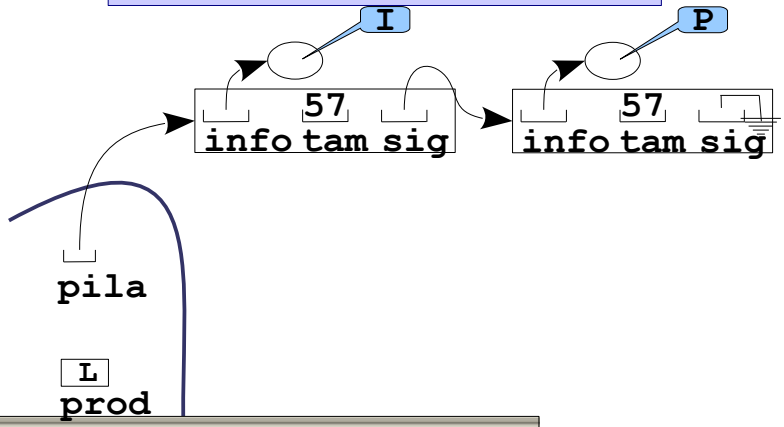
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



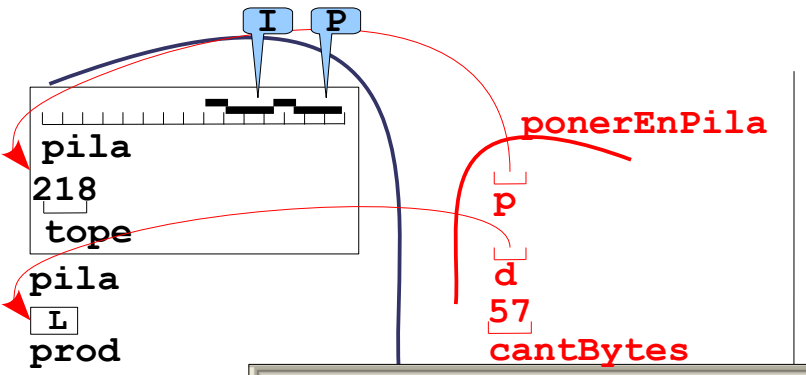
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

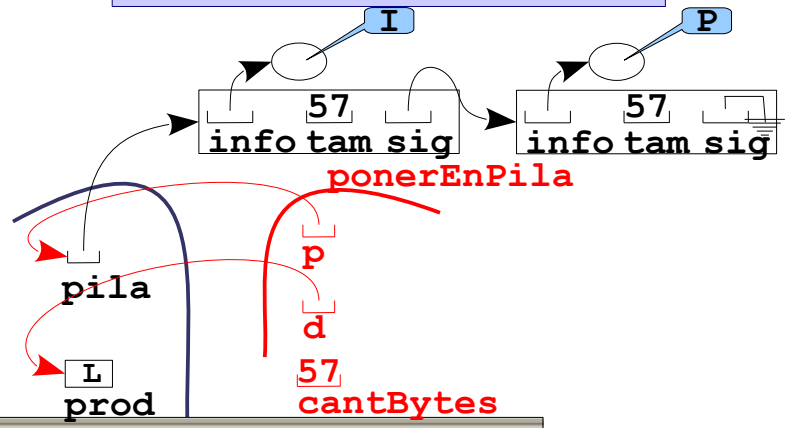
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



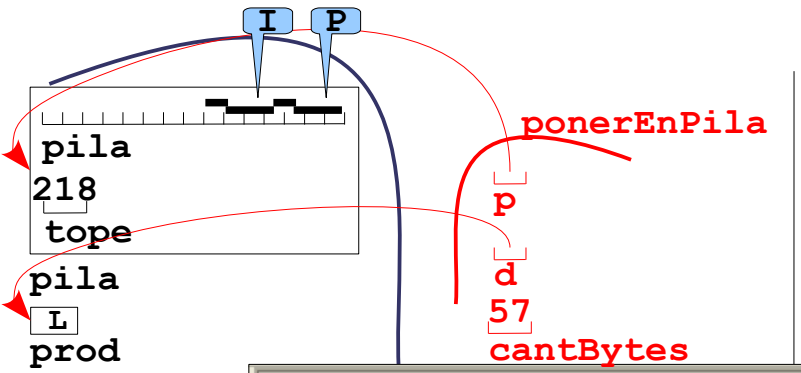
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

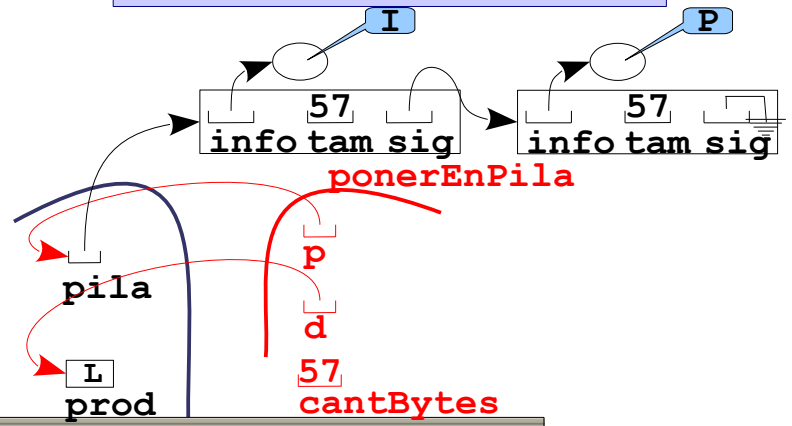
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
```

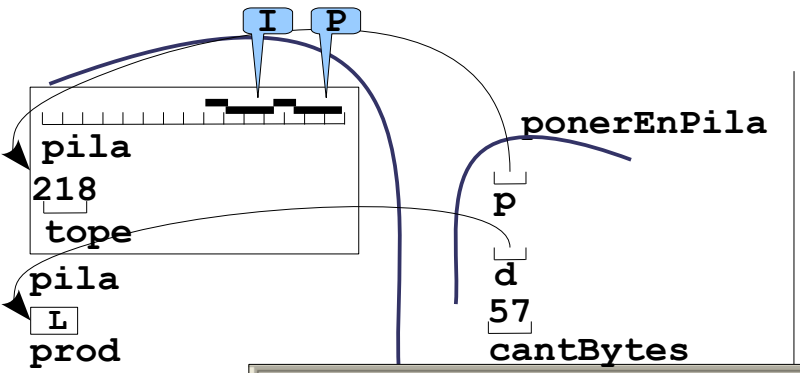
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

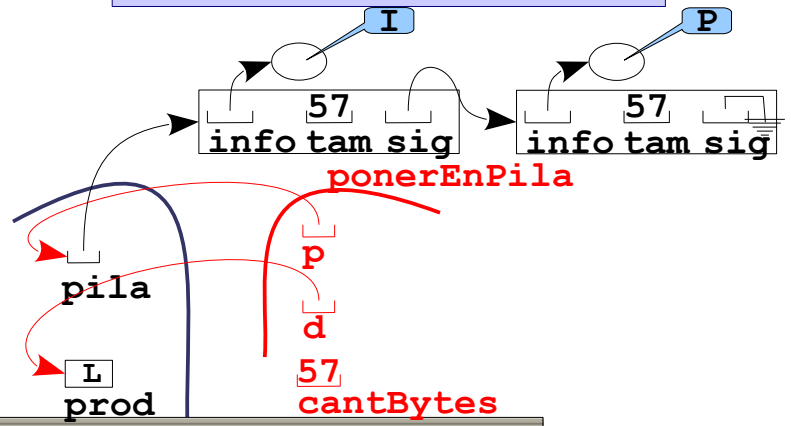
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

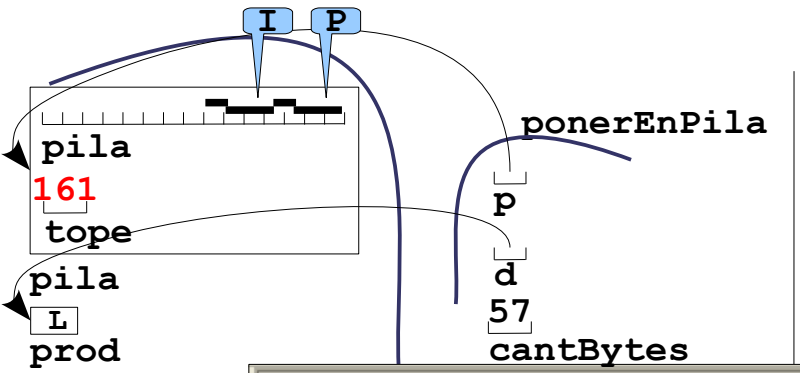
```
86  void probarPonerYSacarDePila(void)
```

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

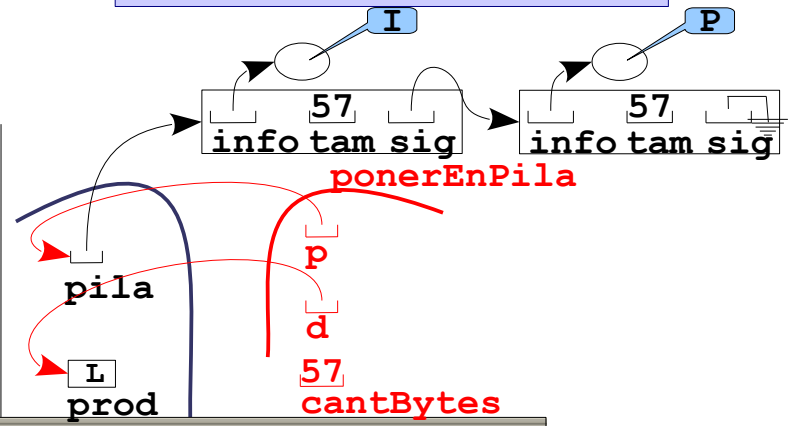
```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

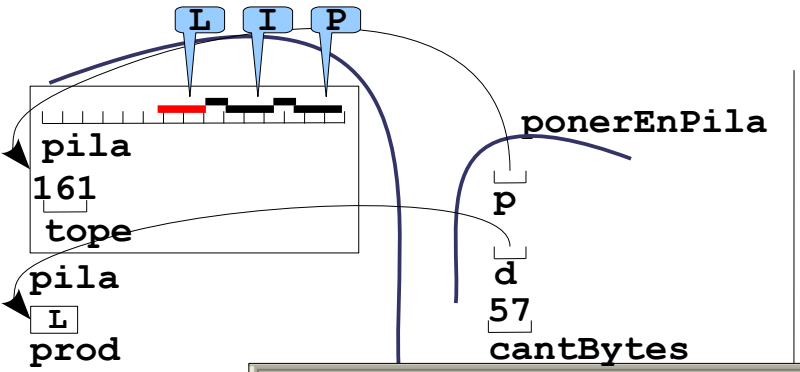
```
86  void probarPonerYSacarDePila(void)
```

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

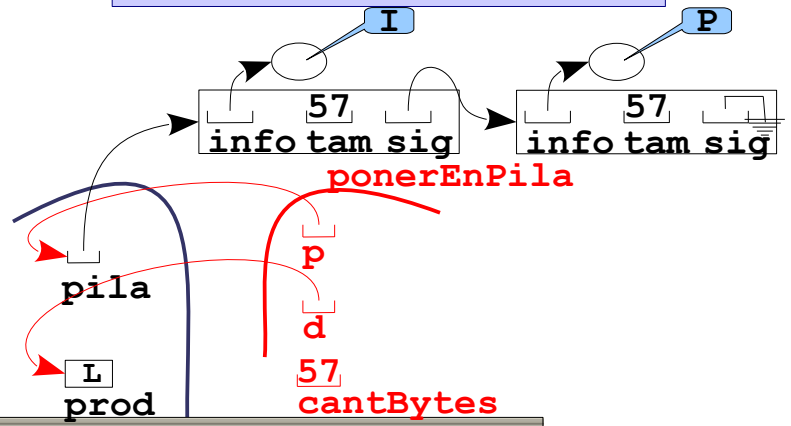
```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica



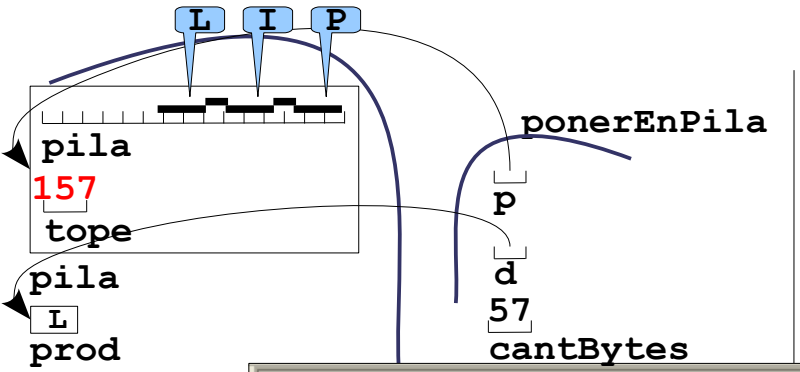

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
86  void probarPonerYSacarDePila(void)
```

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

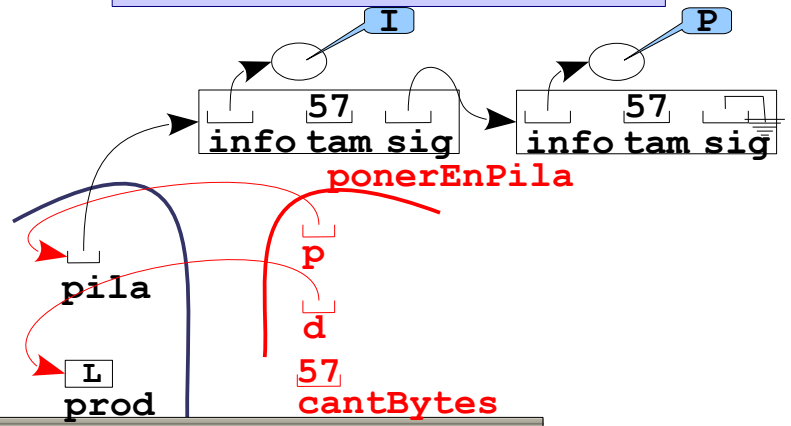
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
```

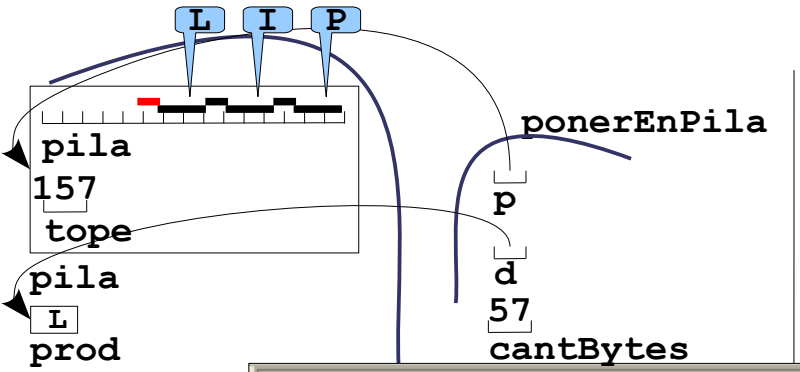
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

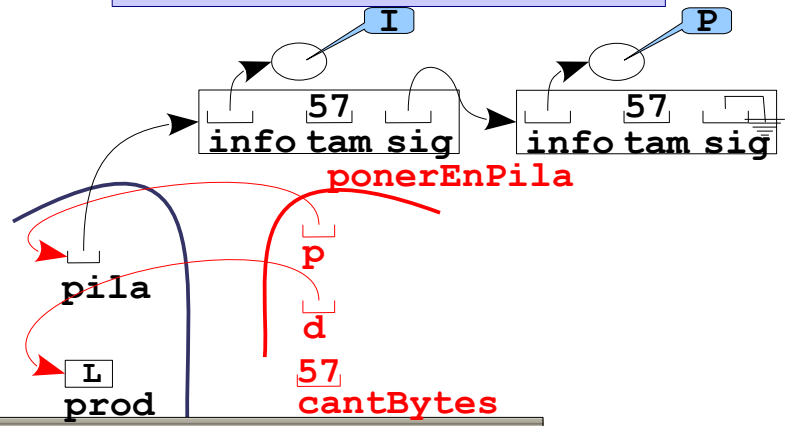
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

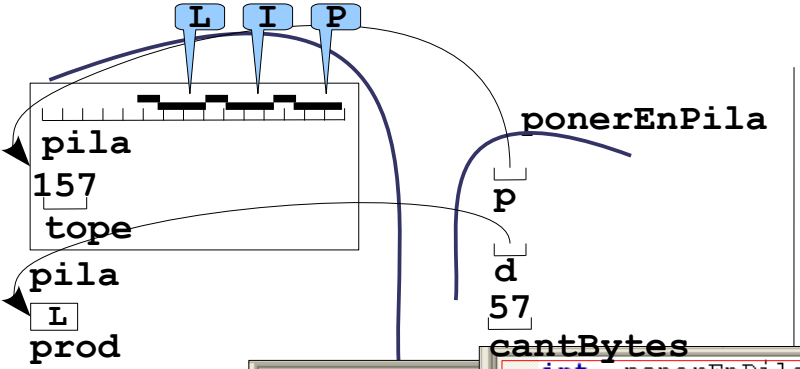
```
86  void probarPonerYSacarDePila(void)
```

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

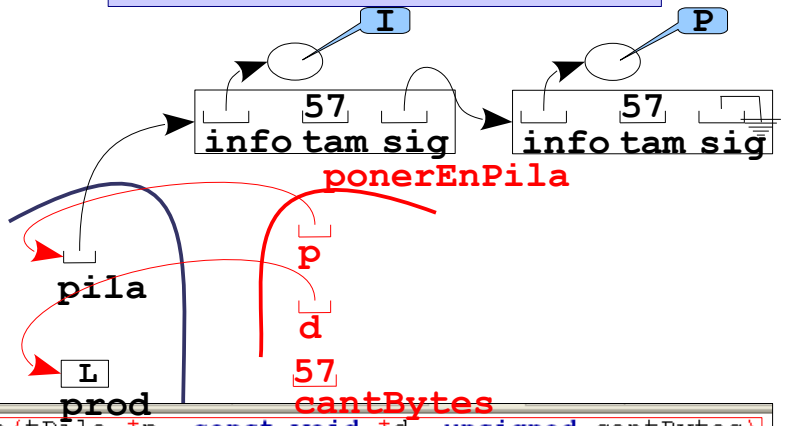
```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);
    memcpy(p->pila + p->tope, &cantBytes, sizeof(unsigned));
    return 1;
}
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica

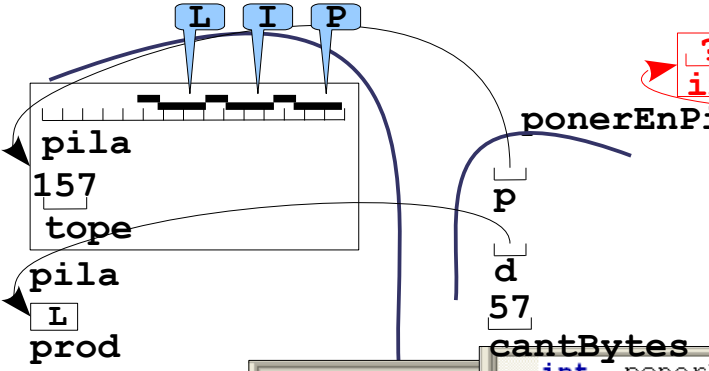



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

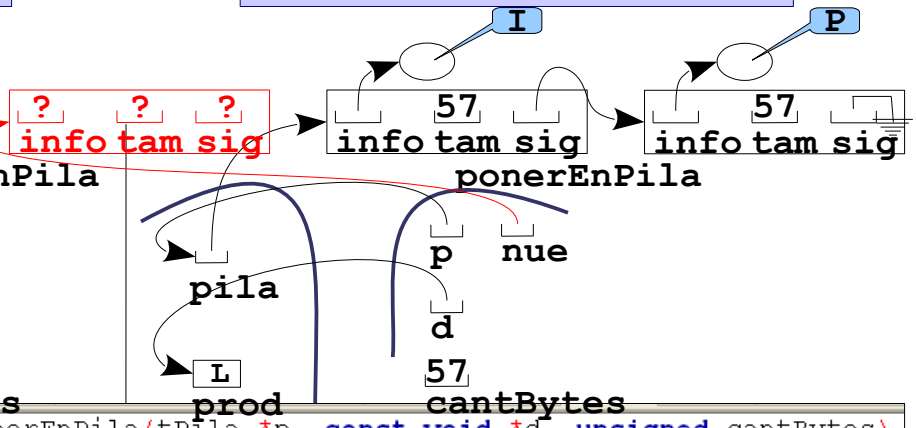
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

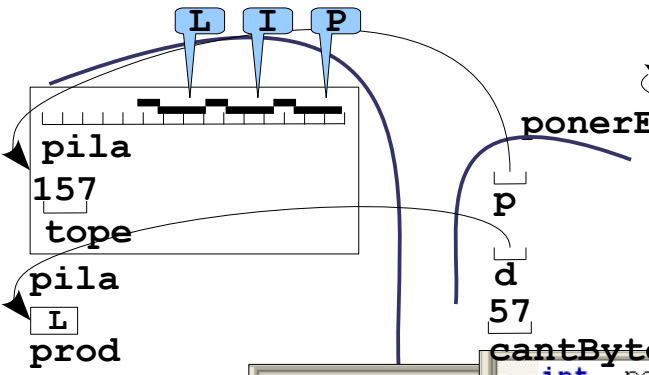



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

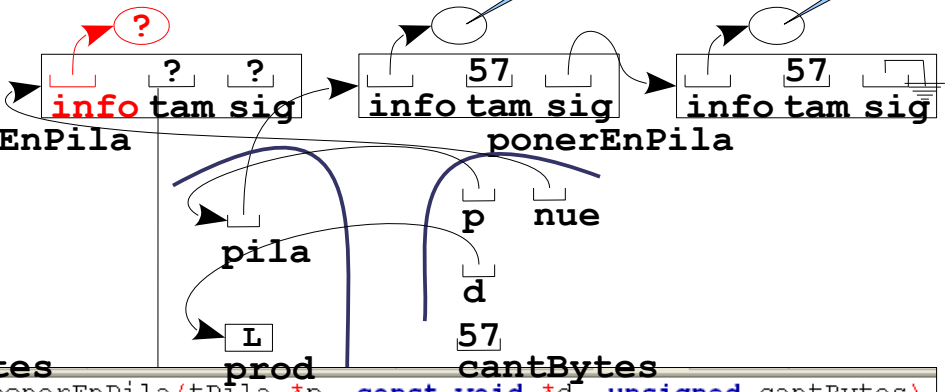
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

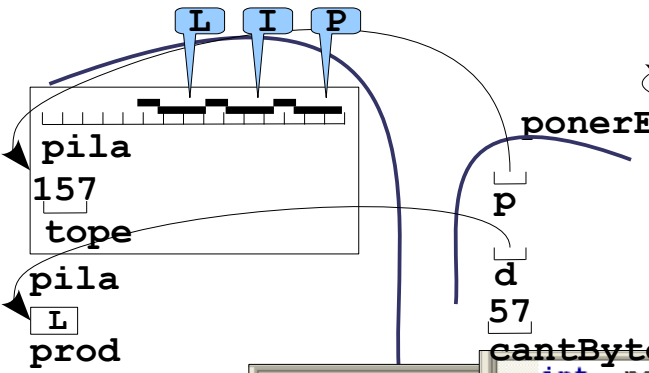



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

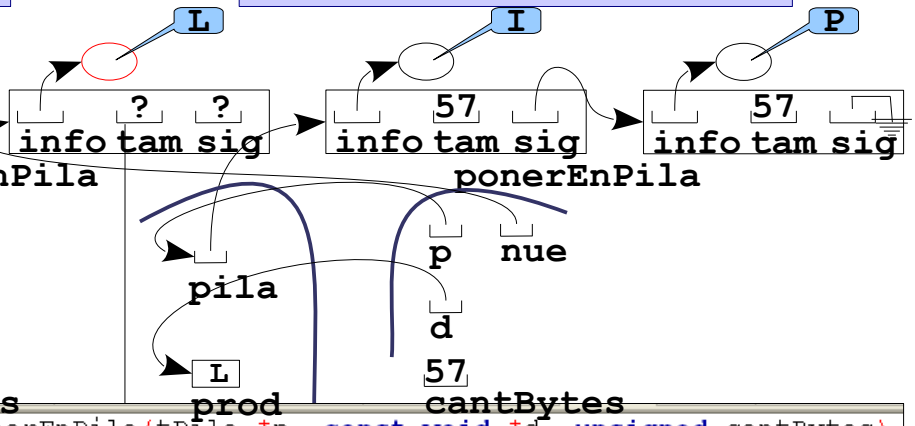
```
int  ponerEnPila(tPila *p
{
    if(p->tope < cantByte
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->t
    p->tope -= sizeof(uns
    memcpy(p->pila + p->t
    return 1;
}
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica

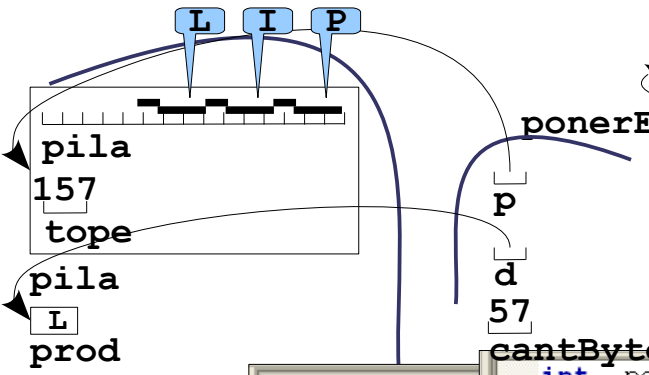



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

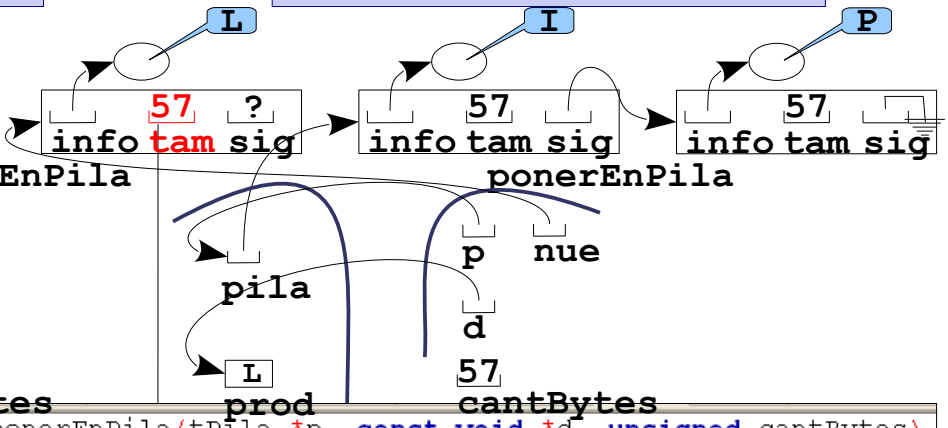
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



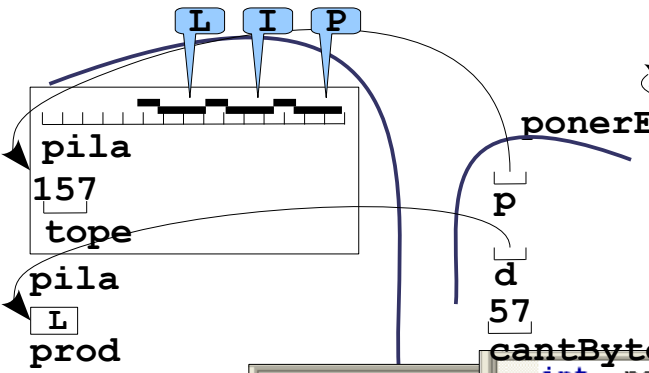
Implementación dinámica



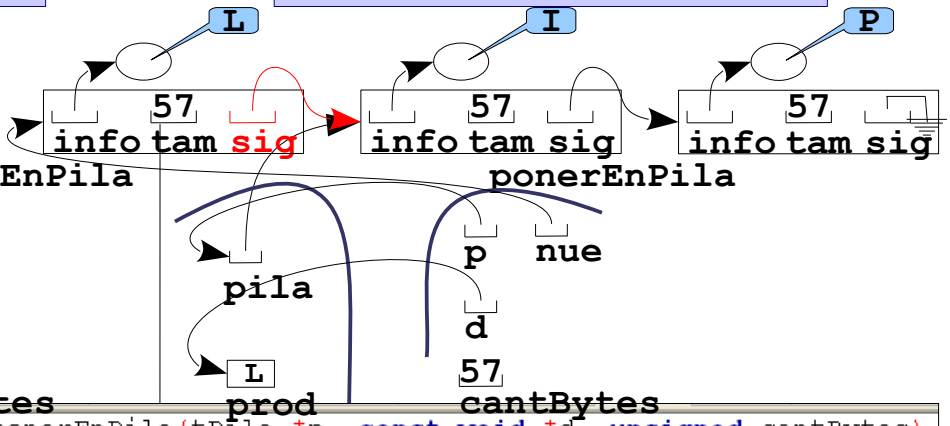

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica

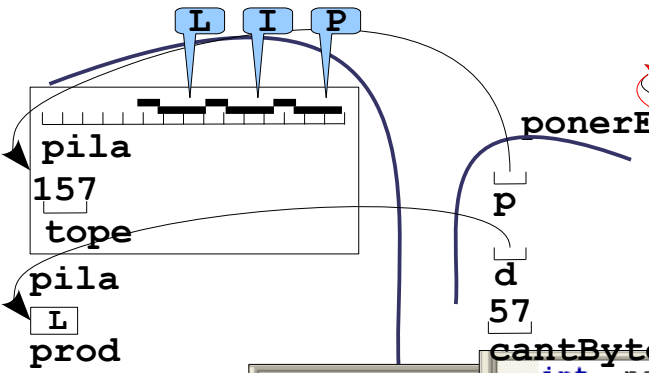



```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

```
int  ponerEnPila(tPila *p
{
    if(p->tope < cantByte
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->t
    p->tope -= sizeof(uns
    memcpy(p->pila + p->t
    return 1;
}
```

Tipo de dato Pila.

Implementación estática



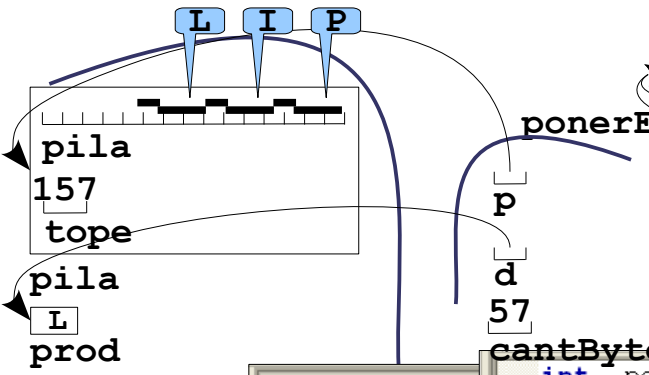
Implementación dinámica


```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

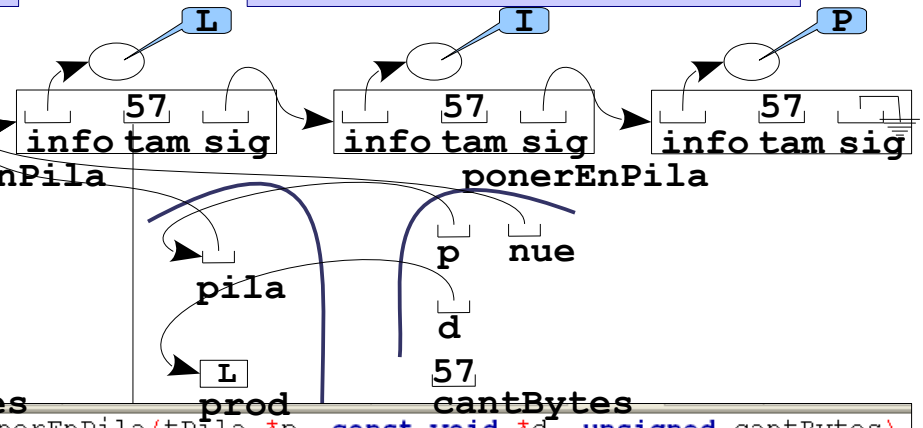
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



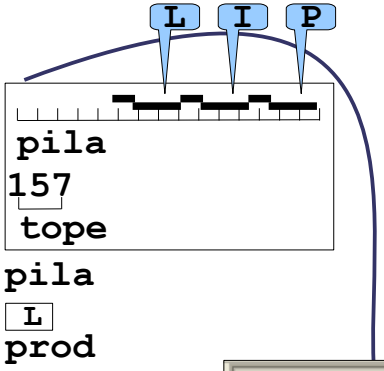

```
int  ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

```
int  ponerEnPila(tPila *p
{
    if(p->tope < cantByte
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->t
    p->tope -= sizeof(uns
    memcpy(p->pila + p->t
    return 1;
}
```

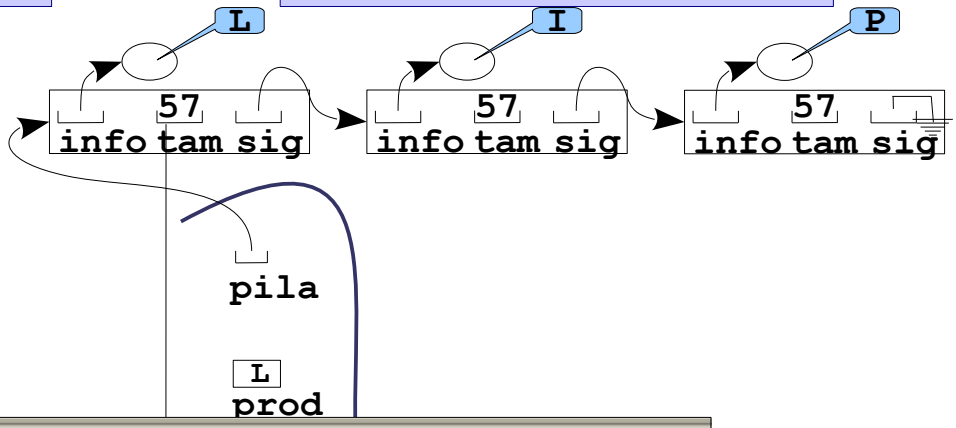
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica



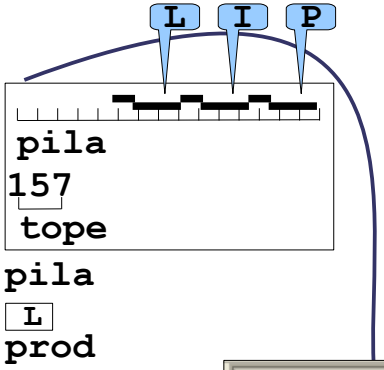
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

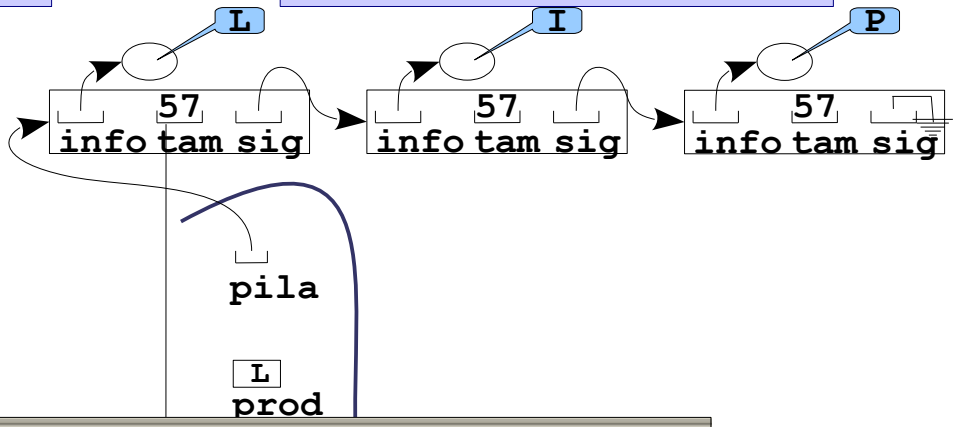
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

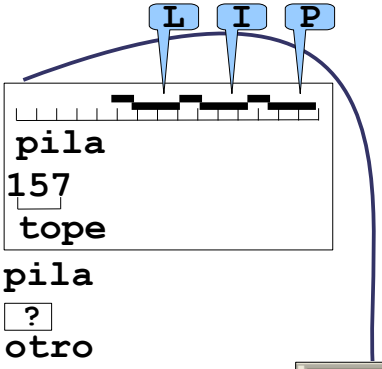


main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

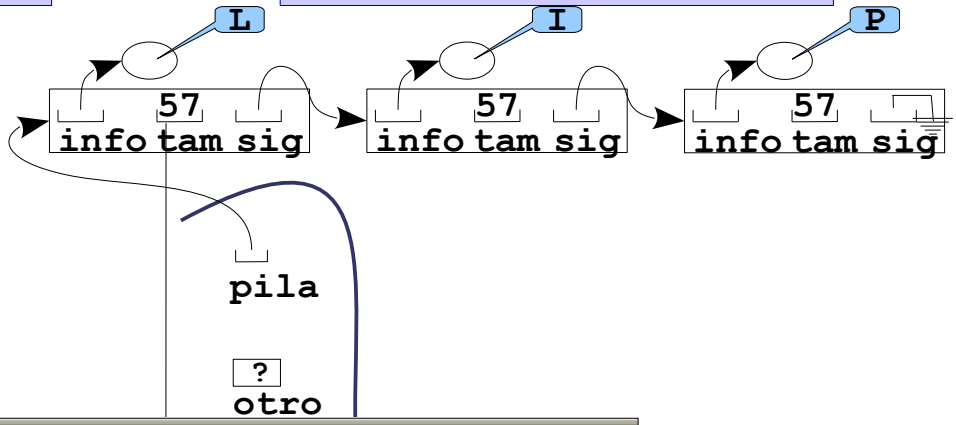
```
86  void probarPonerYSacarDePila(void)
87  {
88      tProd   prod;
89      tPila   pila;
90
91      crearPila(&pila);
92      while(!pilaLlena(&pila, sizeof(prod)) &&
93            ingresarProducto(&prod))
94          if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95              puts("ERROR - inesperado: pila llena\n");
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica

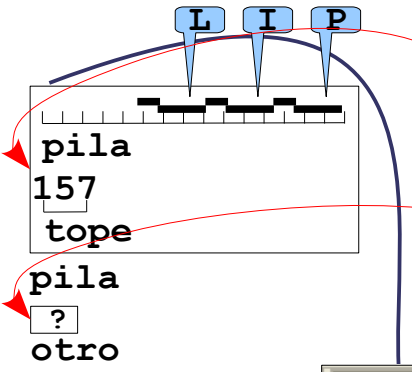


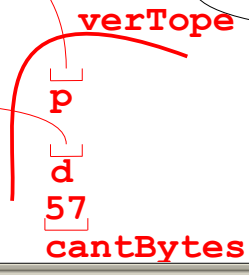
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
107     puts("Ver tope de la pila:");
108     if(verTope(&pila, &otro, sizeof(otro)))
109         mostrarProducto(&otro);
110     else
111         puts("La pila estaba vacia.");
```

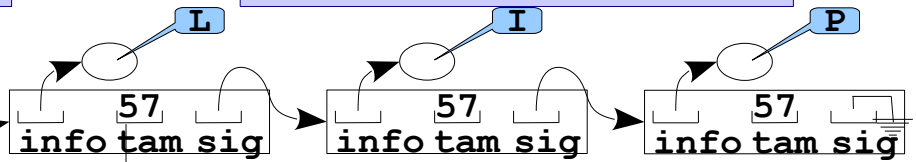
Tipo de dato Pila.

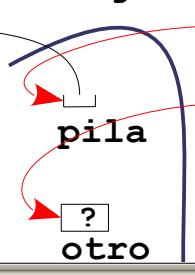
Implementación estática

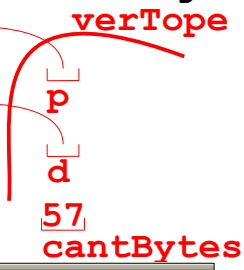




Implementación dinámica



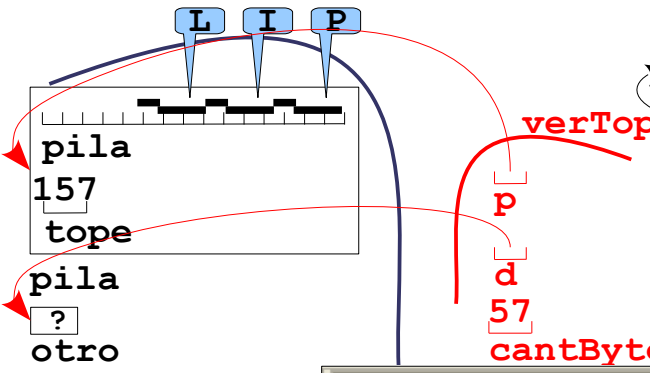




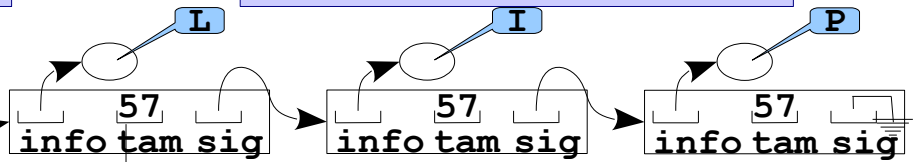

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
107     puts("Ver tope de la pila:");
108     if(verTope(&pila, &otro, sizeof(otro)))
109         mostrarProducto(&otro);
110     else
111         puts("La pila estaba vacia.");
```

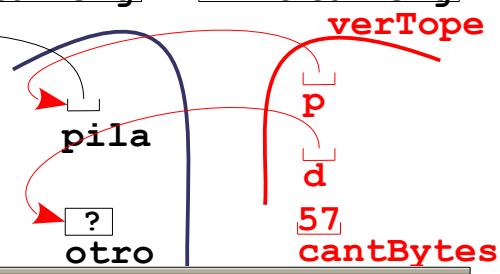
Tipo de dato Pila.

Implementación estática



Implementación dinámica





main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

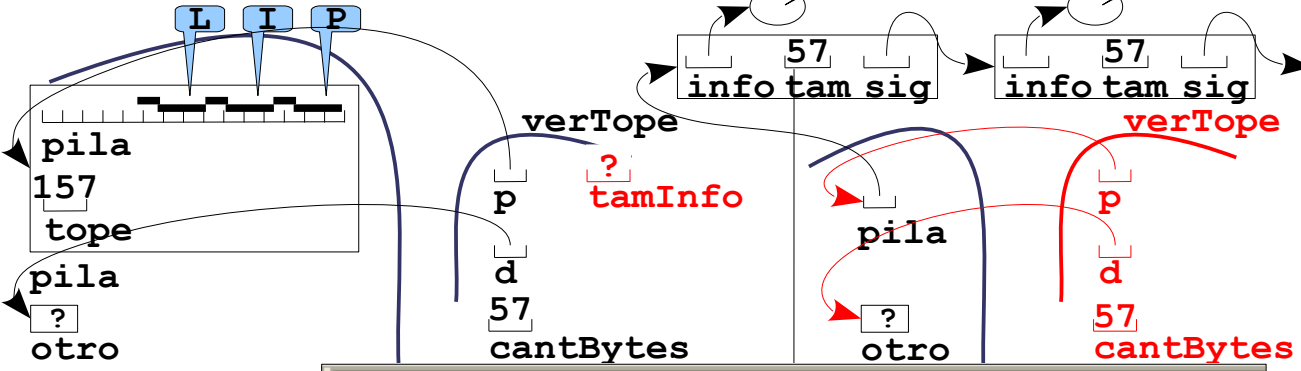
```
107     puts("Ver tope de la pila:");
```

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
30  int  verTope(const tPila *p, void *d, unsigned cantBytes)
31  {
32      unsigned tamInfo;
33
34      if(p->tope == TAM_PILA)
35          return 0;
36      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
37      memcpy(d, p->pila + p->tope + sizeof(unsigned),
38             minimo(cantBytes, tamInfo));
39      return 1;
40  }
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

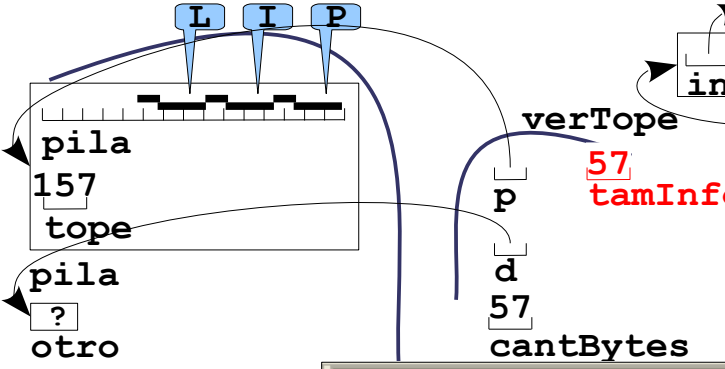
```
107     puts("Ver tope de la pila:");
```

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

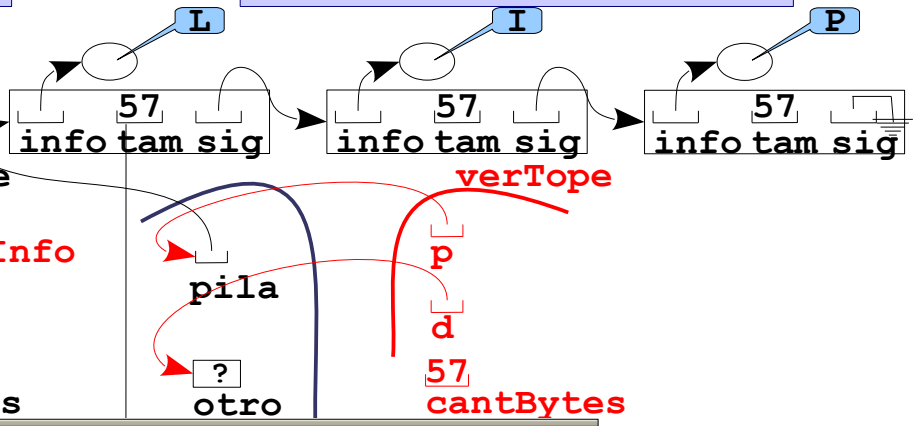
```
30  int  verTope(const tPila *p, void *d, unsigned cantBytes)
31  {
32      unsigned tamInfo;
33
34      if(p->tope == TAM_PILA)
35          return 0;
36      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
37      memcpy(d, p->pila + p->tope + sizeof(unsigned),
38             minimo(cantBytes, tamInfo));
39      return 1;
40  }
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

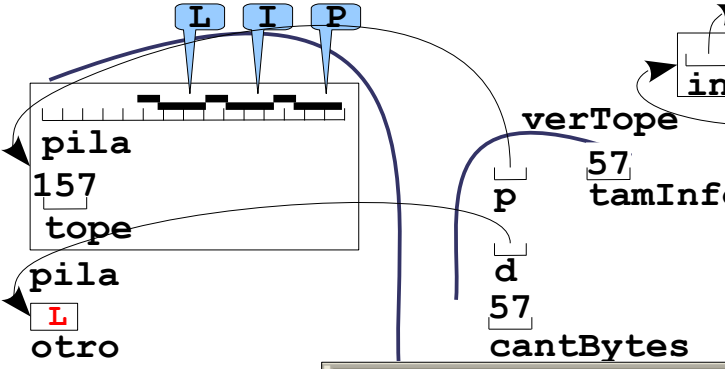
```
107     puts("Ver tope de la pila:");
```

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

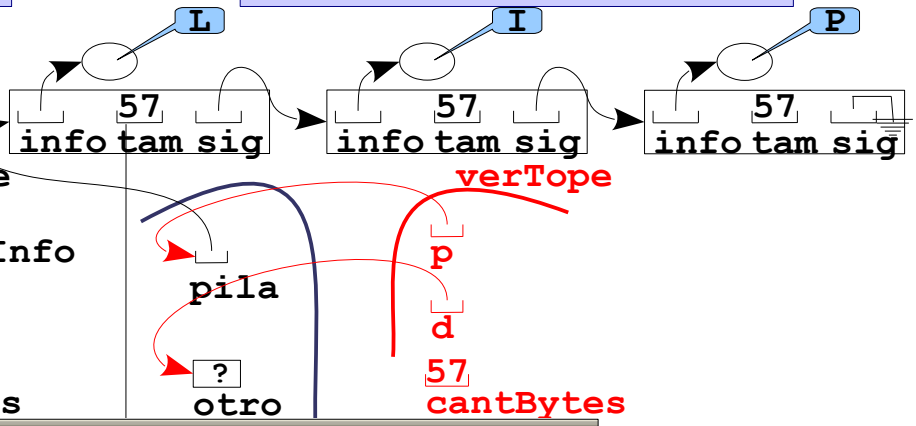
```
30  int  verTope(const tPila *p, void *d, unsigned cantBytes)
31  {
32      unsigned tamInfo;
33
34      if(p->tope == TAM_PILA)
35          return 0;
36      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
37      memcpy(d, p->pila + p->tope + sizeof(unsigned),
38             minimo(cantBytes, tamInfo));
39      return 1;
40  }
```

Tipo de dato Pila.

Implementación estática



Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
107     puts("Ver tope de la pila:");
```

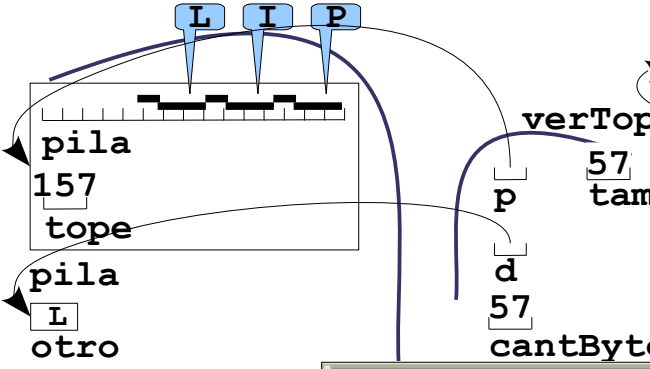
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
30  int  verTope(const tPila *p, void *d, unsigned cantBytes)
31  {
32      unsigned tamInfo;
33
34      if(p->tope == TAM_PILA)
35          return 0;
36      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
37      memcpy(d, p->pila + p->tope + sizeof(unsigned),
38             minimo(cantBytes, tamInfo));
39      return 1;
40  }
```

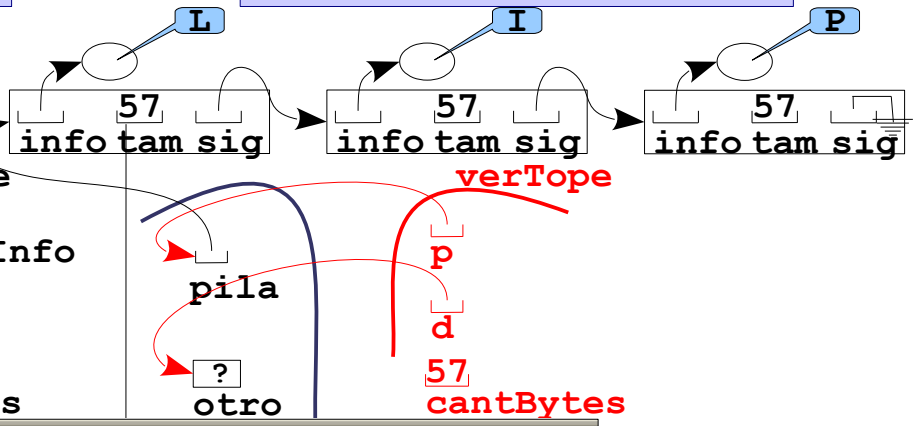
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
107     puts("Ver tope de la pila:");
```

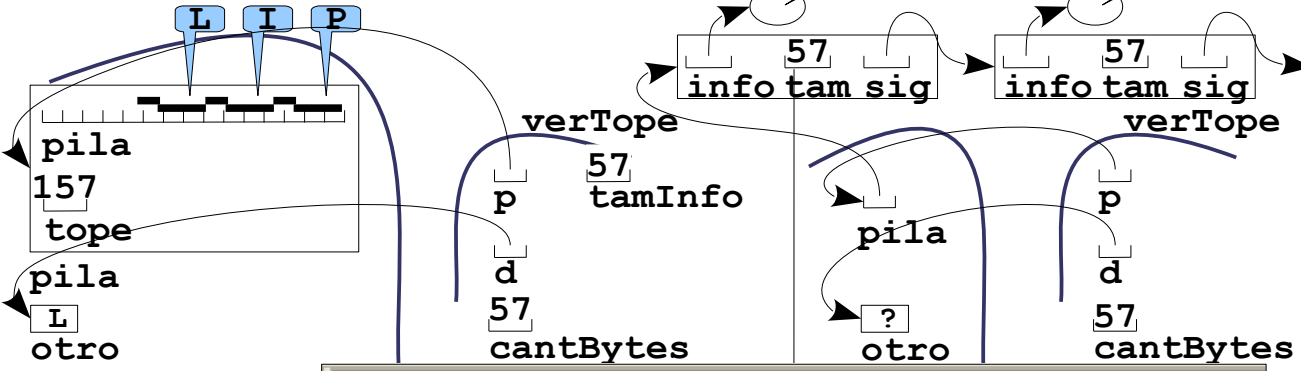
```
main.c | main.h | productos.c | pro
30  int  verTope(const
31  {
32      unsigned tamIn
33
34      if(p->tope ==
35          return 0;
36      memcpy(&tamInf
37      memcpy(d, p->pi
38              minimo(cantBytes, tamInfo));
39      return 1;
40  }
```

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
41  int  verTope(const tPila *p, void *d, unsigned cantBytes)
42  {
43      if(*p == NULL)
44          return 0;
45      memcpy(d, (*p)->info, minimo(cantBytes, (*p)->tamInfo));
46      return 1;
47  }
```

Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

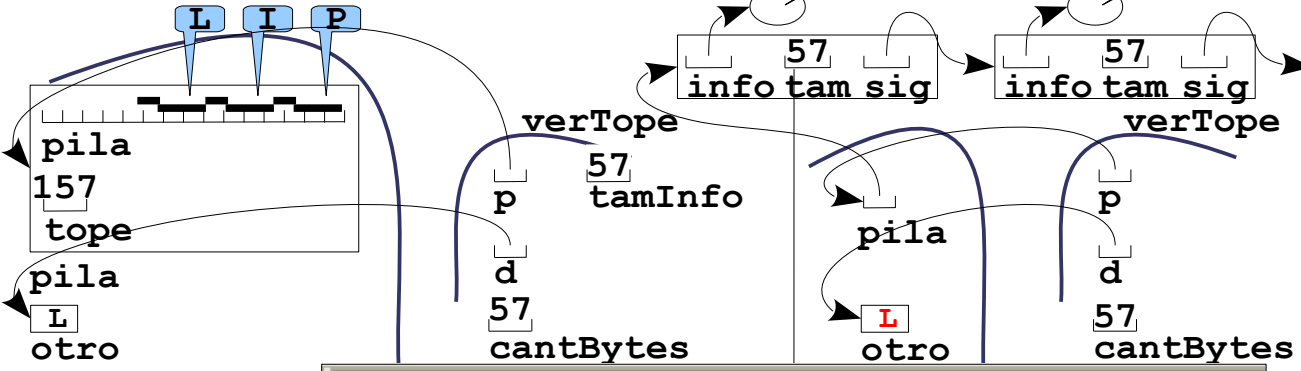

```
30  int  verTope(const
31  {
32      unsigned tamIn
33
34      if(p->tope ==
35          return 0;
36      memcpy(&tamInf
37      memcpy(d, p->pi
38              minimo(cantBytes, tamInfo));
39      return 1;
40  }
```

```
41  int  verTope(const tPila *p, void *d, unsigned cantBytes)
42  {
43      if(*p == NULL)
44          return 0;
45      memcpy(d, (*p)->info, minimo(cantBytes, (*p)->tamInfo));
46      return 1;
47  }
```

Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica


```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
107     puts("Ver tope de la pila:");
```

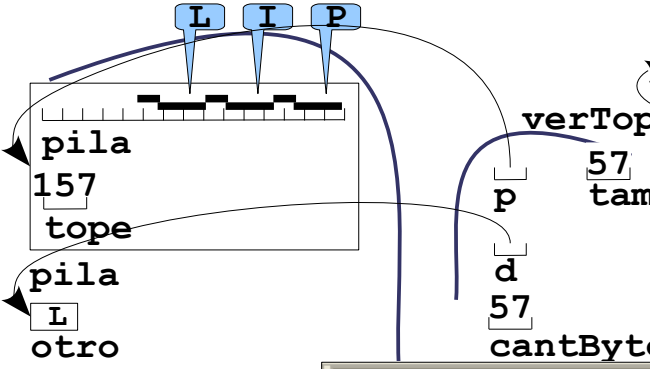
```
main.c | main.h | productos.c | pro
30  int  verTope(const
31  {
32      unsigned tamIn
33
34      if(p->tope ==
35          return 0;
36      memcpy(&tamInf
37      memcpy(d, p->pi
38              minimo(cantBytes, tamInfo));
39      return 1;
40  }
```

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
41  int  verTope(const tPila *p, void *d, unsigned cantBytes)
42  {
43      if(*p == NULL)
44          return 0;
45      memcpy(d, (*p)->info, minimo(cantBytes, (*p)->tamInfo));
46      return 1;
47  }
```

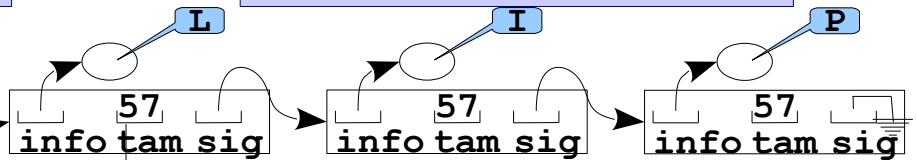
Tipos de Datos Abstractos

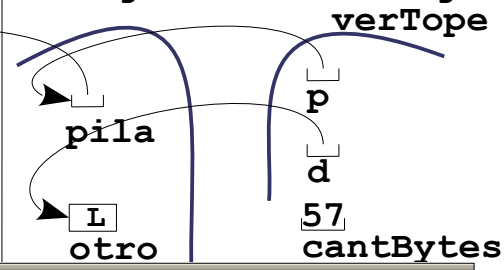
Tipo de dato Pila.

Implementación estática



Implementación dinámica






```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
107     puts("Ver tope de la pila:");
```

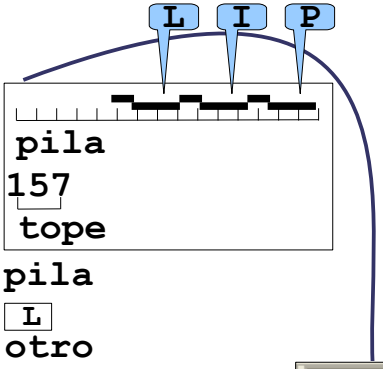
```
main.c | main.h | productos.c | pro
30  int  verTope(const
31  {
32      unsigned tamIn
33
34      if(p->tope ==
35          return 0;
36      memcpy(&tamInf
37      memcpy(d, p->pi
38              minimo(cantBytes, tamInfo));
39      return 1;
40  }
```

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
41  int  verTope(const tPila *p, void *d, unsigned cantBytes)
42  {
43      if(*p == NULL)
44          return 0;
45      memcpy(d, (*p)->info, minimo(cantBytes, (*p)->tamInfo));
46      return 1;
47  }
```

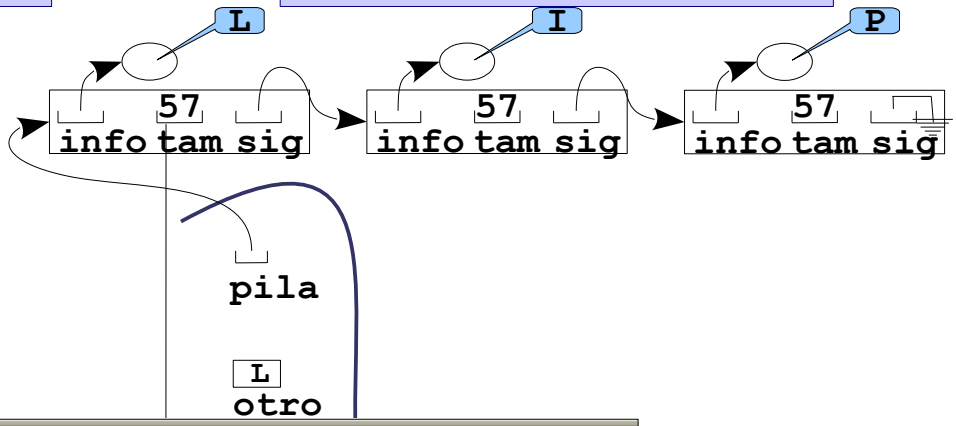
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

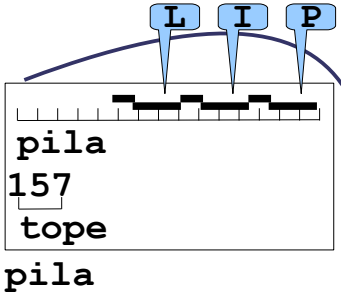



```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

107     puts("Ver tope de la pila:");
108     if(verTope(&pila, &otro, sizeof(otro)))
109         mostrarProducto(&otro);
110     else
111         puts("La pila estaba vacia.");
```

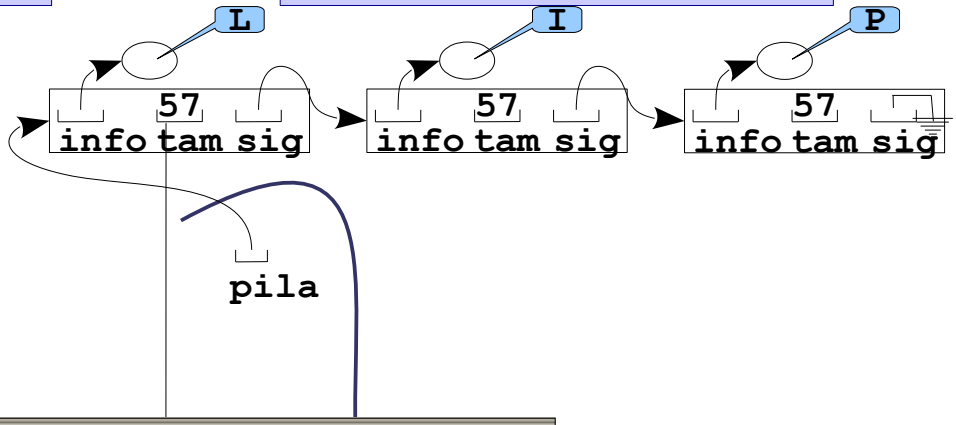
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

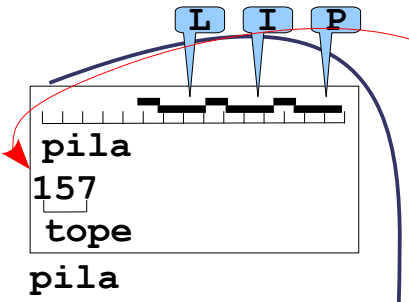


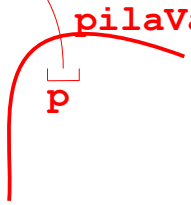

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
120         puts("La pila esta vacía");
121     else
122     {
123         puts("Vaciando la pila");
124         vaciarPila(&pila);
125     }
```

Tipos de Datos Abstractos

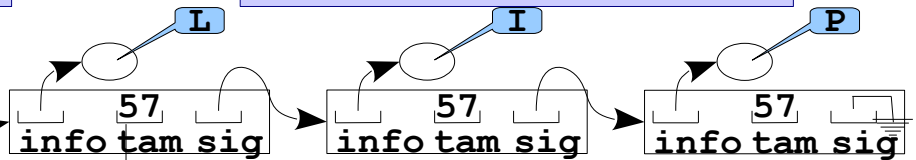
Tipo de dato Pila.

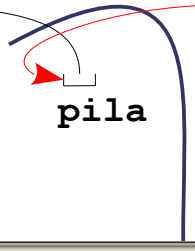
Implementación estática

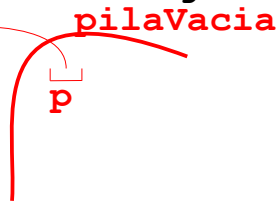




Implementación dinámica





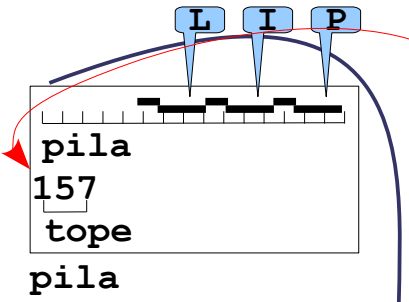


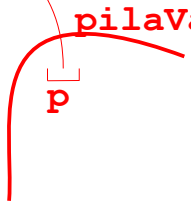

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
120         puts("La pila esta vacía");
121     else
122     {
123         puts("Vaciando la pila");
124         vaciarPila(&pila);
125     }
```

Tipos de Datos Abstractos

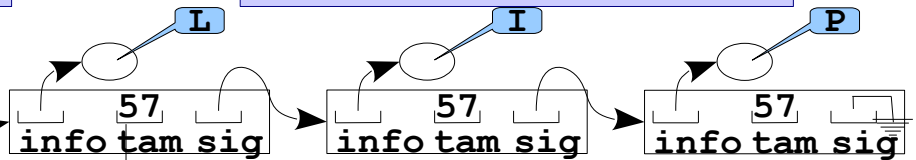
Tipo de dato Pila.

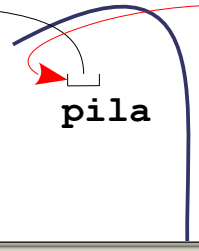
Implementación estática

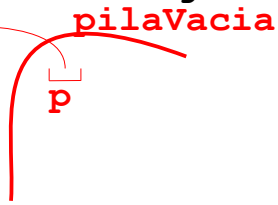




Implementación dinámica








```
119     if(pilaVacia(&pila))
120         puts("La pila esta vacía");
121     else
122     {
123         puts("Vaciando la pila");
```

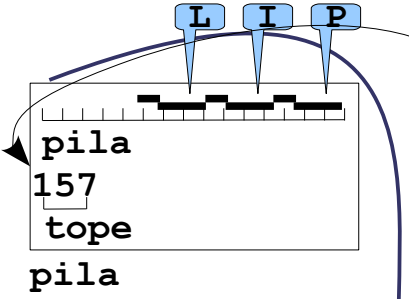
```
42  int  pilaVacia(const tPila *p)
43  {
44      return p->tope == TAM_PILA;
45  }
```

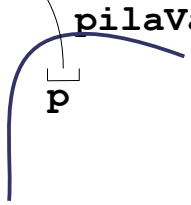
```
49  int  pilaVacia(const tPila *p)
50  {
51      return *p == NULL;
52  }
```

Tipos de Datos Abstractos

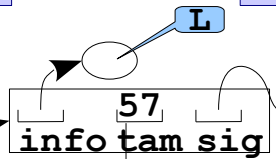
Tipo de dato Pila.

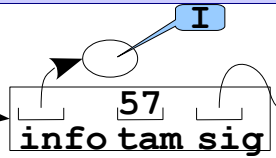
Implementación estática

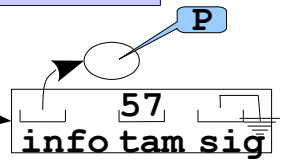


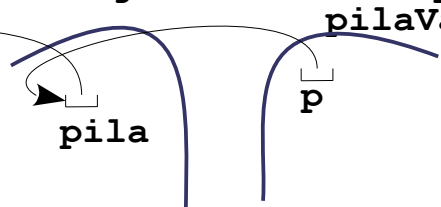


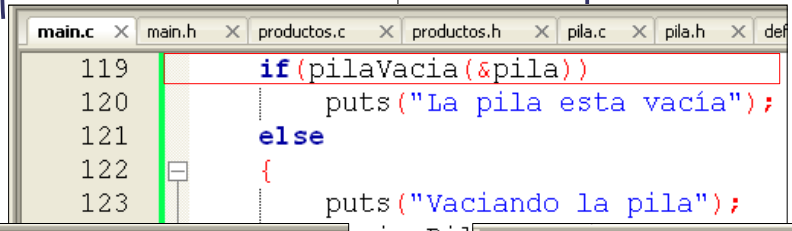
Implementación dinámica

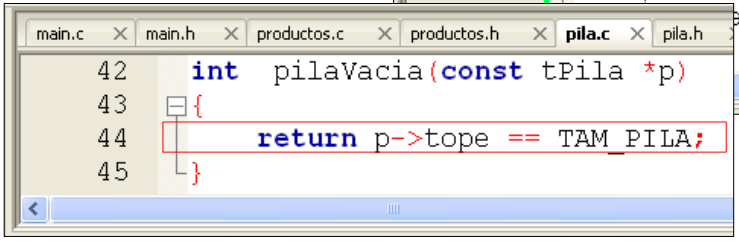


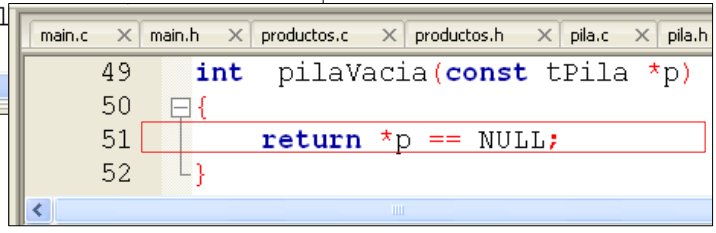








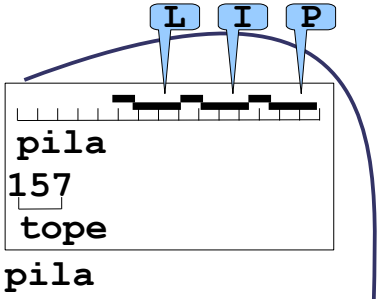




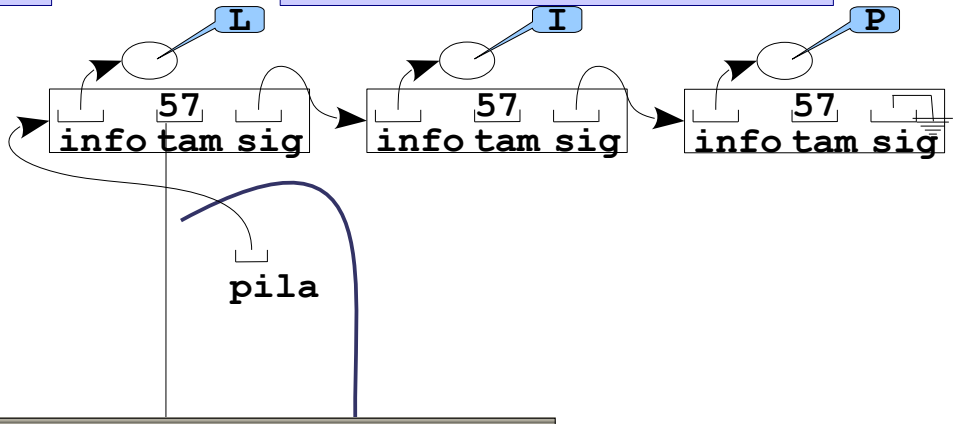
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica

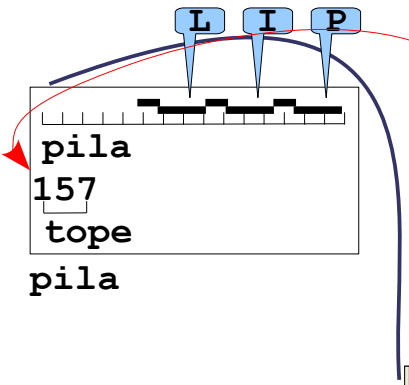


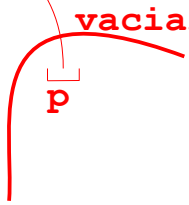

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
120         puts("La pila esta vacía");
121     else
122     {
123         puts("Vaciando la pila");
124         vaciarPila(&pila);
125     }
```

Tipos de Datos Abstractos

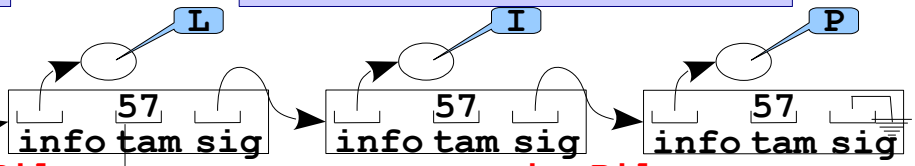
Tipo de dato Pila.

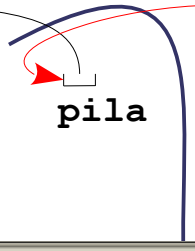
Implementación estática

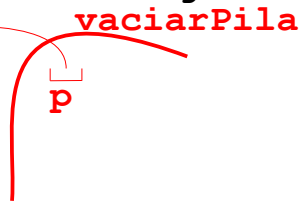




Implementación dinámica





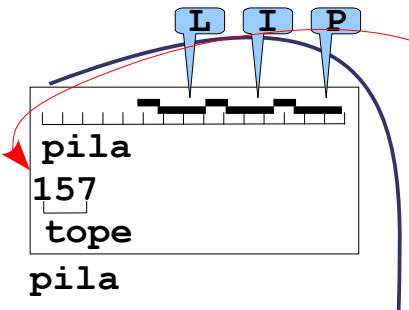


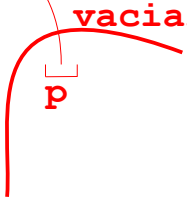

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
120         puts("La pila esta vacía");
121     else
122     {
123         puts("Vaciando la pila");
124         vaciarPila(&pila);
125     }
```

Tipos de Datos Abstractos

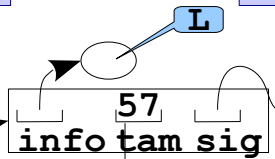
Tipo de dato Pila.

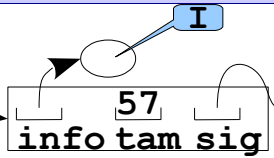
Implementación estática

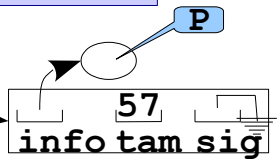


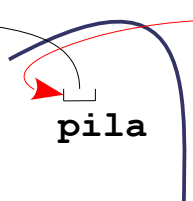


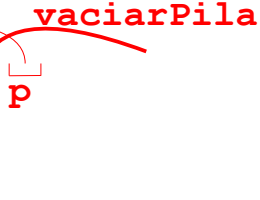
Implementación dinámica












```
119     if(pilaVacia(&pila))
            puts("La pil
        else

            puts("Vacian
            vaciarPila(&
```

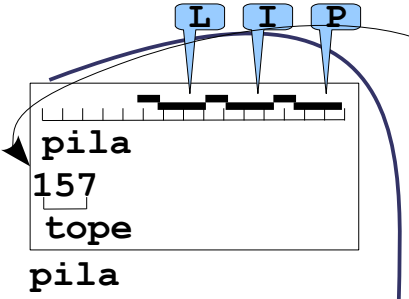
```
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

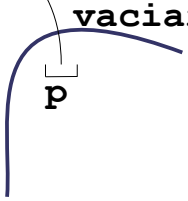
```
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

Tipos de Datos Abstractos

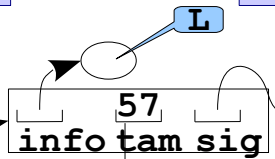
Tipo de dato Pila.

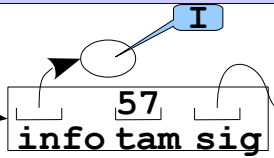
Implementación estática

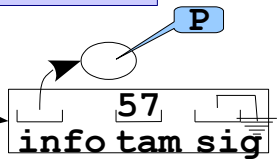


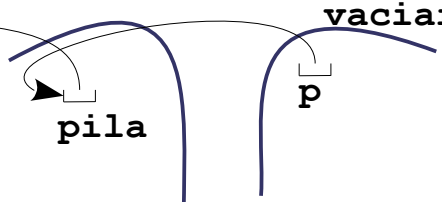


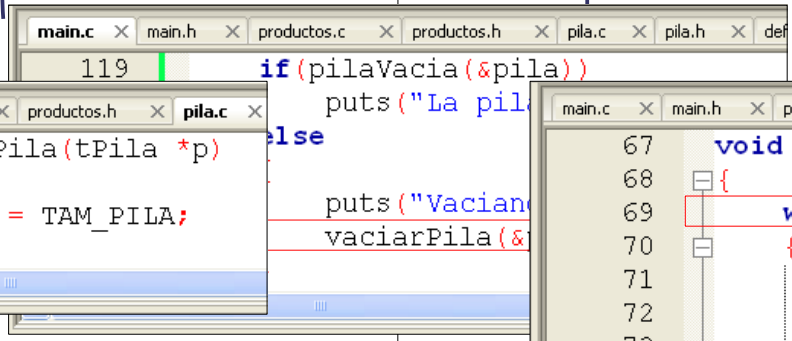
Implementación dinámica

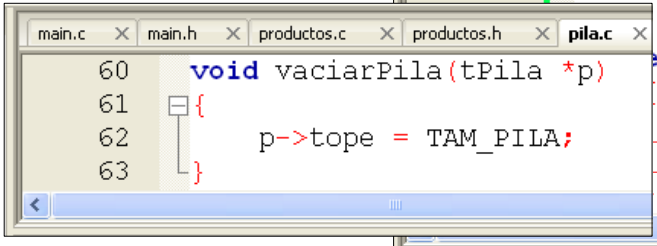


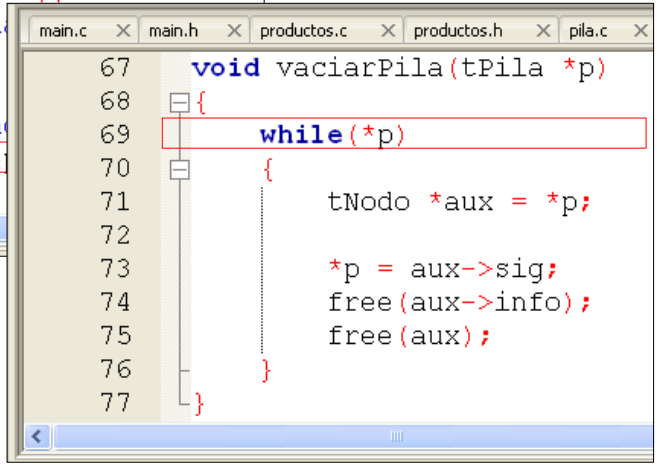








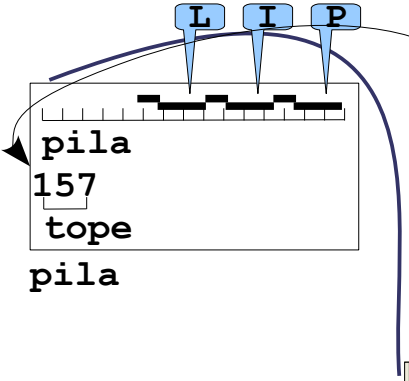


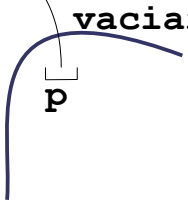


Tipos de Datos Abstractos

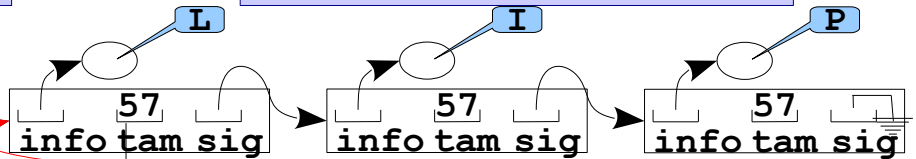
Tipo de dato Pila.

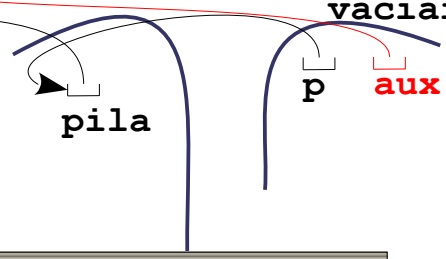
Implementación estática

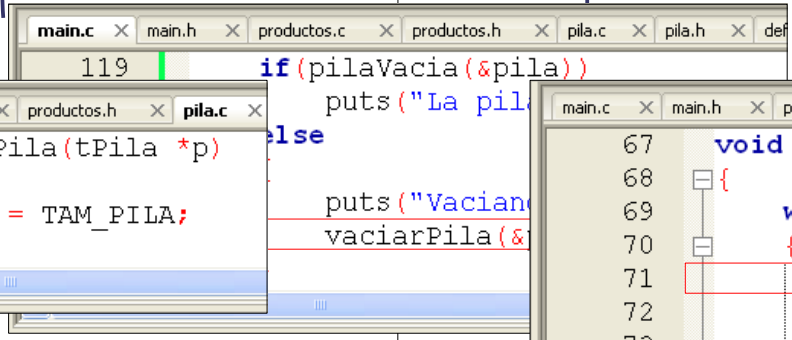


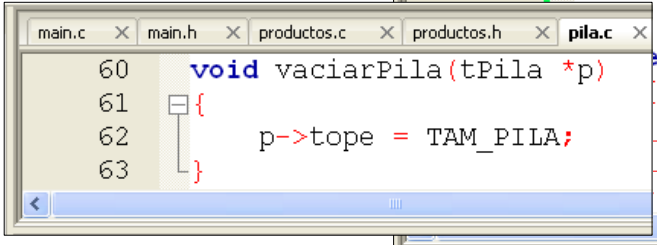


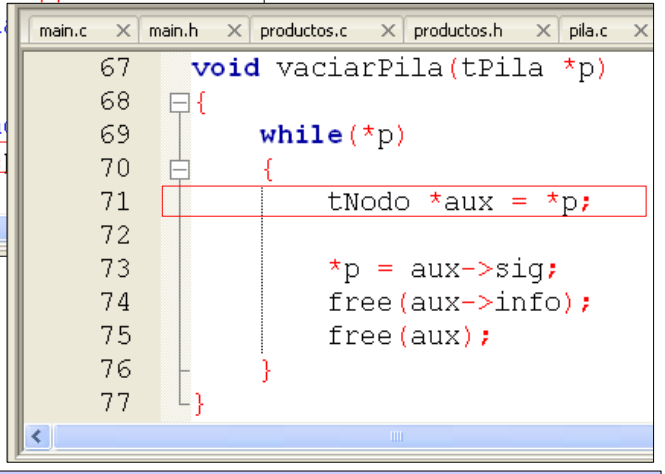
Implementación dinámica







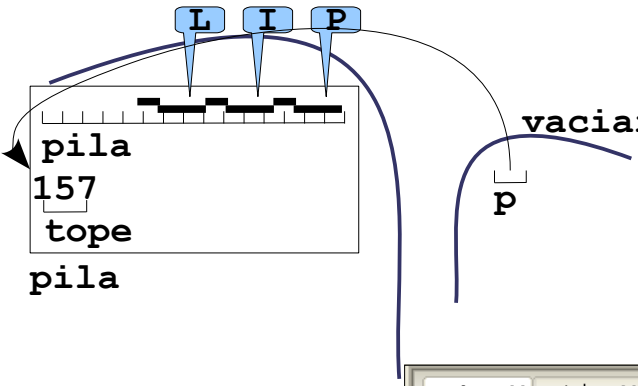




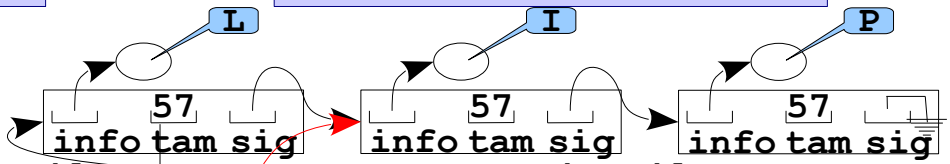
Tipos de Datos Abstractos

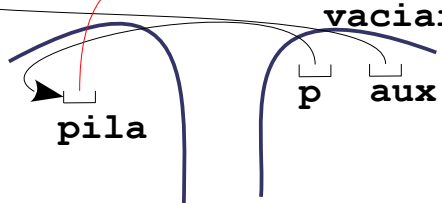
Tipo de dato Pila.

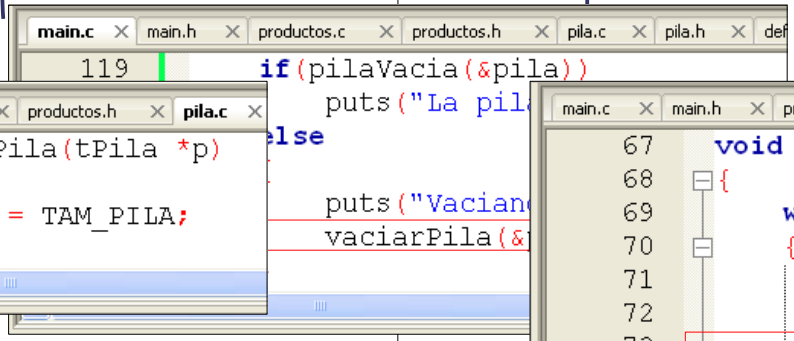
Implementación estática

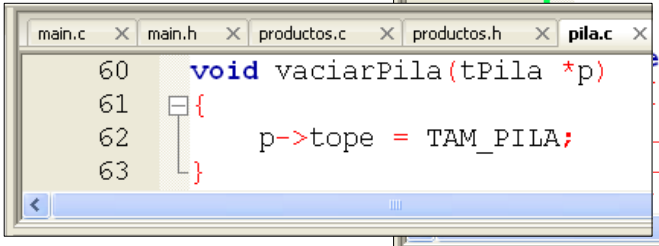


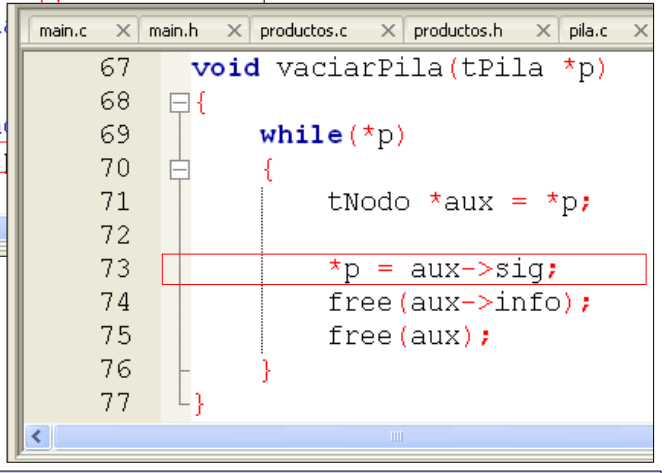
Implementación dinámica







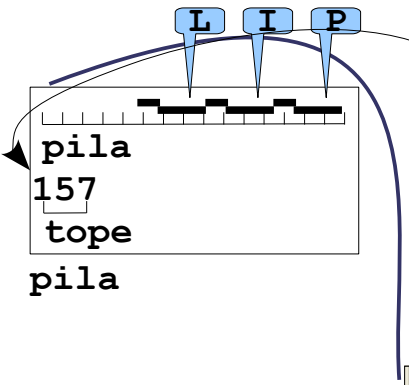


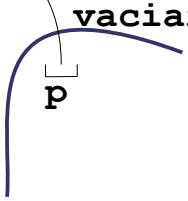


Tipos de Datos Abstractos

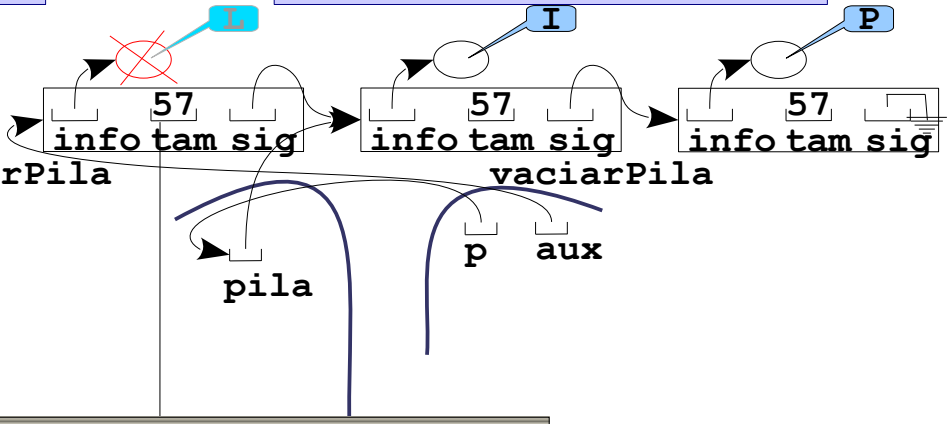
Tipo de dato Pila.

Implementación estática





Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

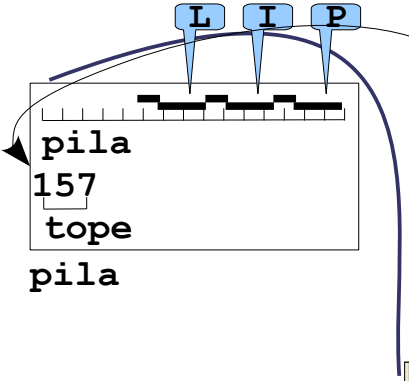
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

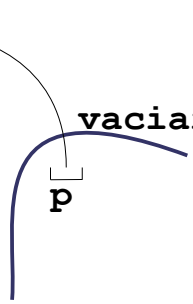
```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

Tipos de Datos Abstractos

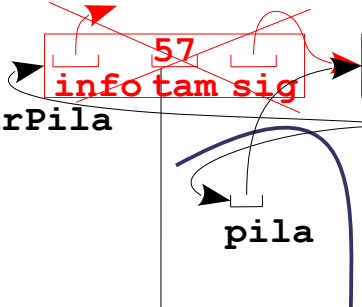
Tipo de dato Pila.

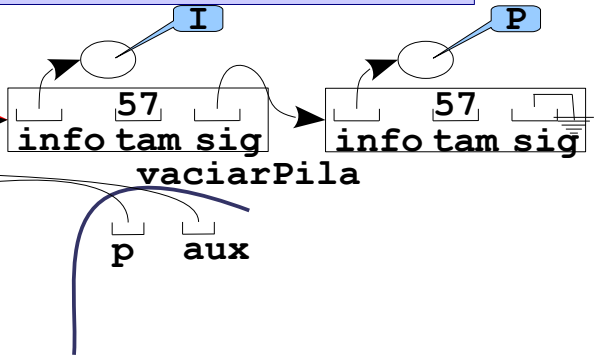
Implementación estática





Implementación dinámica






```
119         if(pilaVacia(&pila))
                puts("La pil
            else

                puts("Vacian
                vaciarPila(&
```

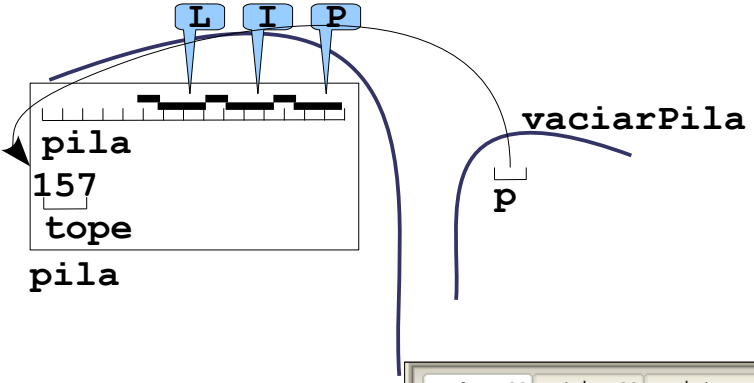
```
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

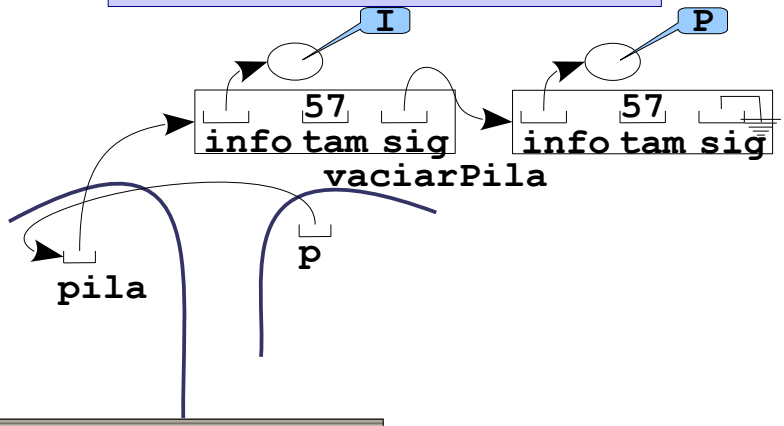
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

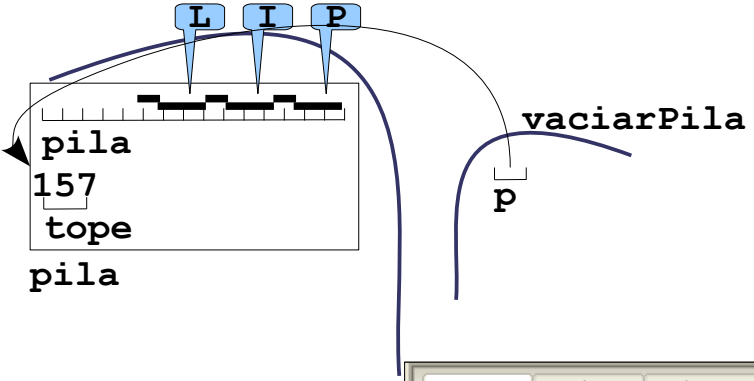
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

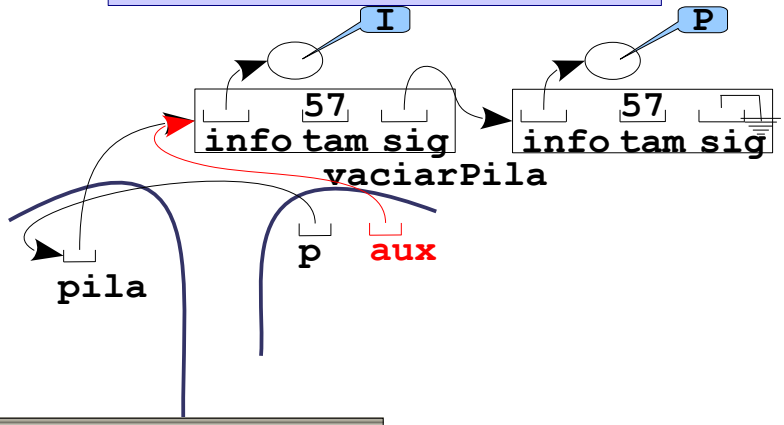
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

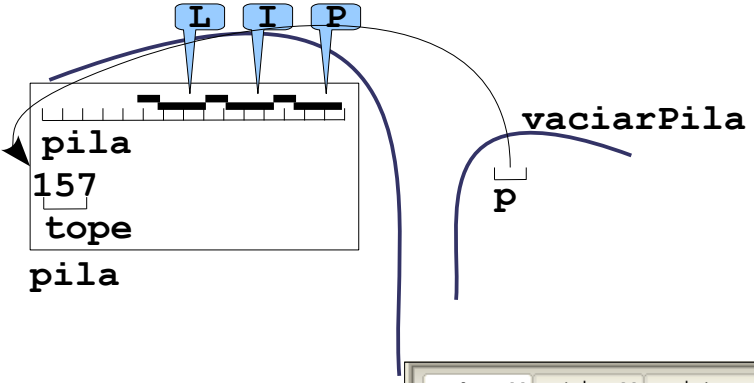
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

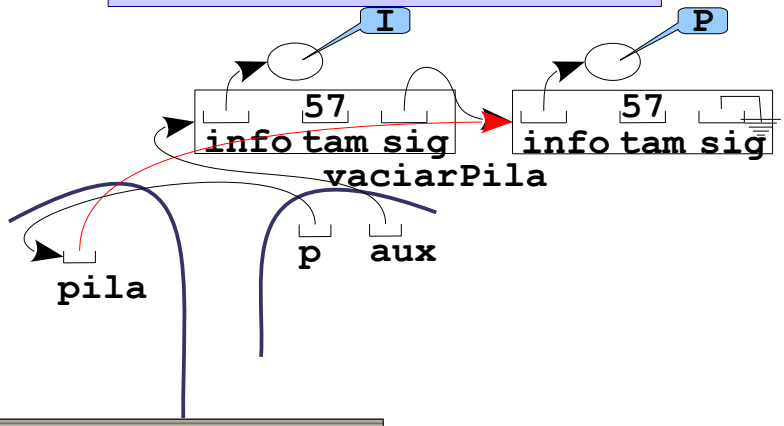
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

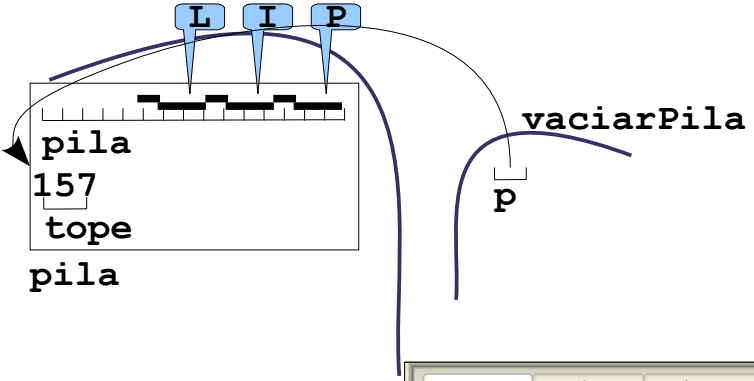
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

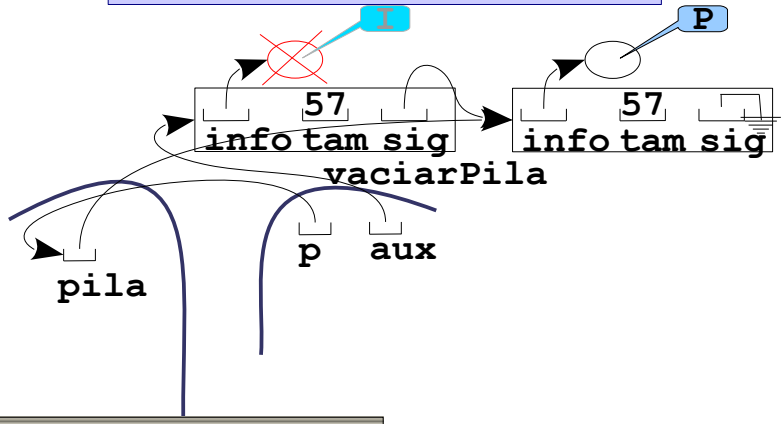
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

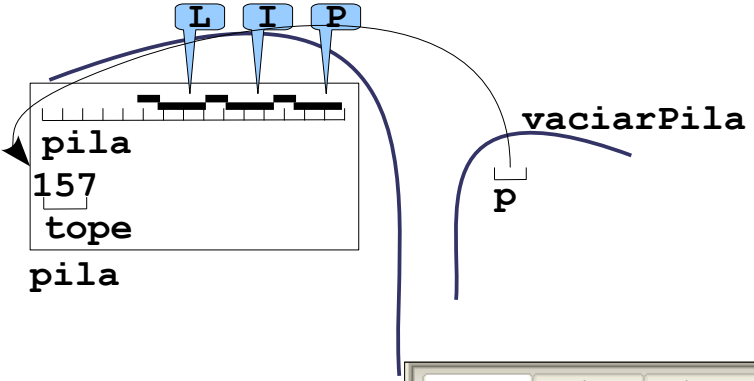
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

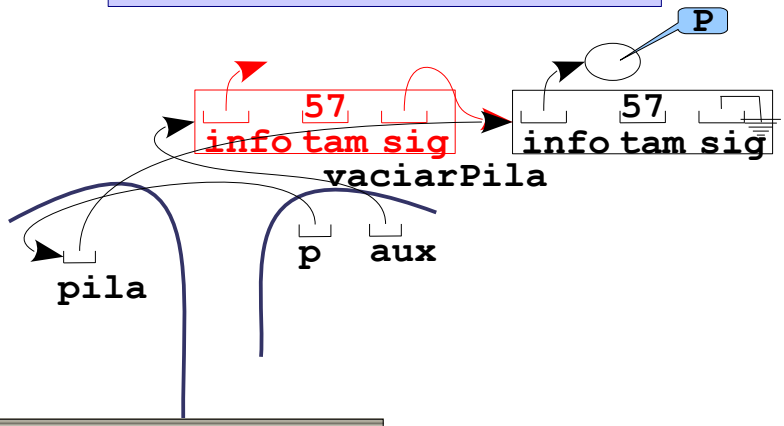
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

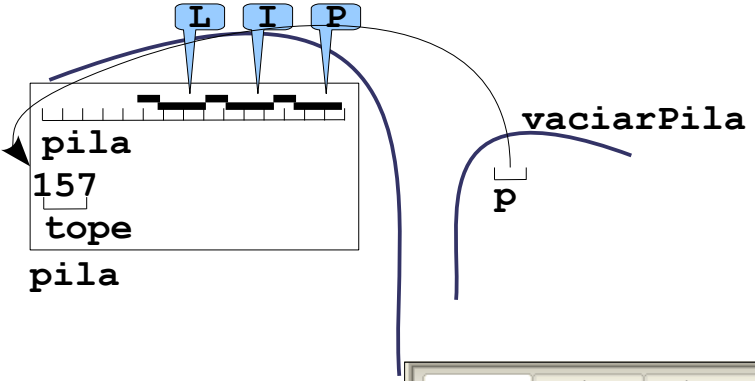
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

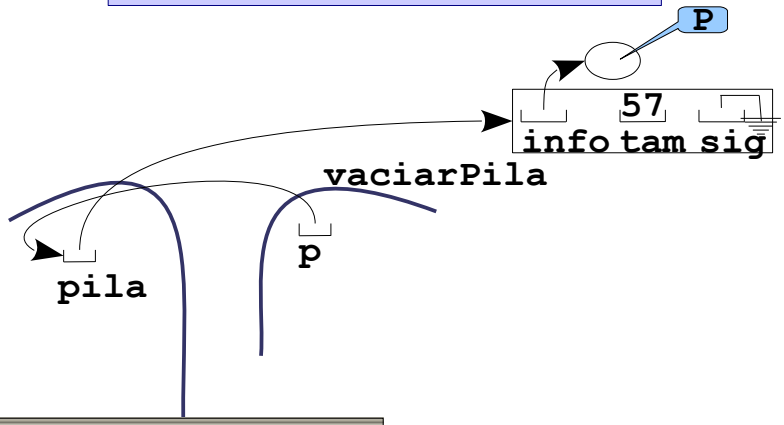
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

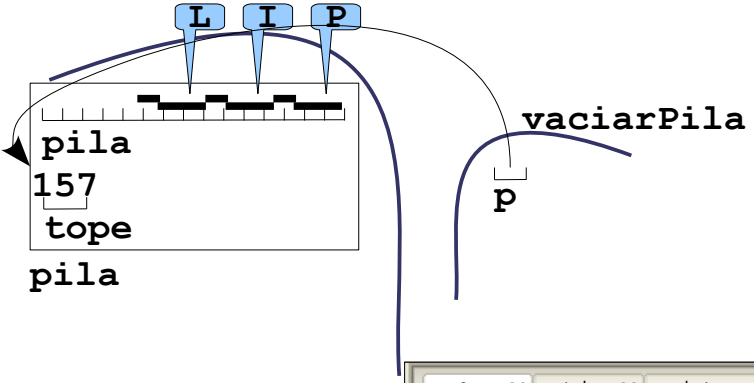
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

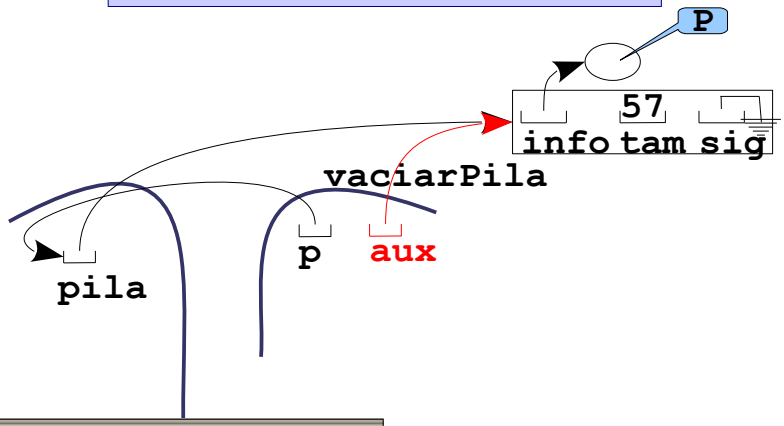
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

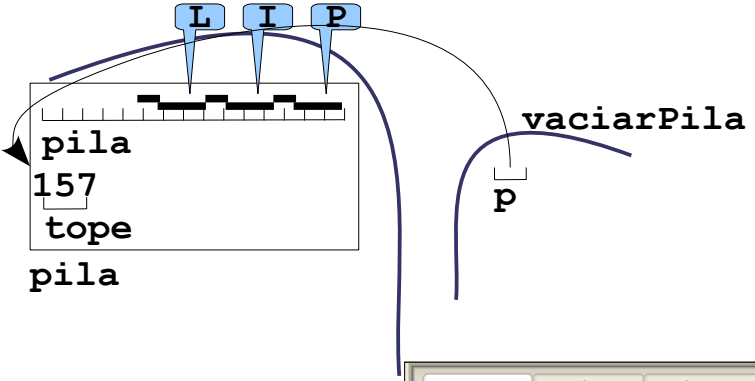
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

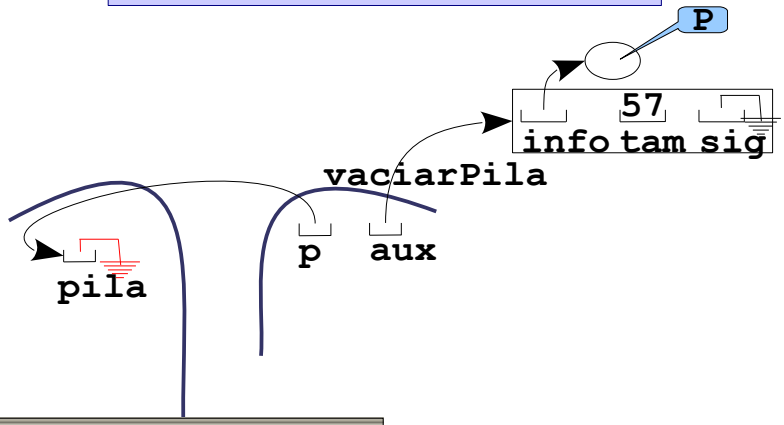
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

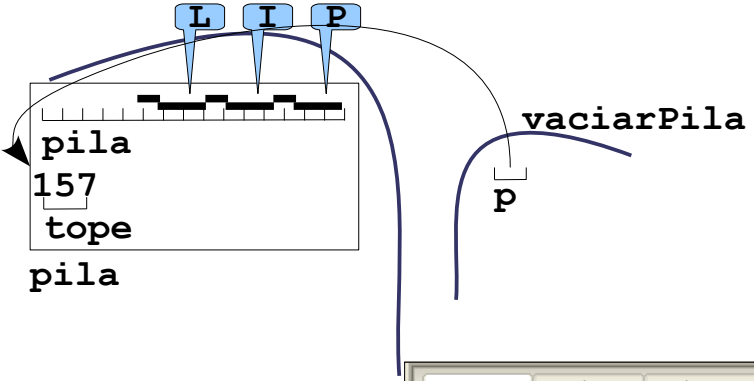
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

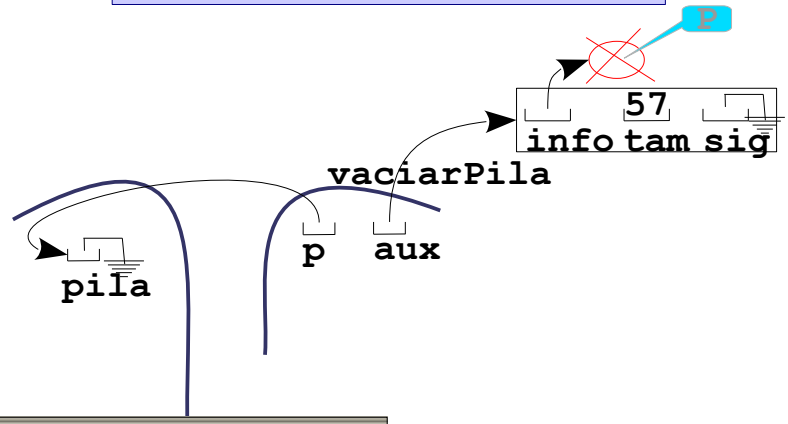
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

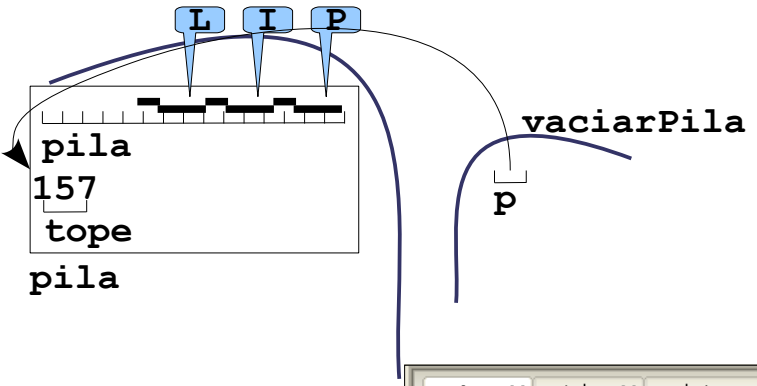
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

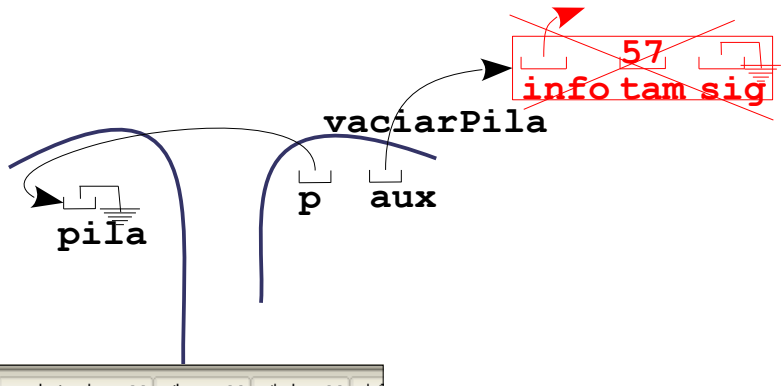
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

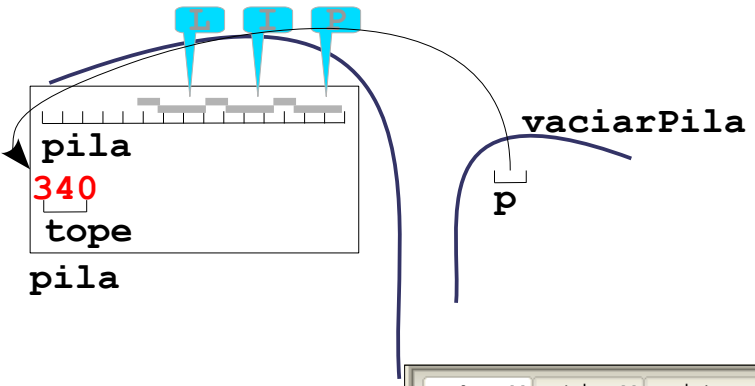
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

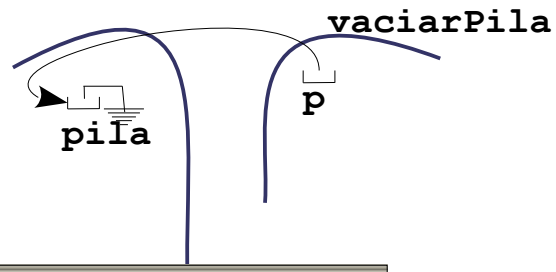
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática



Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else
            puts("Vacian
            vaciarPila(&
```

```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

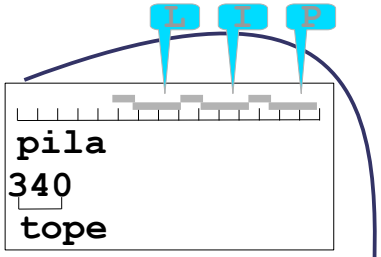
```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

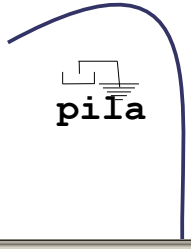
Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

Implementación dinámica






```
main.c | main.h | productos.c | productos.h | pila.c | pila.h
119     if(pilaVacia(&pila))
            puts("La pil
        else

            puts("Vacian
            vaciarPila(&
```

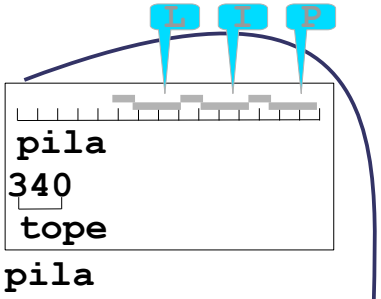
```
main.c | main.h | productos.c | productos.h | pila.c
60  void vaciarPila(tPila *p)
61  {
62      p->tope = TAM_PILA;
63  }
```

```
main.c | main.h | productos.c | productos.h | pila.c
67  void vaciarPila(tPila *p)
68  {
69      while(*p)
70      {
71          tNodo *aux = *p;
72
73          *p = aux->sig;
74          free(aux->info);
75          free(aux);
76      }
77  }
```

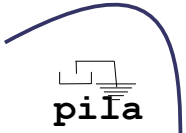
Tipos de Datos Abstractos

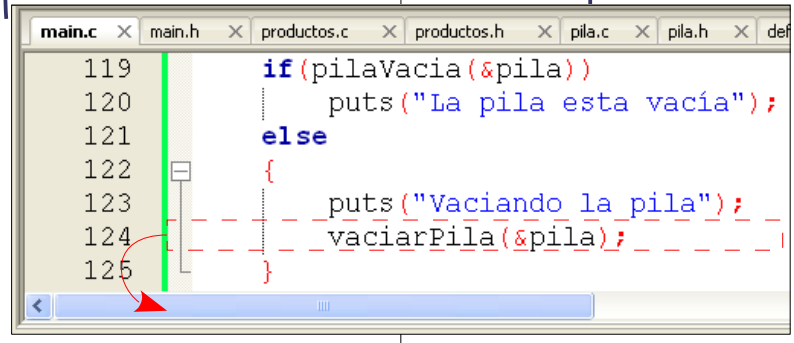
Tipo de dato Pila.

Implementación estática



Implementación dinámica

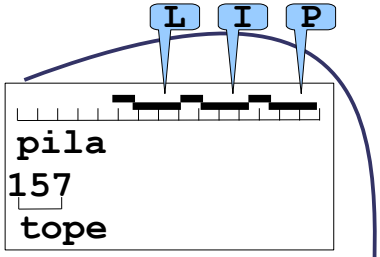




Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

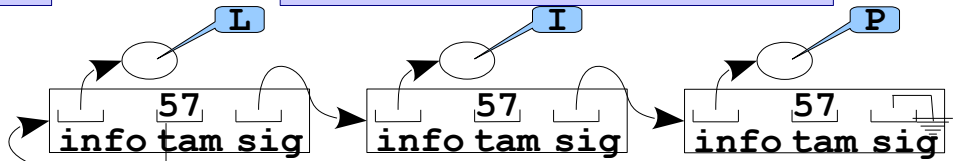


pila

L

prod

Implementación dinámica



pila

L

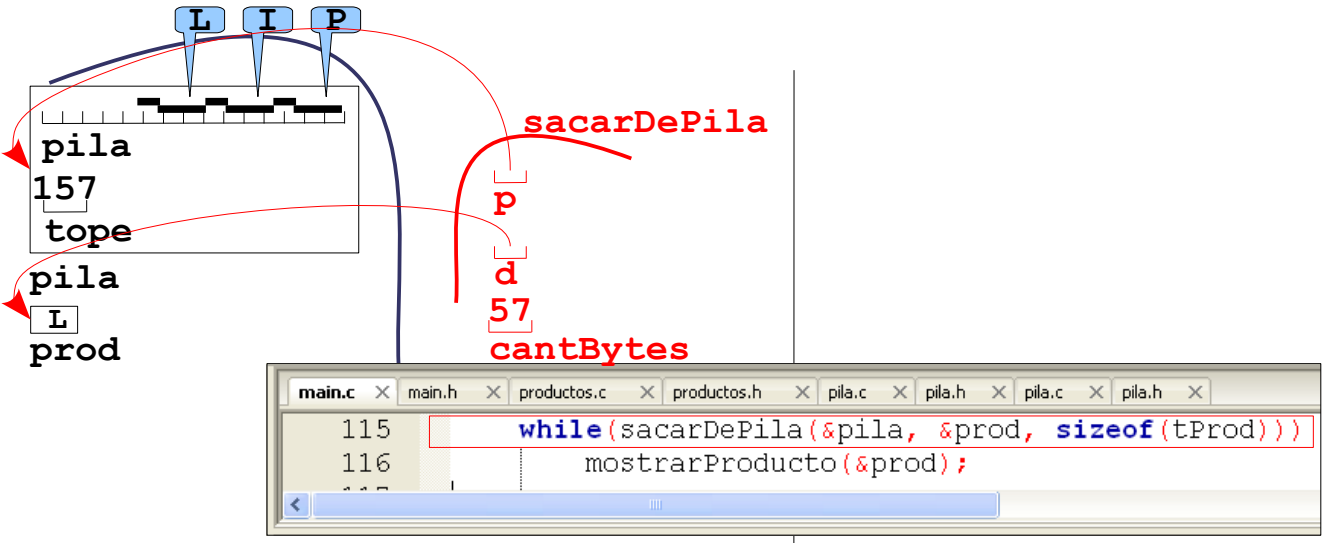
prod

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
115     while(sacarDePila(&pila, &prod, sizeof(tProd)))
116         mostrarProducto(&prod);
```

Tipo de dato Pila.

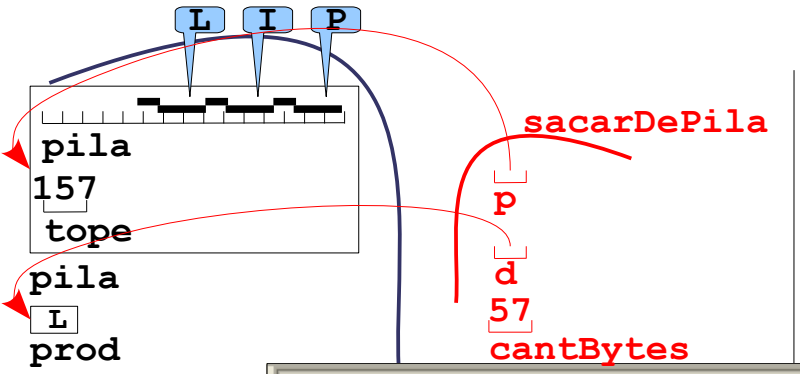
Implementación estática

Implementación dinámica



Tipo de dato Pila.

Implementación estática | Implementación dinámica

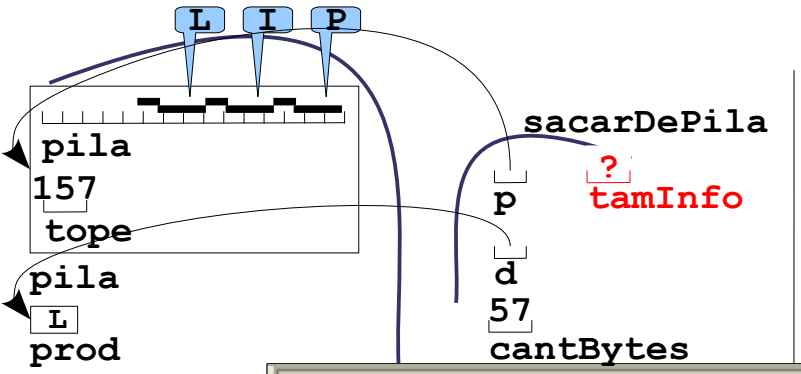


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

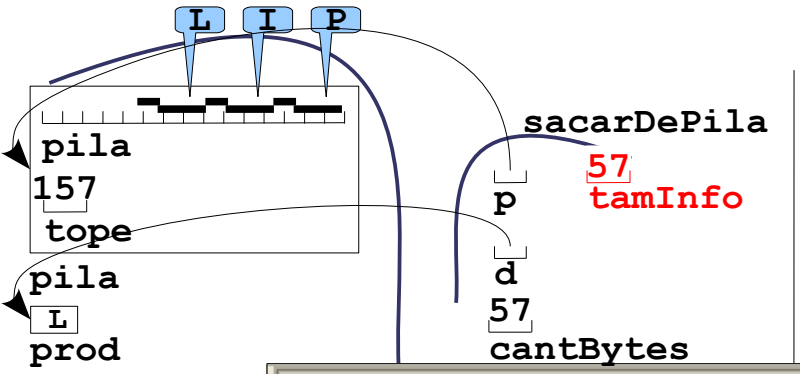


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

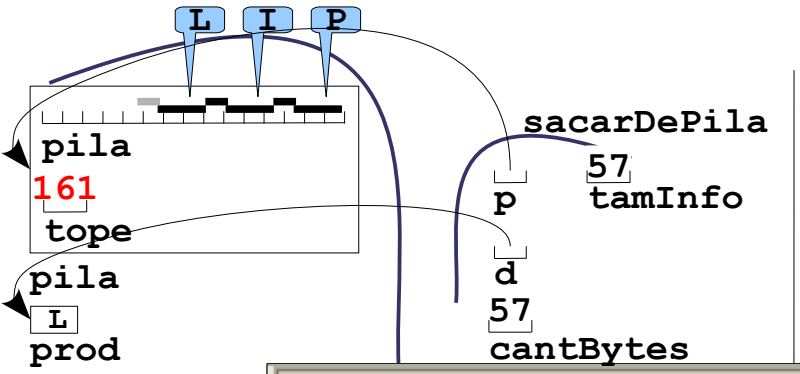


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

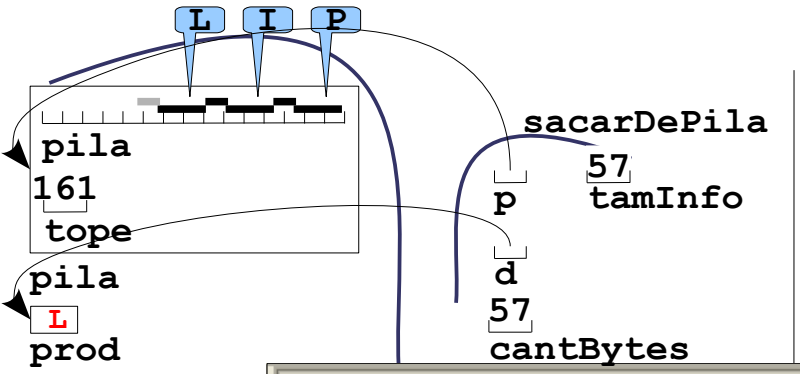


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

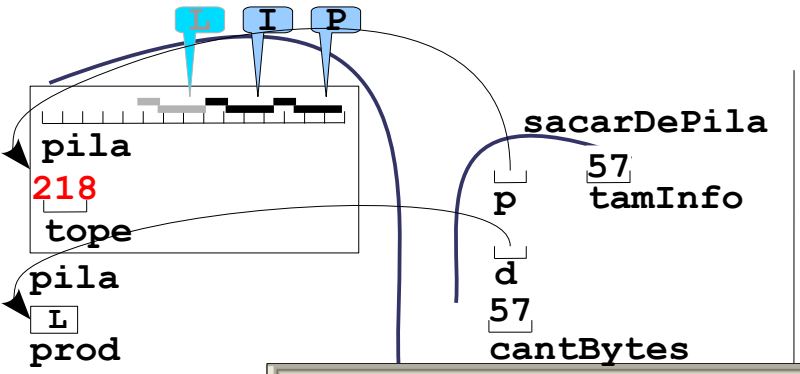


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

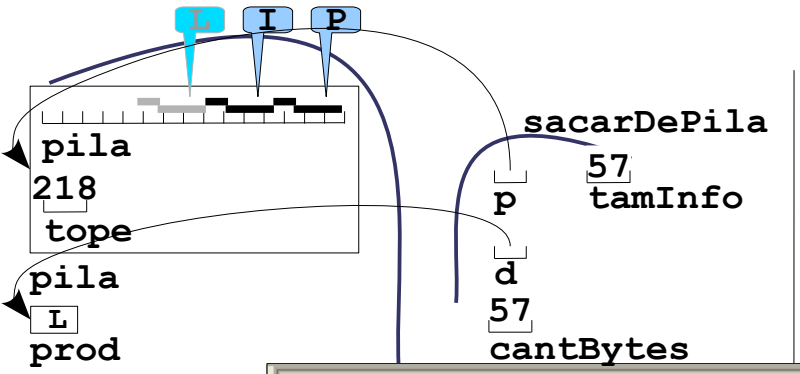


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica



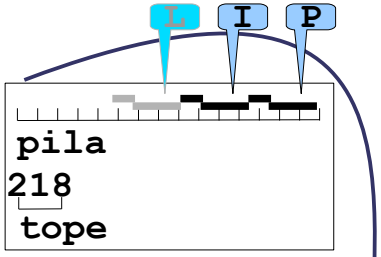
main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática

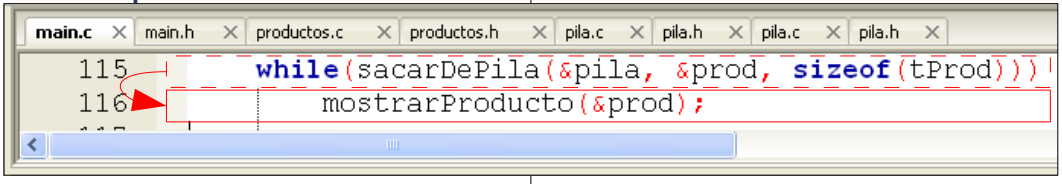
Implementación dinámica



pila

L

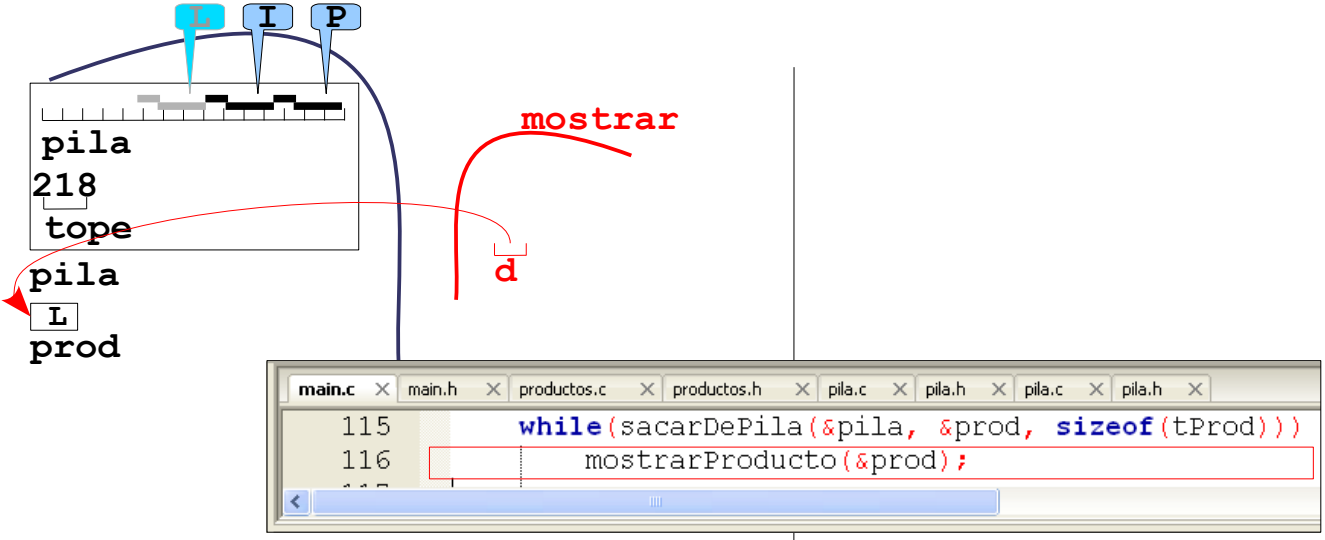
prod



Tipo de dato Pila.

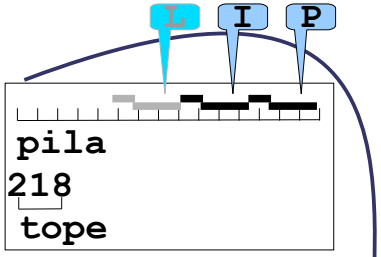
Implementación estática

Implementación dinámica



Tipo de dato Pila.

Implementación estática | Implementación dinámica

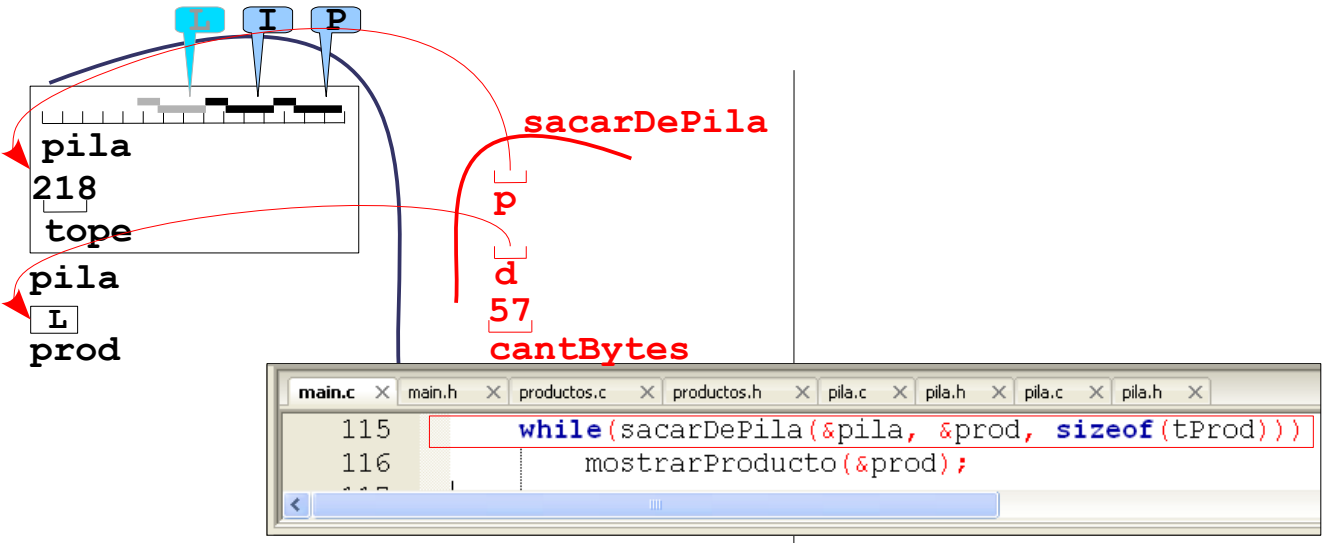



```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
115     while(sacarDePila(&pila, &prod, sizeof(tProd)))
116         mostrarProducto(&prod);
```

Tipo de dato Pila.

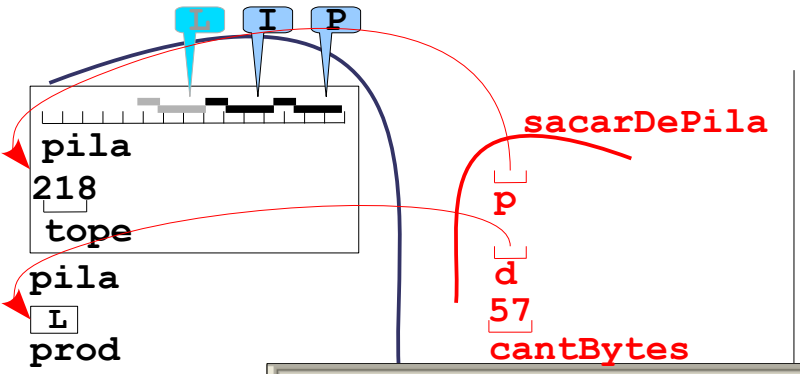
Implementación estática

Implementación dinámica



Tipo de dato Pila.

Implementación estática | Implementación dinámica

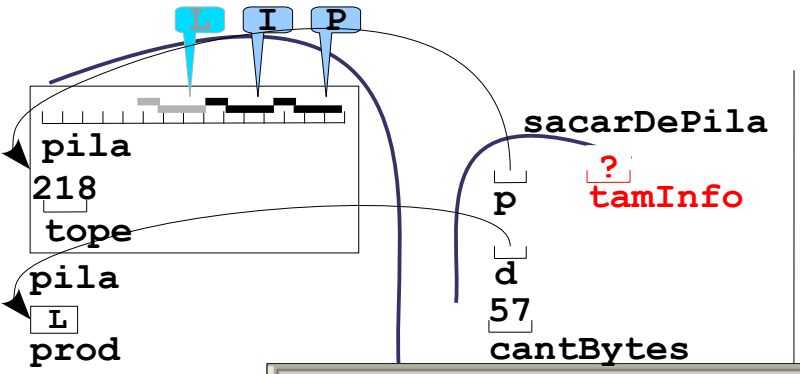


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

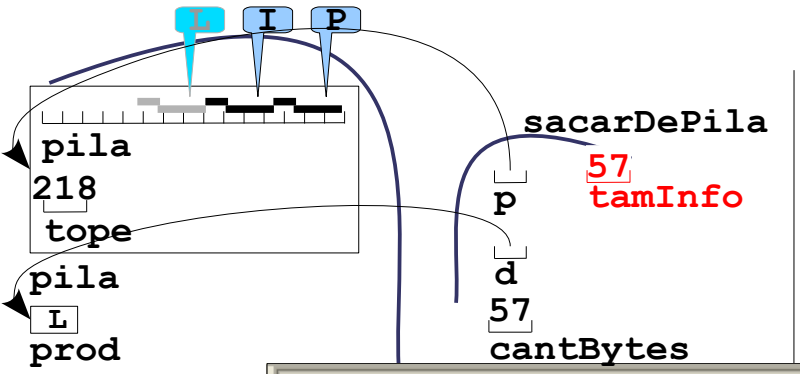


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

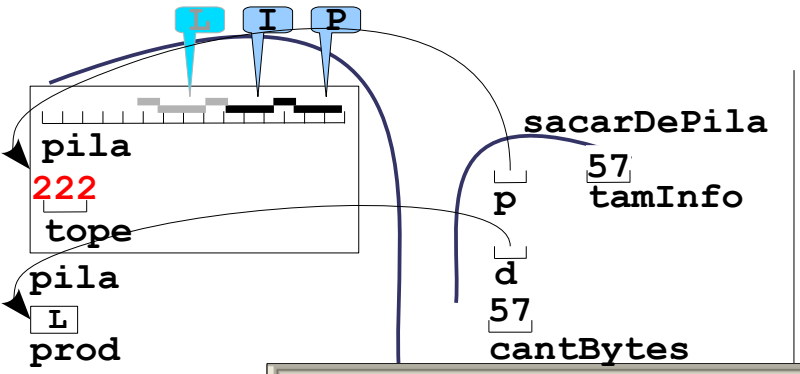


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

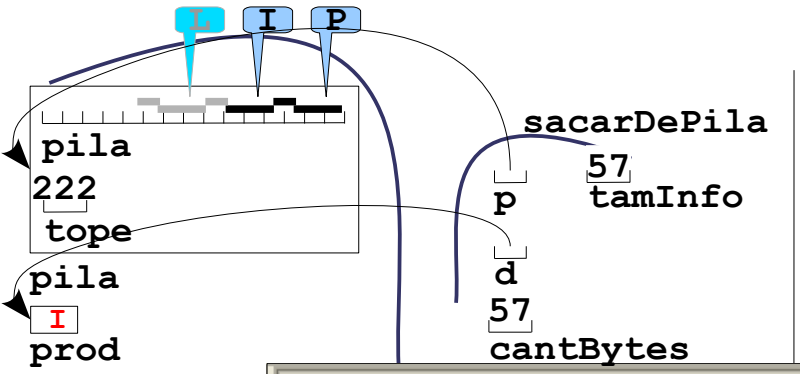


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

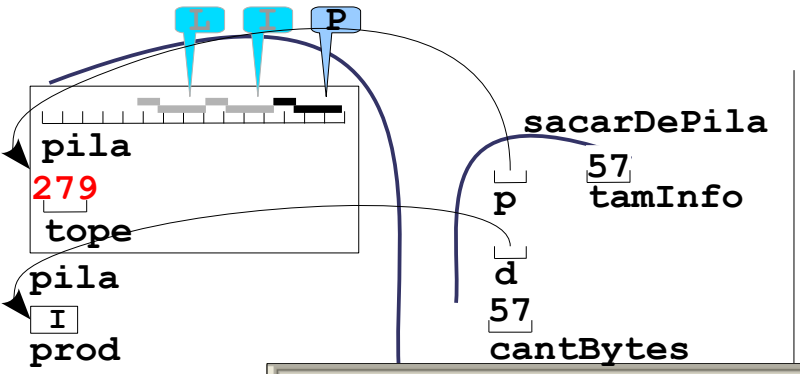


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

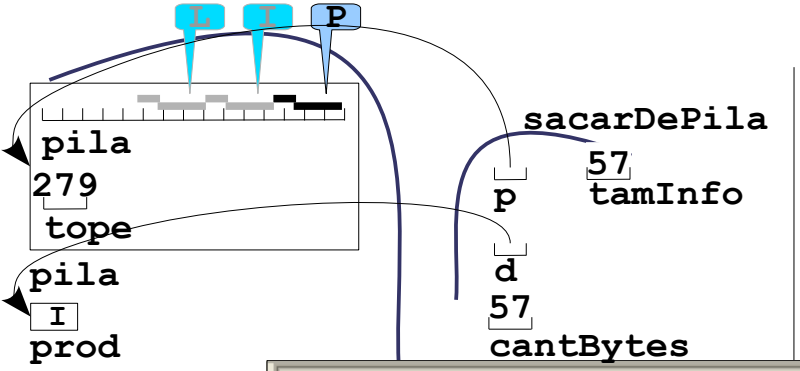


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

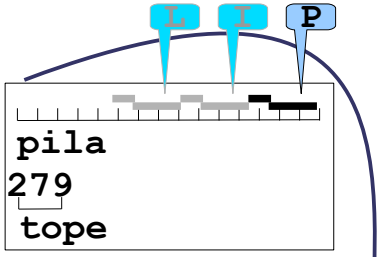


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

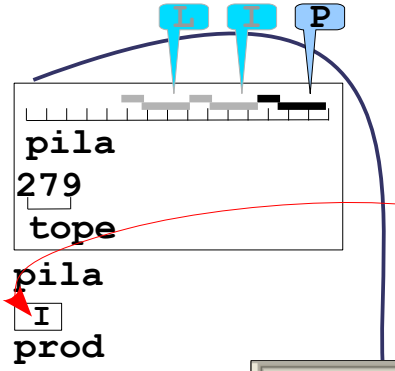
Implementación estática | Implementación dinámica

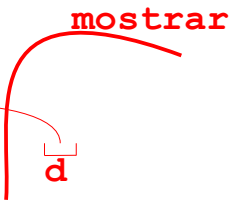



```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
115     while(sacarDePila(&pila, &prod, sizeof(tProd)))
116         mostrarProducto(&prod);
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica



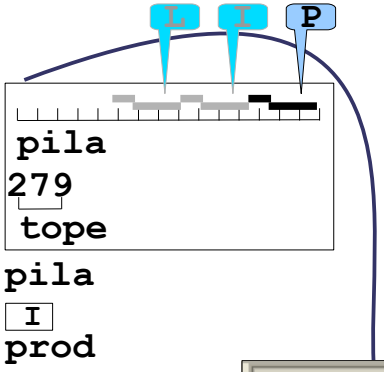



```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
115     while(sacarDePila(&pila, &prod, sizeof(tProd)))
116         mostrarProducto(&prod);
```

Tipos de Datos Abstractos

Tipo de dato Pila.

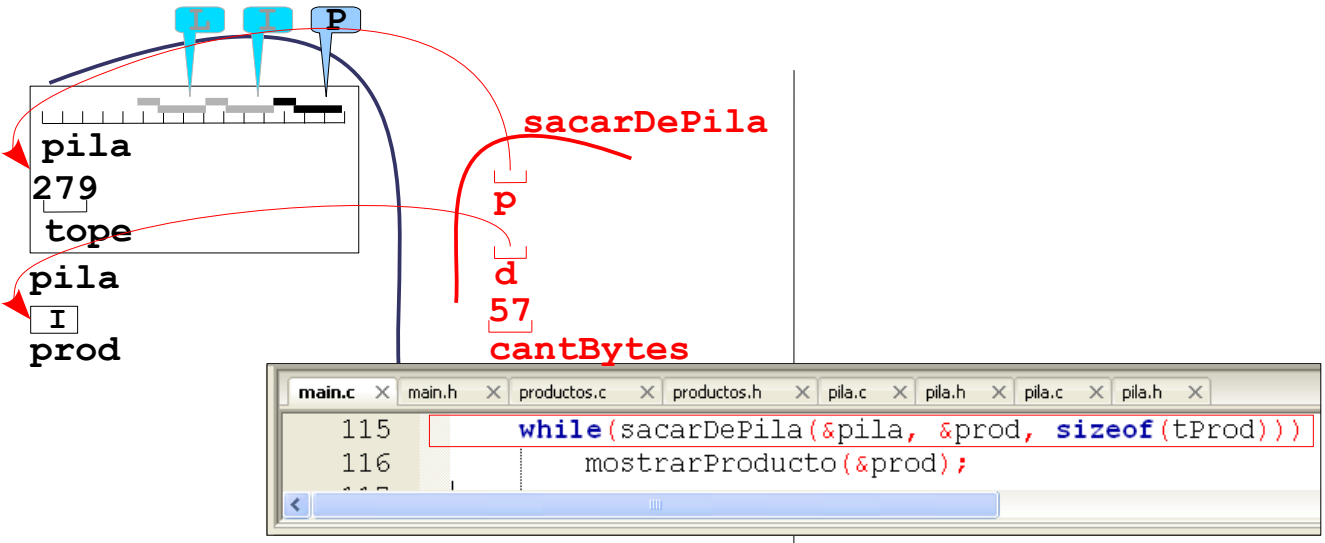
Implementación estática | Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
115     while(sacarDePila(&pila, &prod, sizeof(tProd)))
116         mostrarProducto(&prod);
```

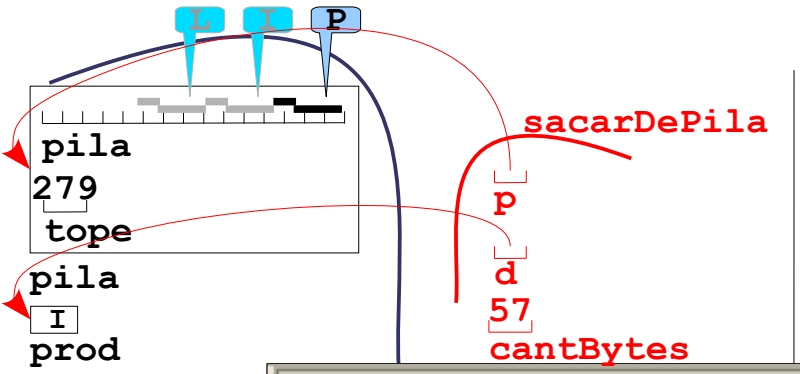
Tipo de dato Pila.

Implementación estática | Implementación dinámica



Tipo de dato Pila.

Implementación estática | Implementación dinámica

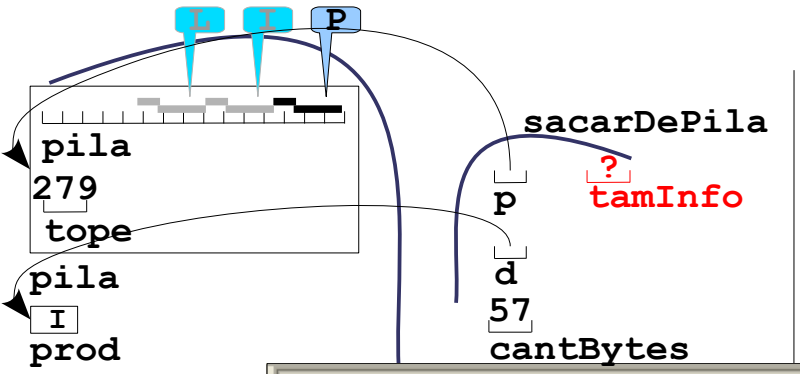


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

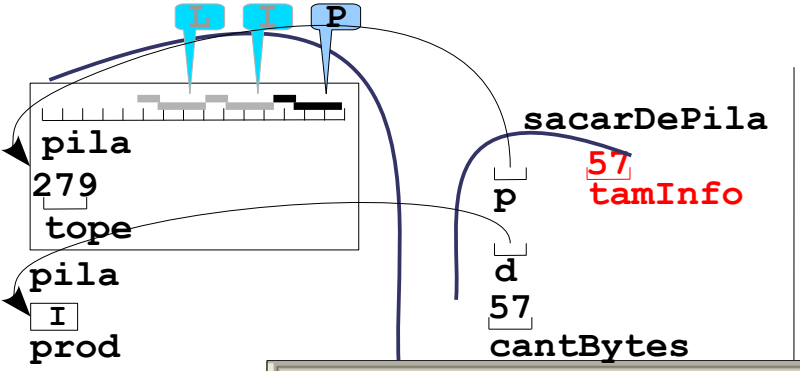


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

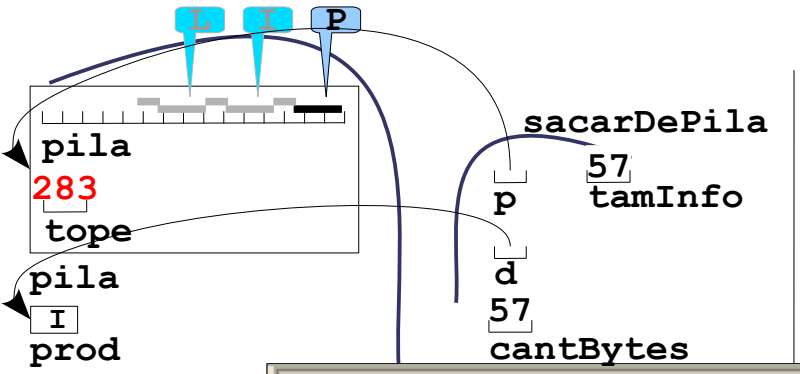


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

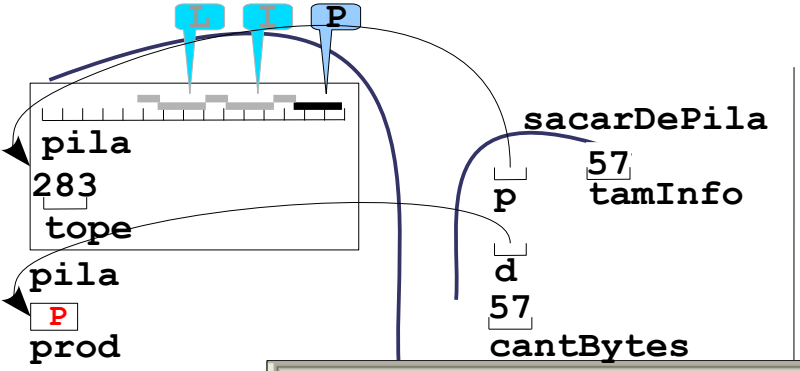


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

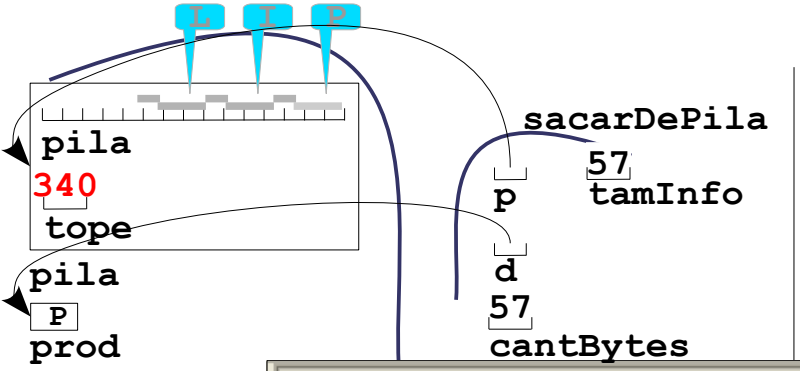


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

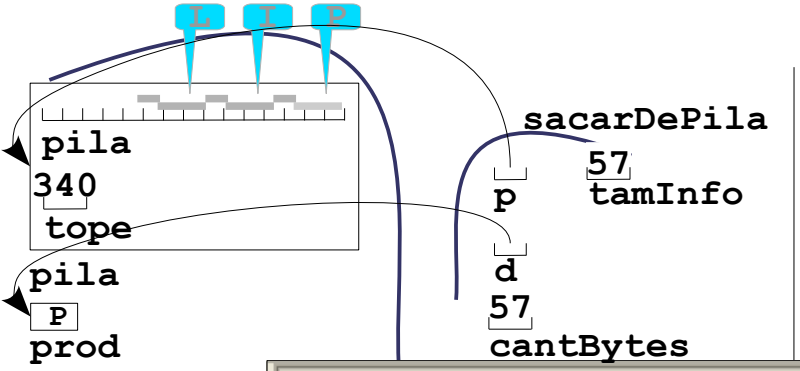


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica



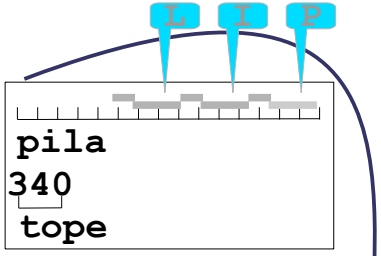
main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica



pila

P

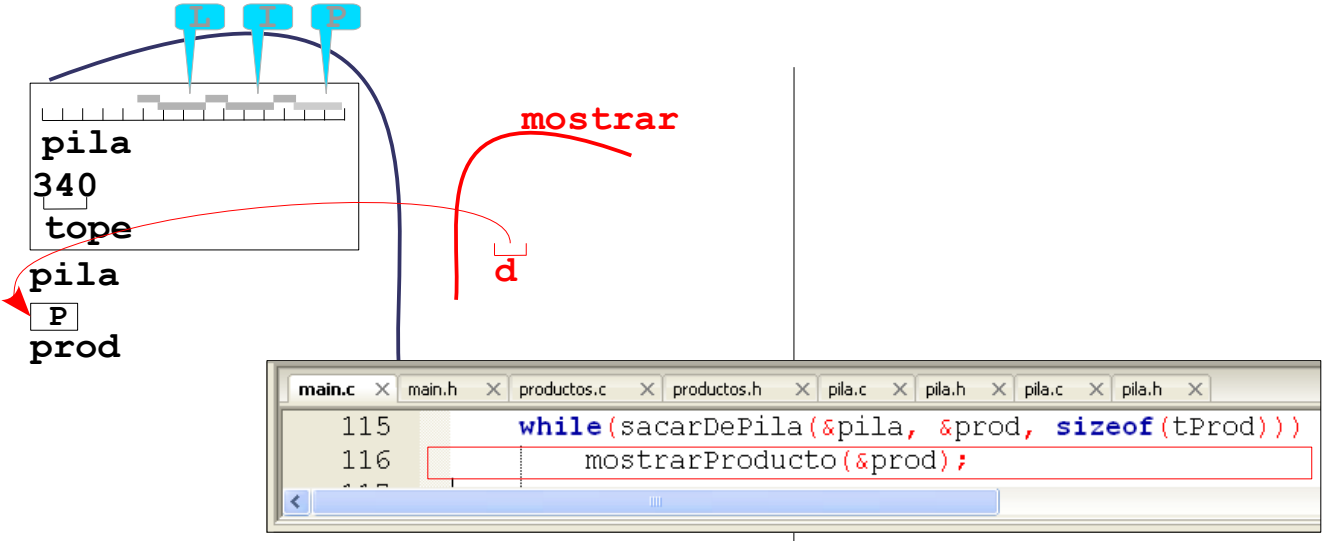
prod

main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
115     while(sacarDePila(&pila, &prod, sizeof(tProd)))
116         mostrarProducto(&prod);
```

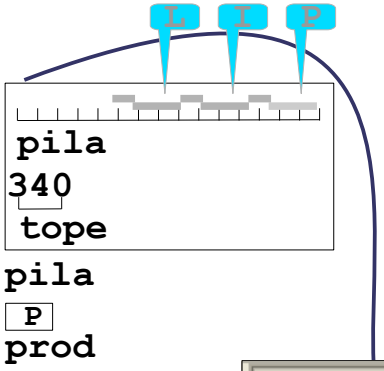
Tipo de dato Pila.

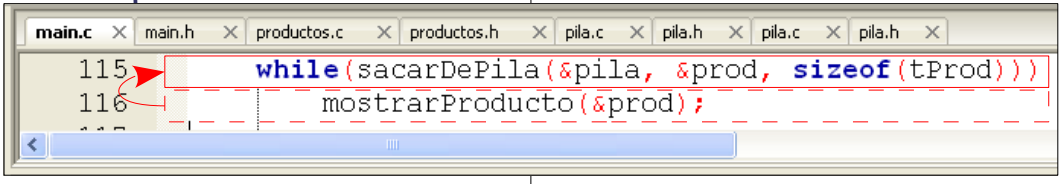
Implementación estática | Implementación dinámica



Tipo de dato Pila.

Implementación estática | Implementación dinámica

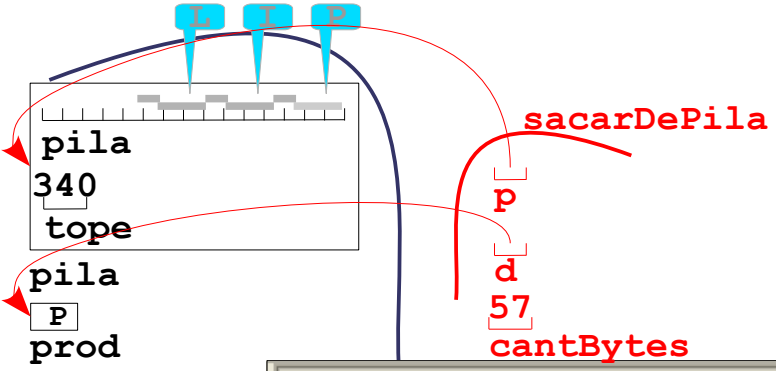




Tipo de dato Pila.

Implementación estática

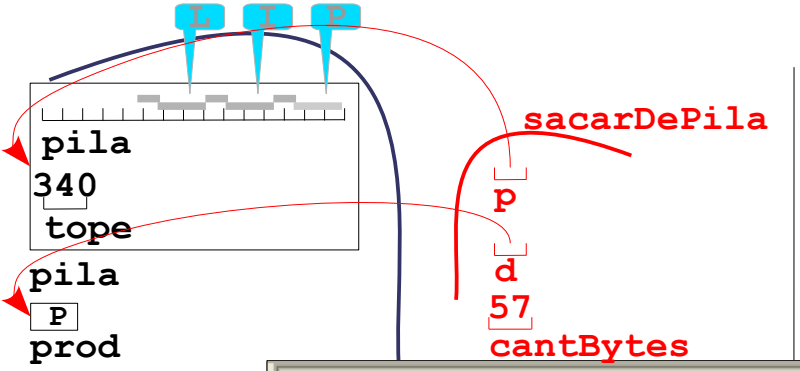
Implementación dinámica




```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
115     while(sacarDePila(&pila, &prod, sizeof(tProd)))
116         mostrarProducto(&prod);
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

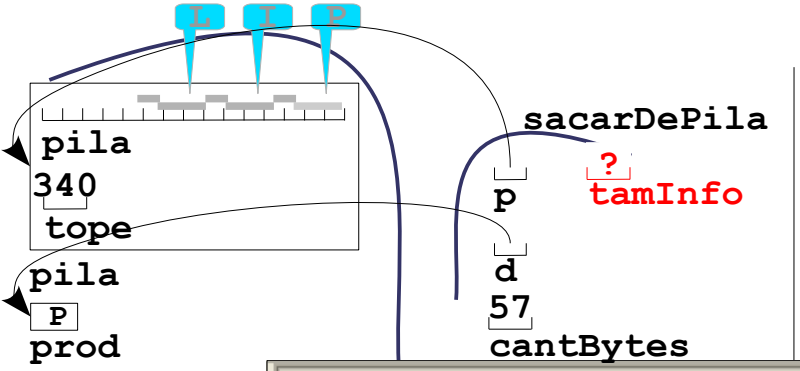


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

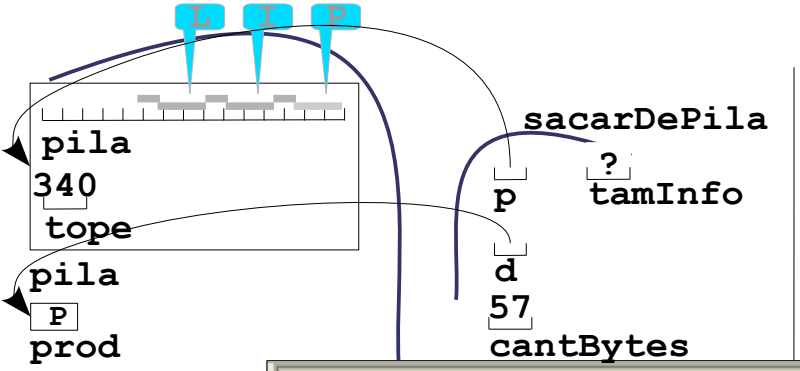


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

Implementación estática | Implementación dinámica

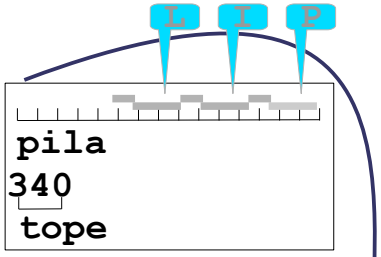


main.c | main.h | productos.c | productos.h | **pila.c** | pila.h | pila.c | pila.h

```
47  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
48  {
49      unsigned tamInfo;
50
51      if(p->tope == TAM_PILA)
52          return 0;
53      memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54      p->tope += sizeof(unsigned);
55      memcpy(d, p->pila + p->tope, minimo(cantBytes, tamInfo));
56      p->tope += tamInfo;
57      return 1;
58  }
```

Tipo de dato Pila.

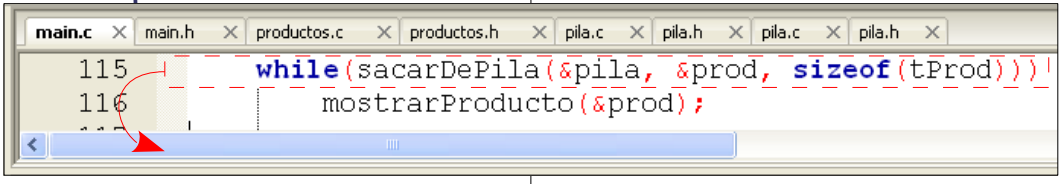
Implementación estática | Implementación dinámica



pila

P

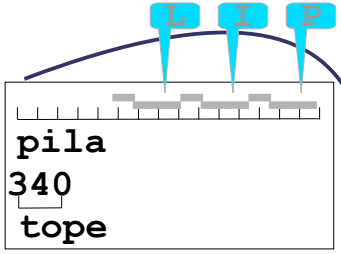
prod



Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

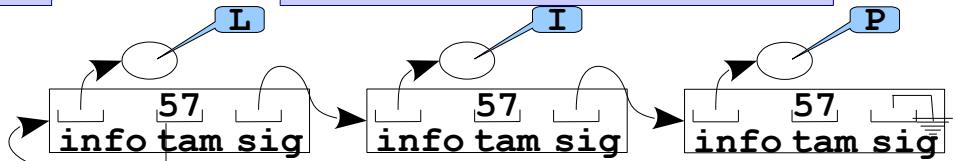


pila

P

prod

Implementación dinámica



pila

L

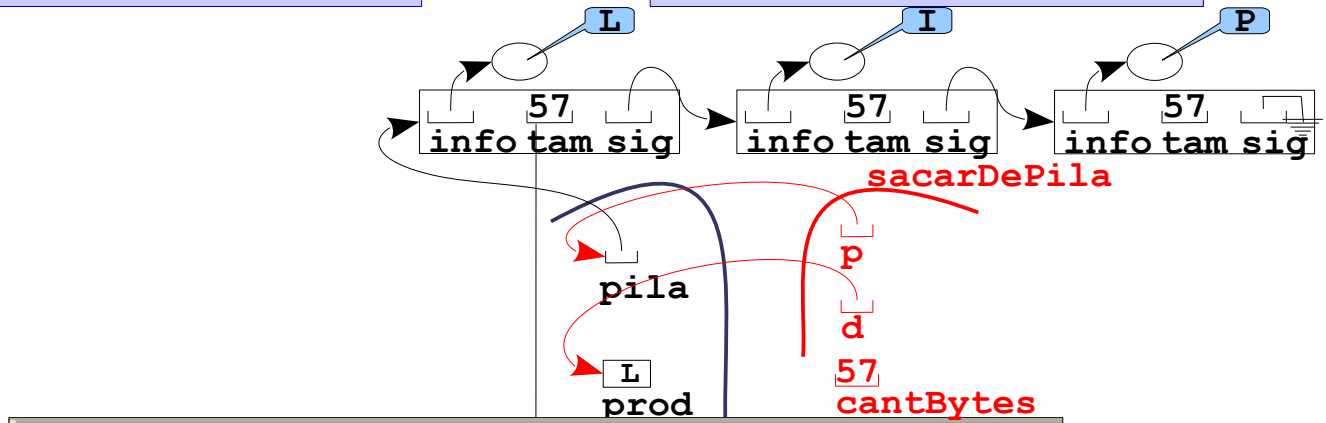
prod

```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
115     while(sacarDePila(&pila, &prod, sizeof(tProd)))
116         mostrarProducto(&prod);
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica

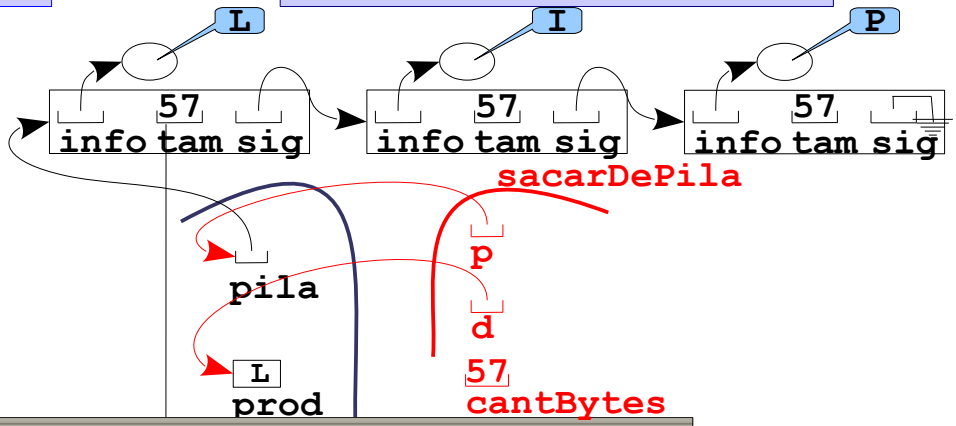



```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
115     while(sacarDePila(&pila, &prod, sizeof(tProd)))
116         mostrarProducto(&prod);
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica

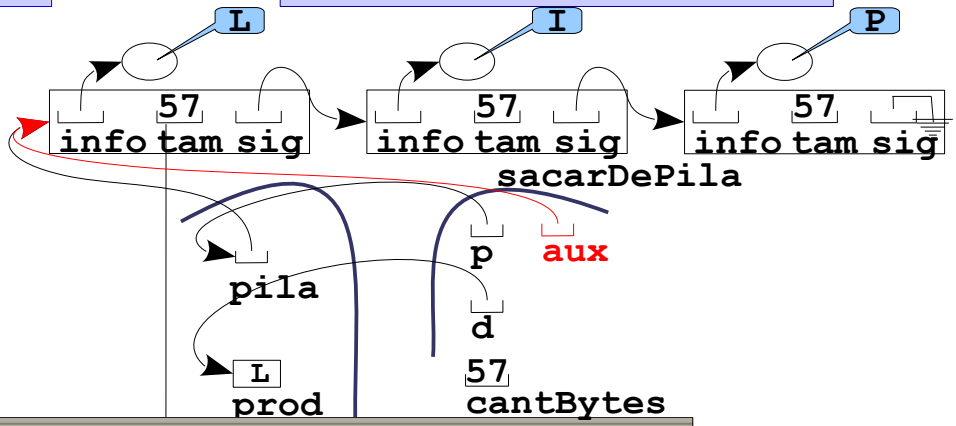



```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

115
116

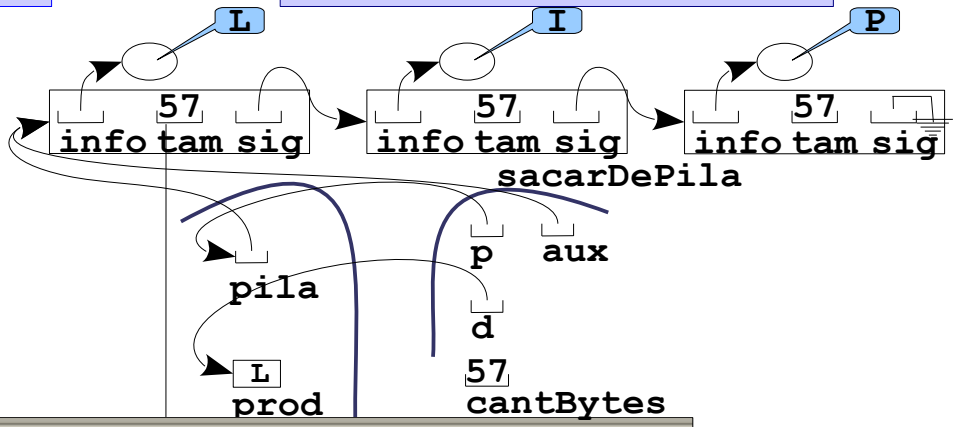
main.c | main.h | productos.c | productos.h | pila.c | pila.h | *pila.c | pila.h

```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
115
116
```

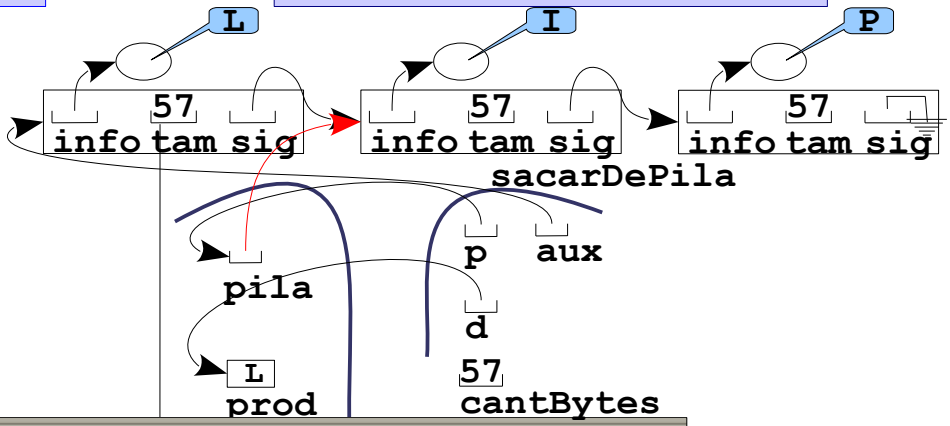
main.c | main.h | productos.c | productos.h | pila.c | pila.h | *pila.c | pila.h

```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
115
116
```

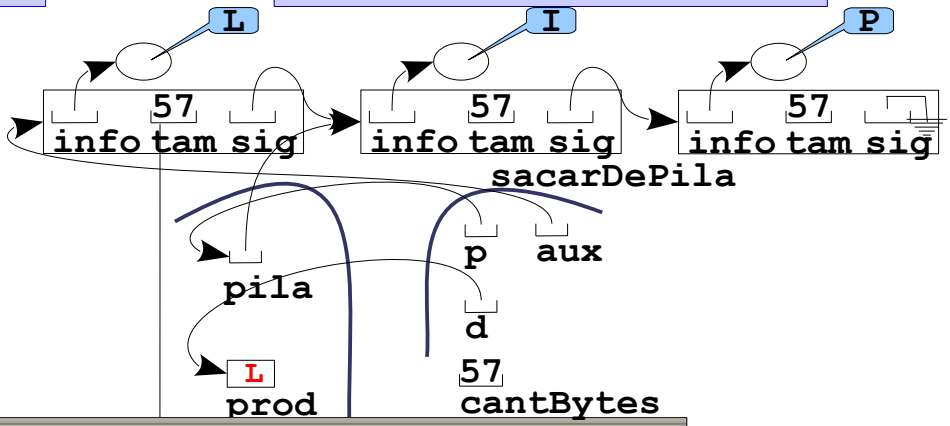
main.c | main.h | productos.c | productos.h | pila.c | pila.h | *pila.c | pila.h

```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
115
116
```

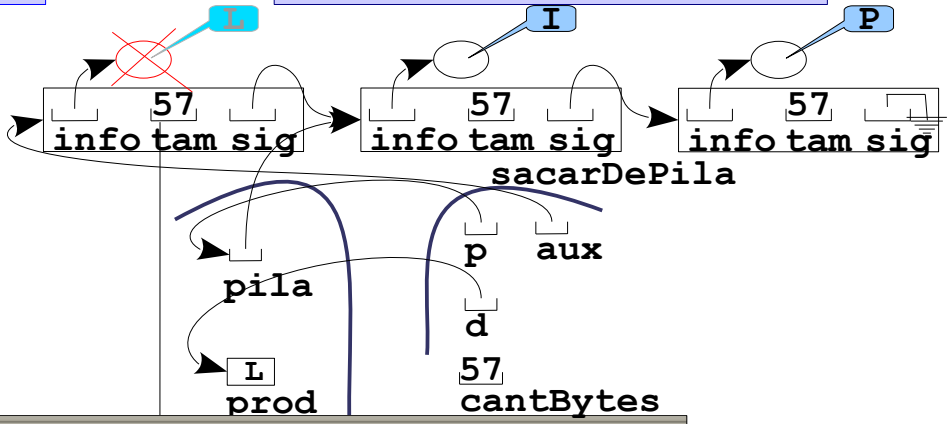
main.c | main.h | productos.c | productos.h | pila.c | pila.h | *pila.c | pila.h

```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
115
116
```

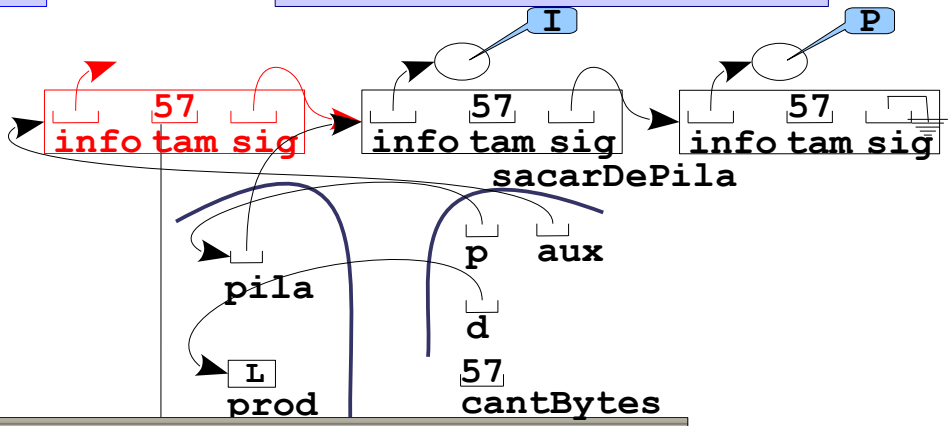
main.c | main.h | productos.c | productos.h | pila.c | pila.h | *pila.c | pila.h

```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

115
116

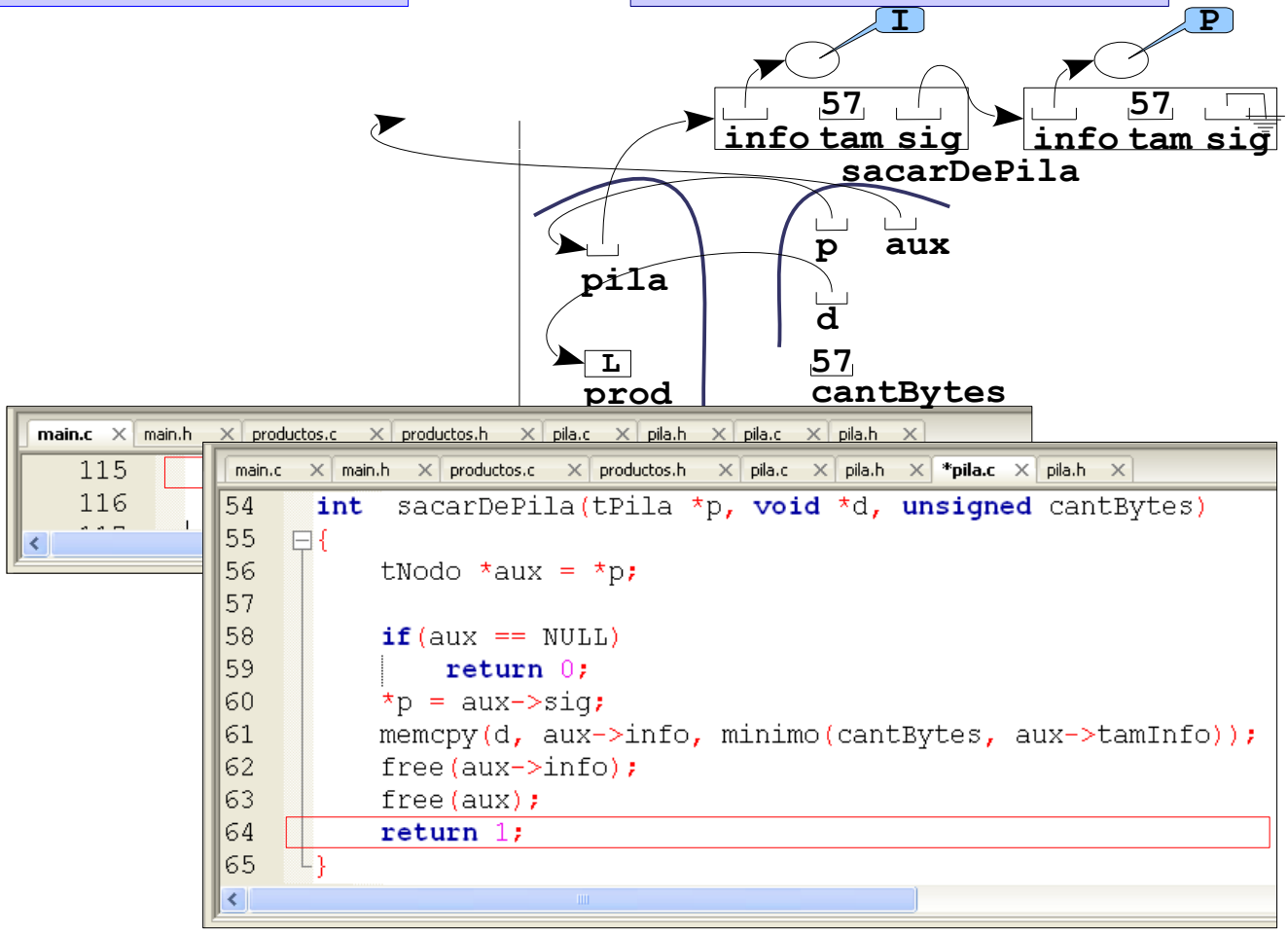
main.c | main.h | productos.c | productos.h | pila.c | pila.h | *pila.c | pila.h

```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica

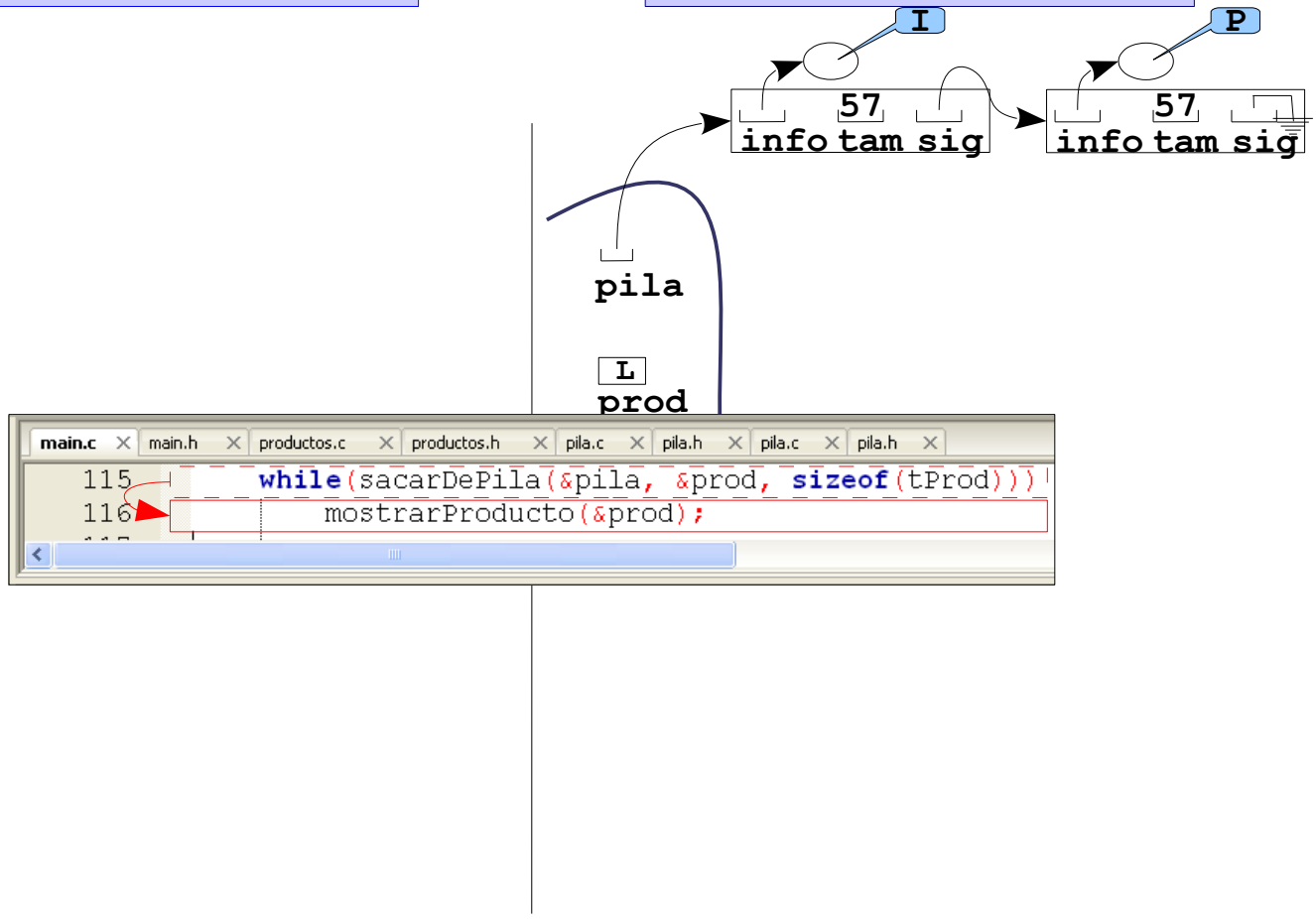



```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

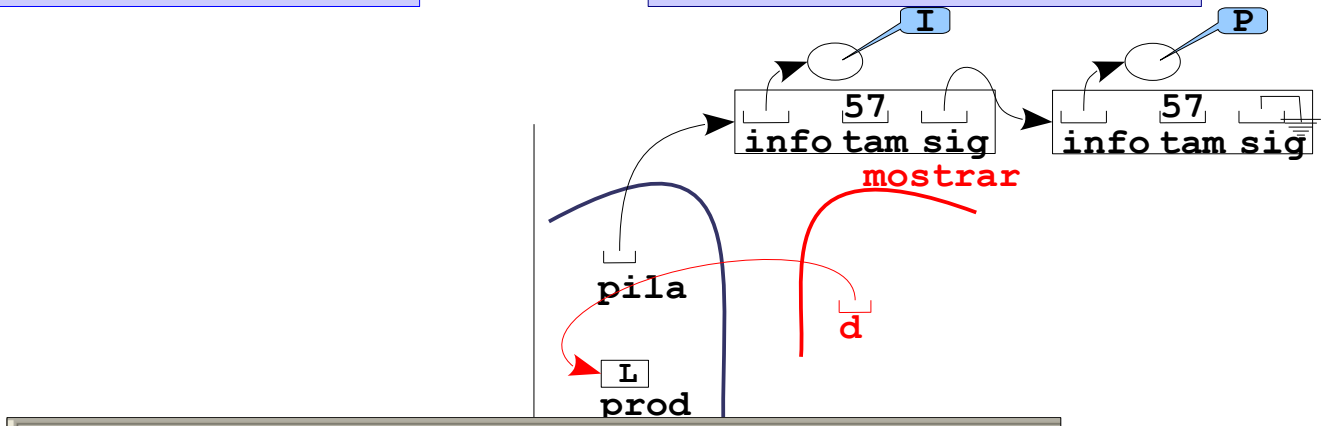
Implementación dinámica



Tipo de dato Pila.

Implementación estática

Implementación dinámica

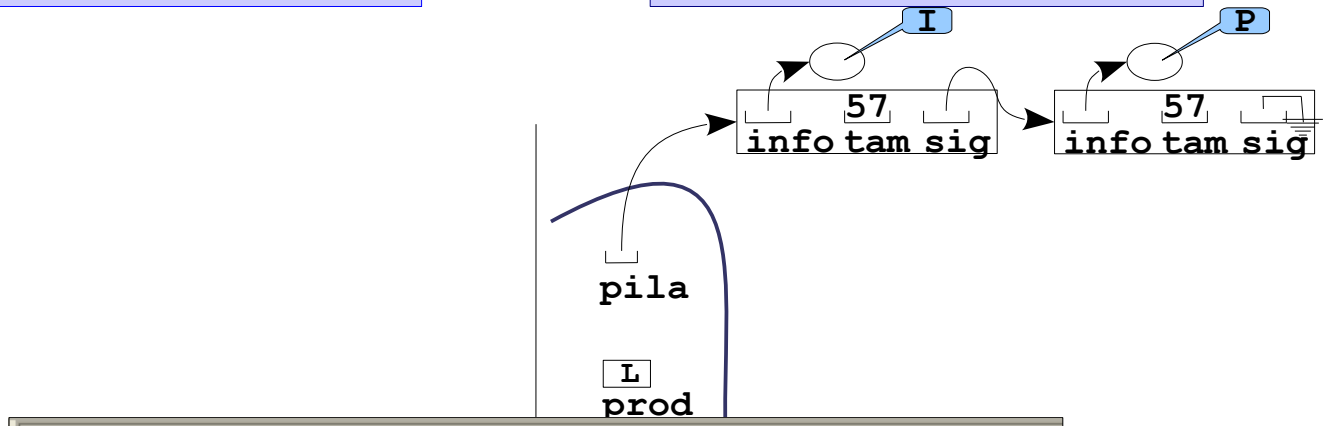



```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
115     while(sacarDePila(&pila, &prod, sizeof(tProd)))
116         mostrarProducto(&prod);
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica

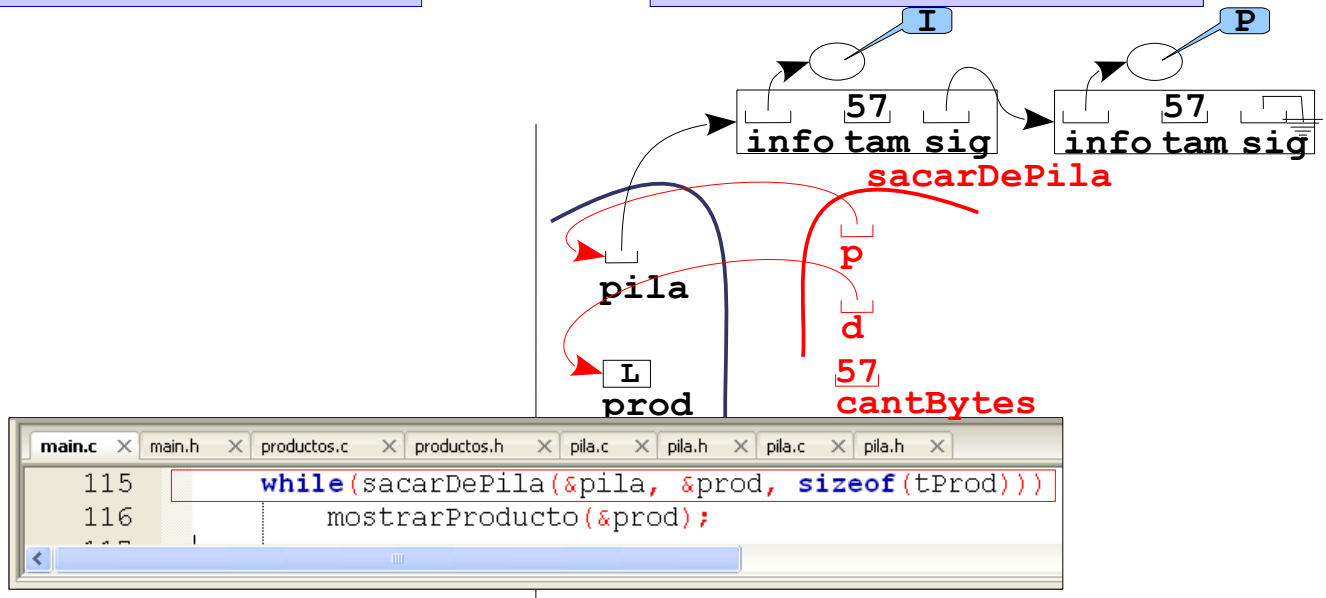



```
main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h
115     while(sacarDePila(&pila, &prod, sizeof(tProd)))
116         mostrarProducto(&prod);
```

Tipo de dato Pila.

Implementación estática

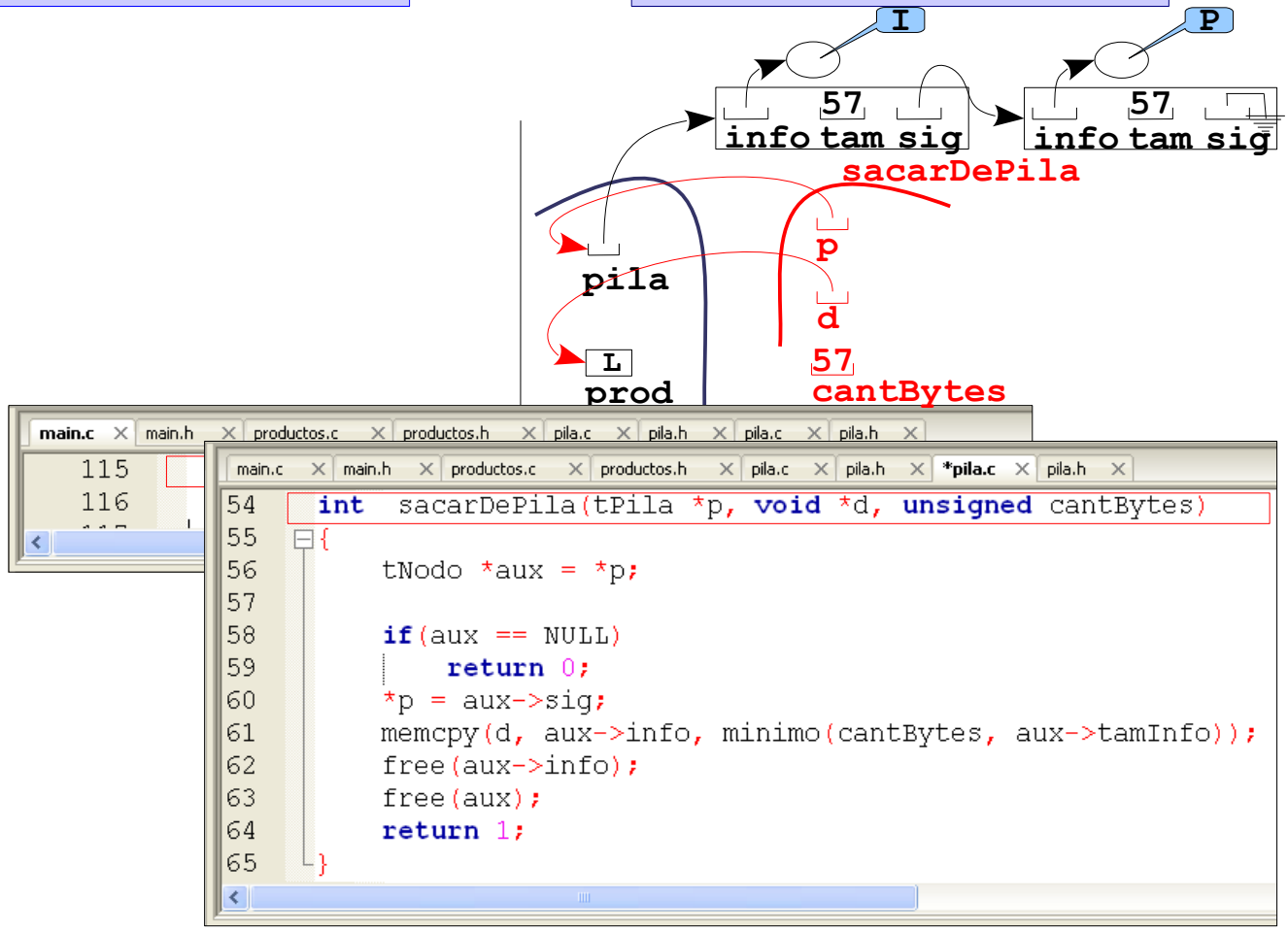
Implementación dinámica



Tipo de dato Pila.

Implementación estática

Implementación dinámica

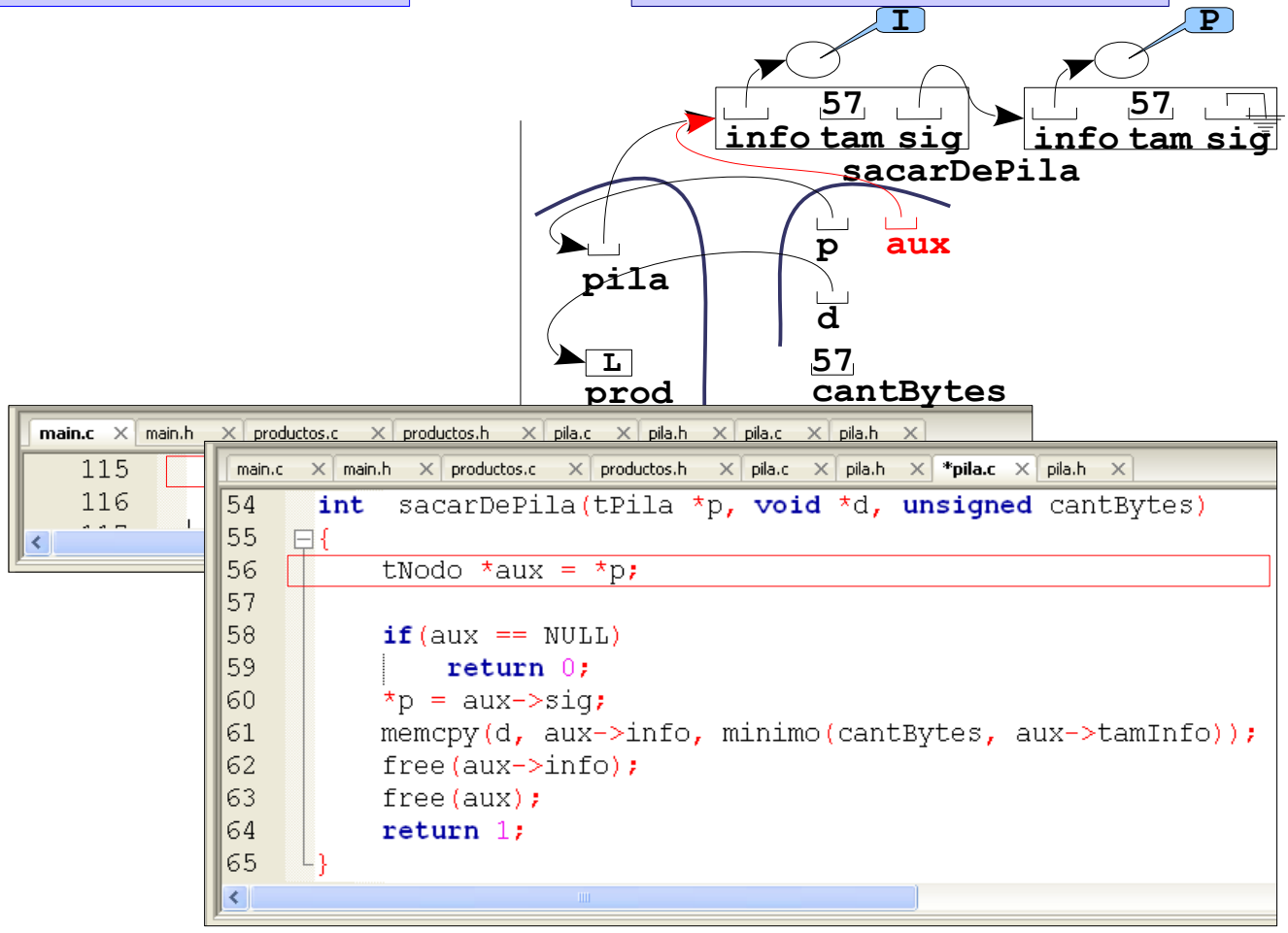



```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica

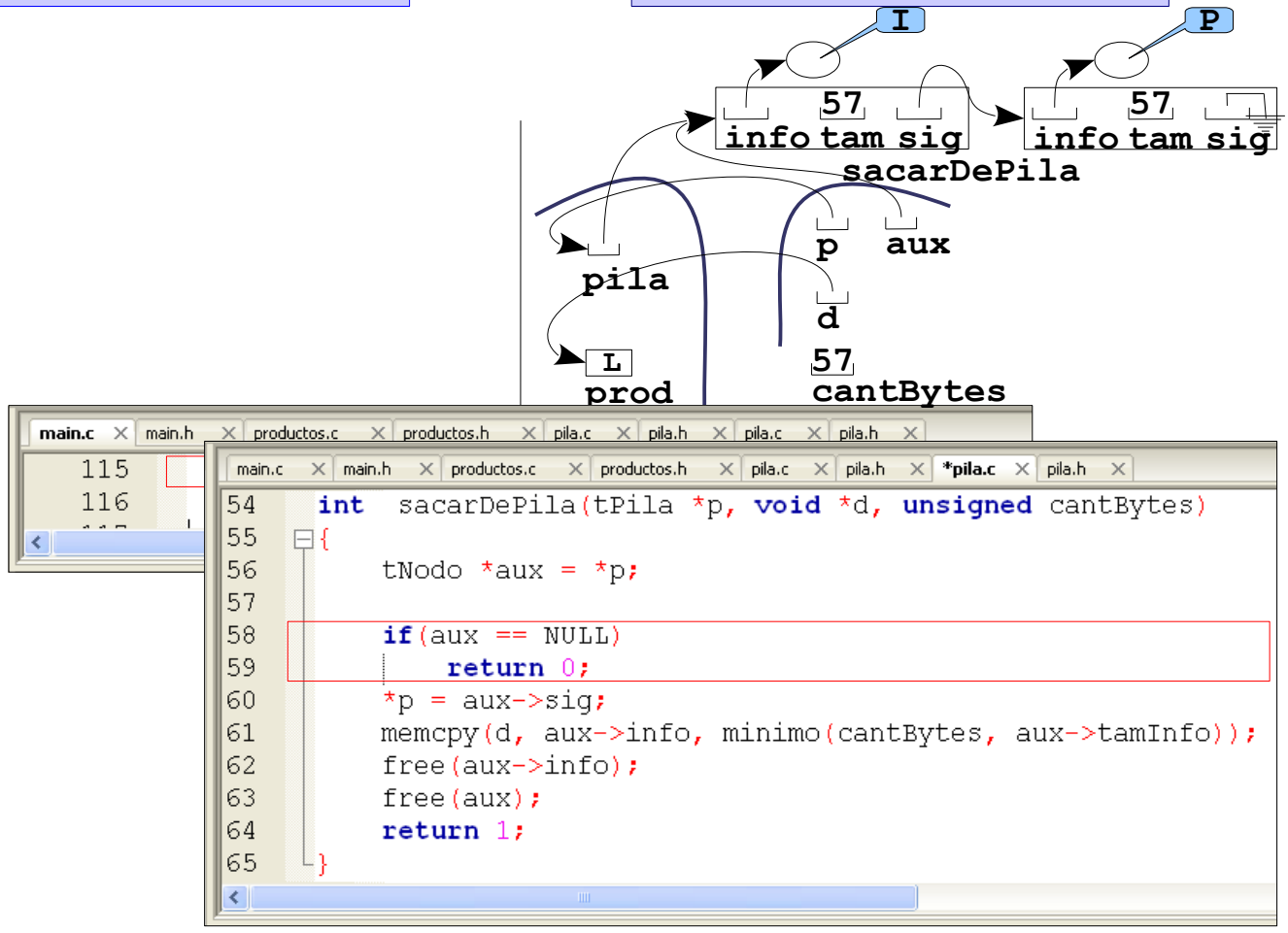



```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

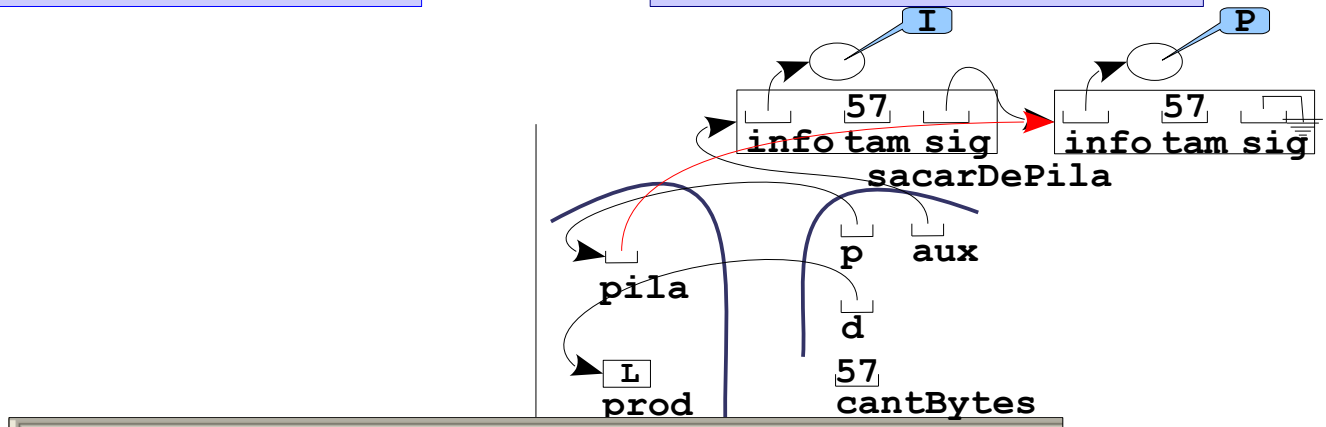
Implementación dinámica



Tipo de dato Pila.

Implementación estática

Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

115
116

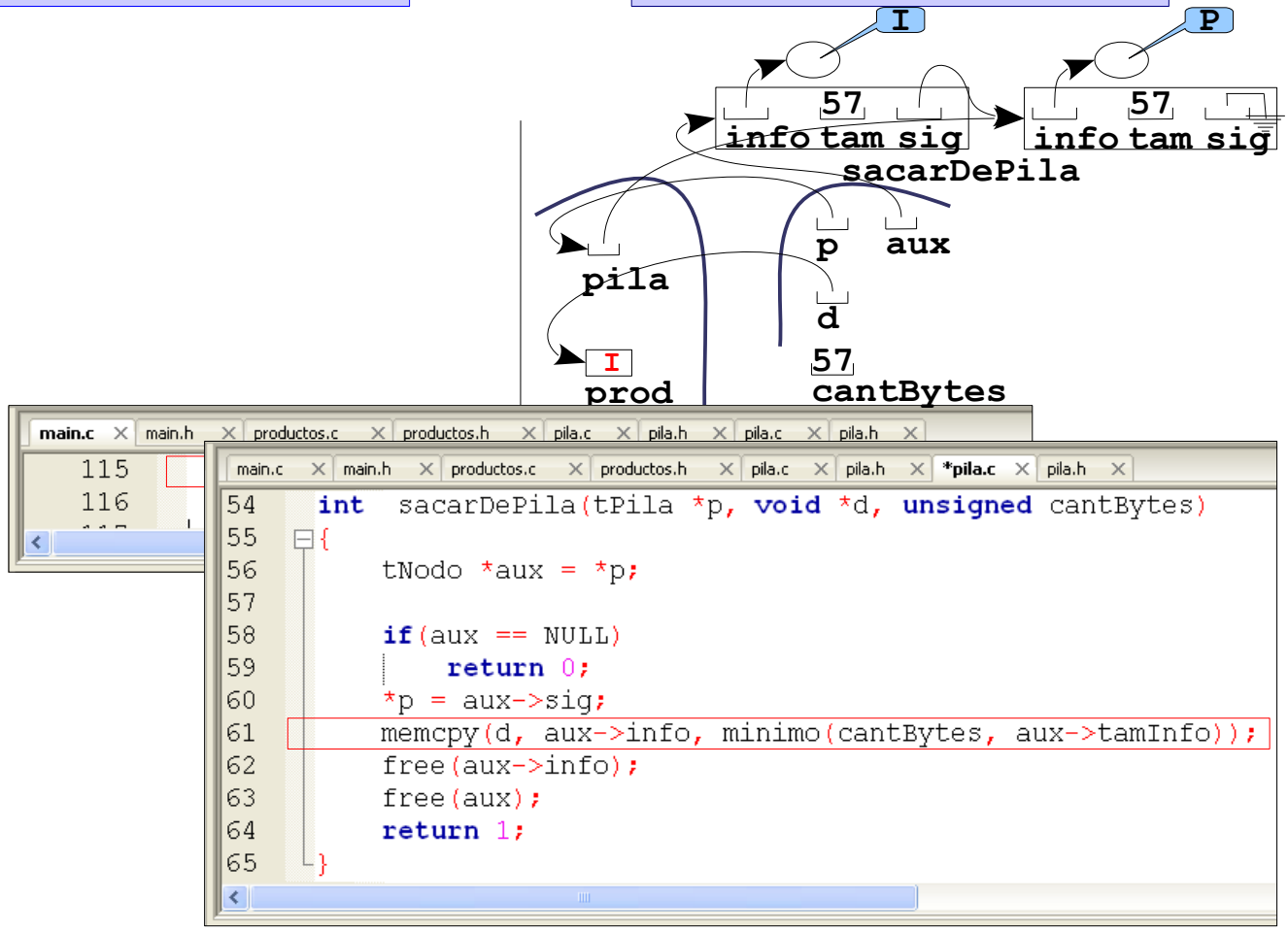
main.c | main.h | productos.c | productos.h | pila.c | pila.h | *pila.c | pila.h

```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

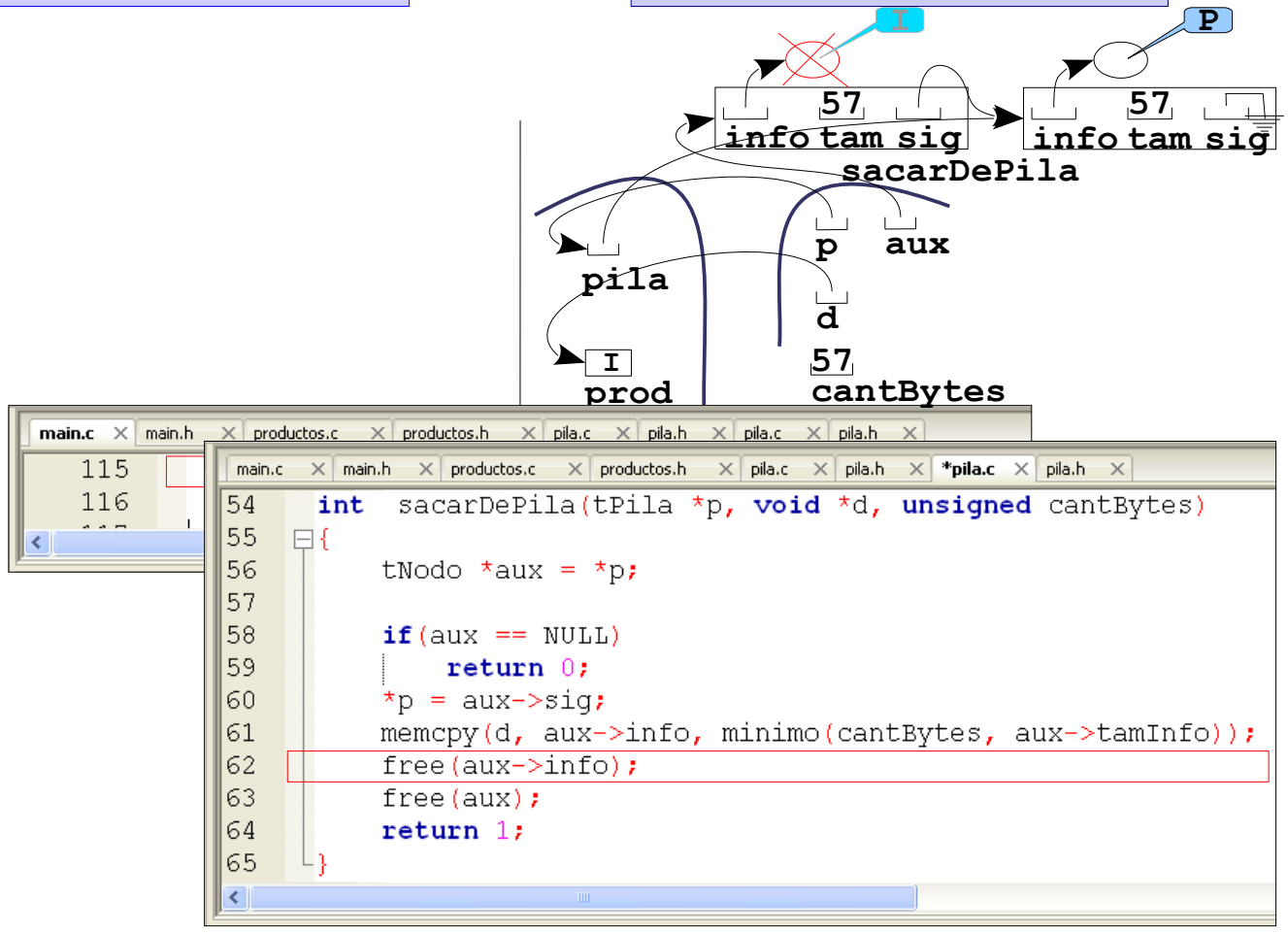
Implementación dinámica



Tipo de dato Pila.

Implementación estática

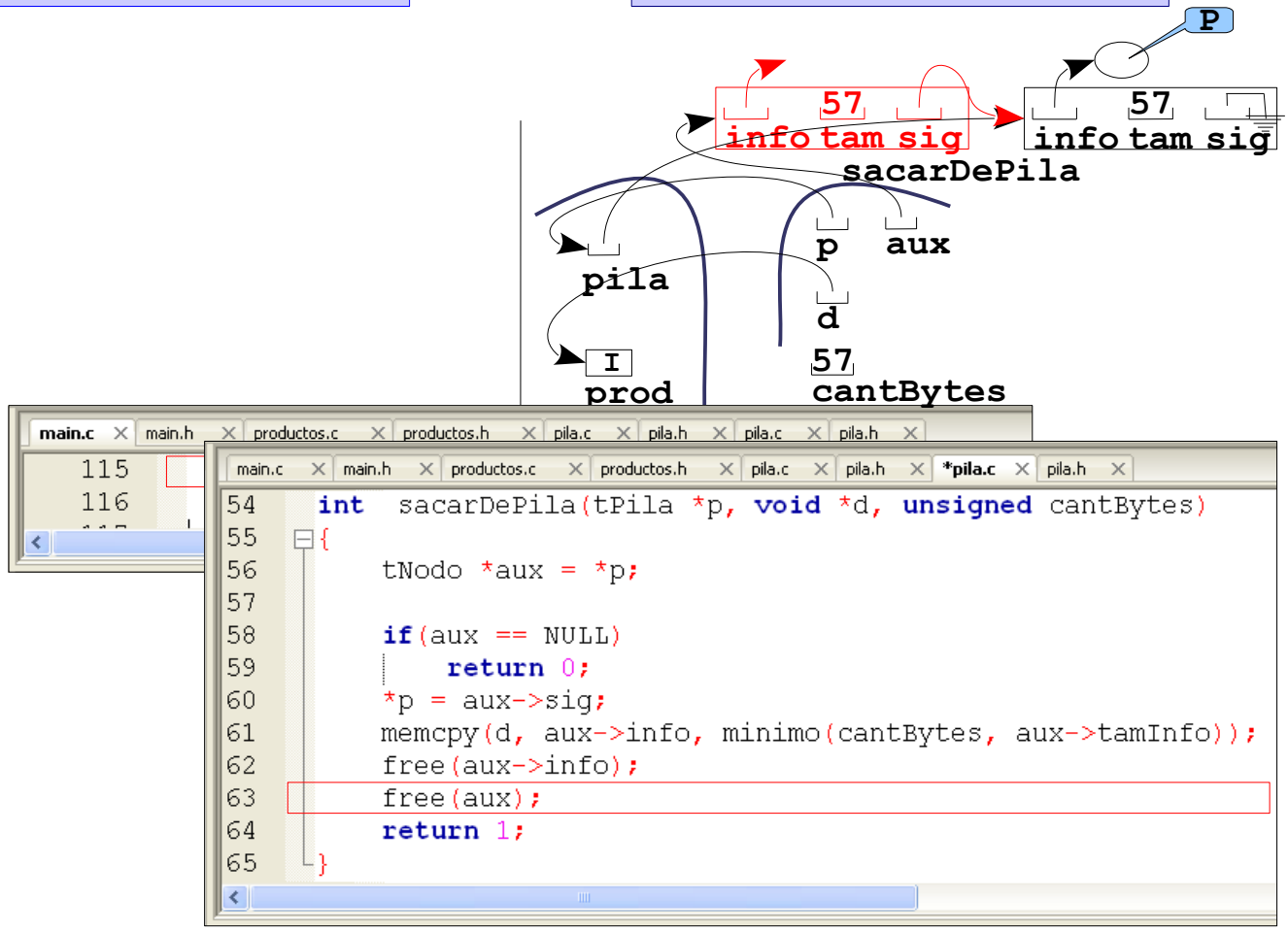
Implementación dinámica



Tipo de dato Pila.

Implementación estática

Implementación dinámica

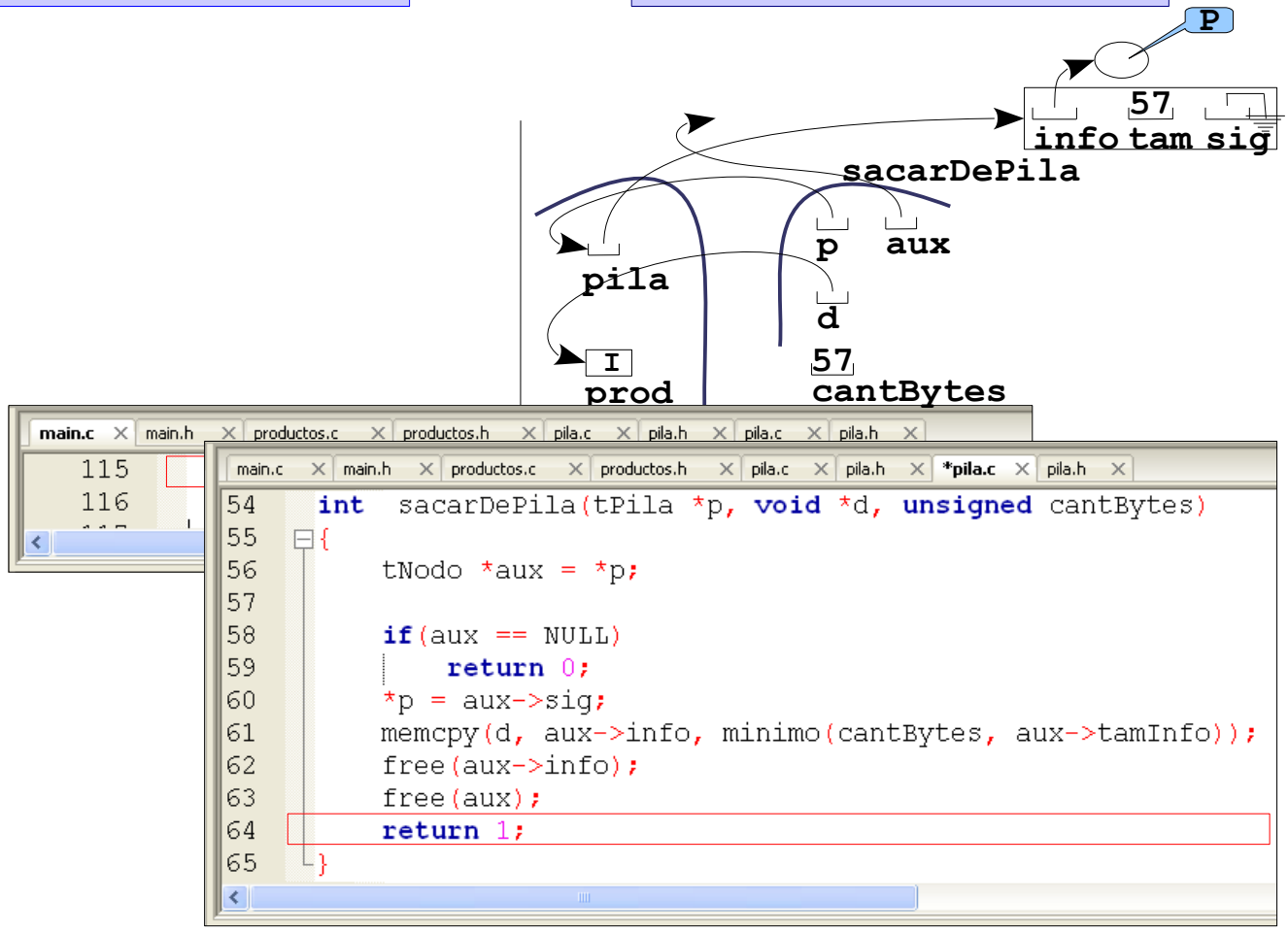



```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

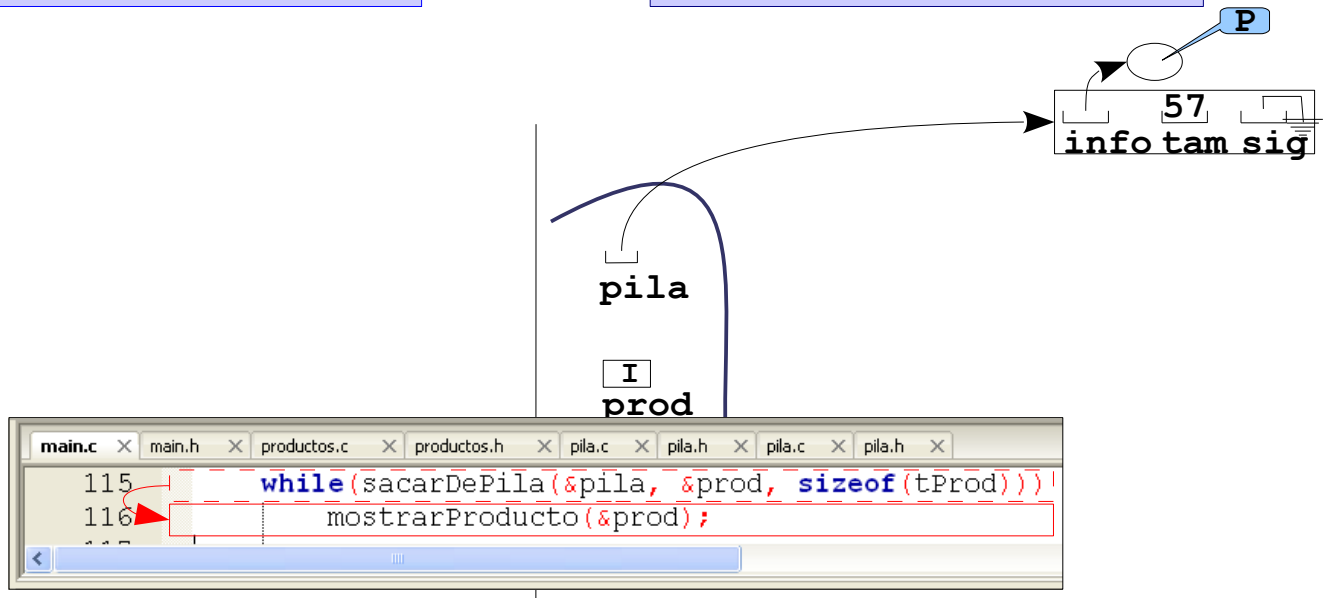
Implementación dinámica



Tipo de dato Pila.

Implementación estática

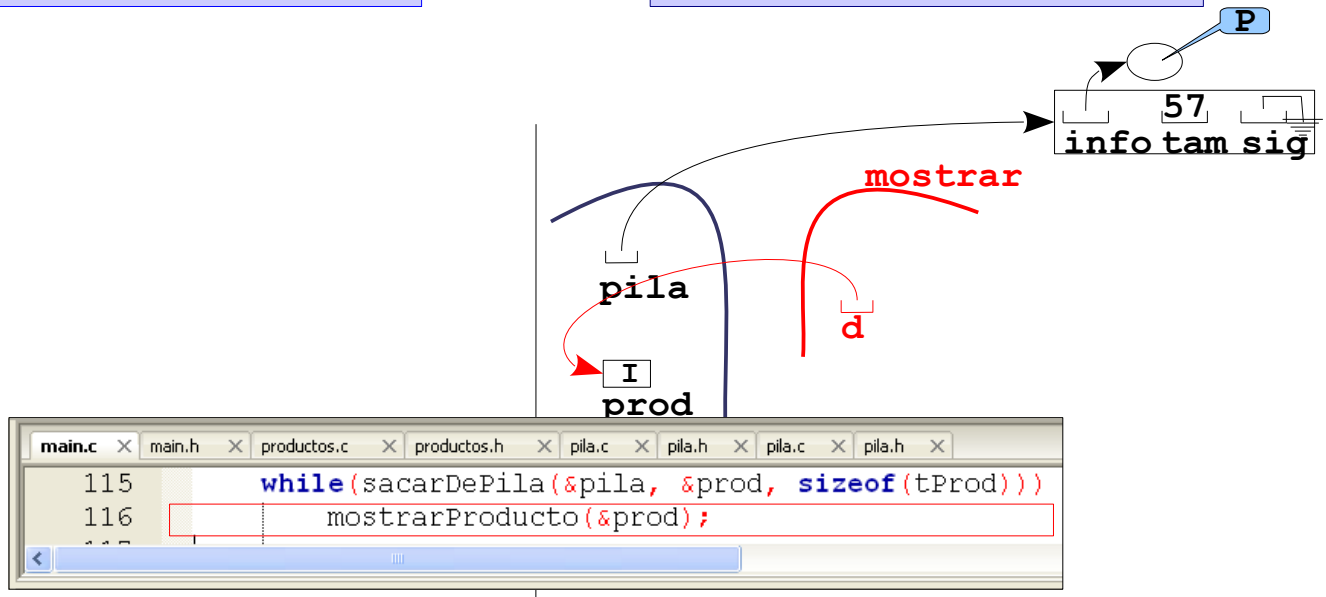
Implementación dinámica



Tipo de dato Pila.

Implementación estática

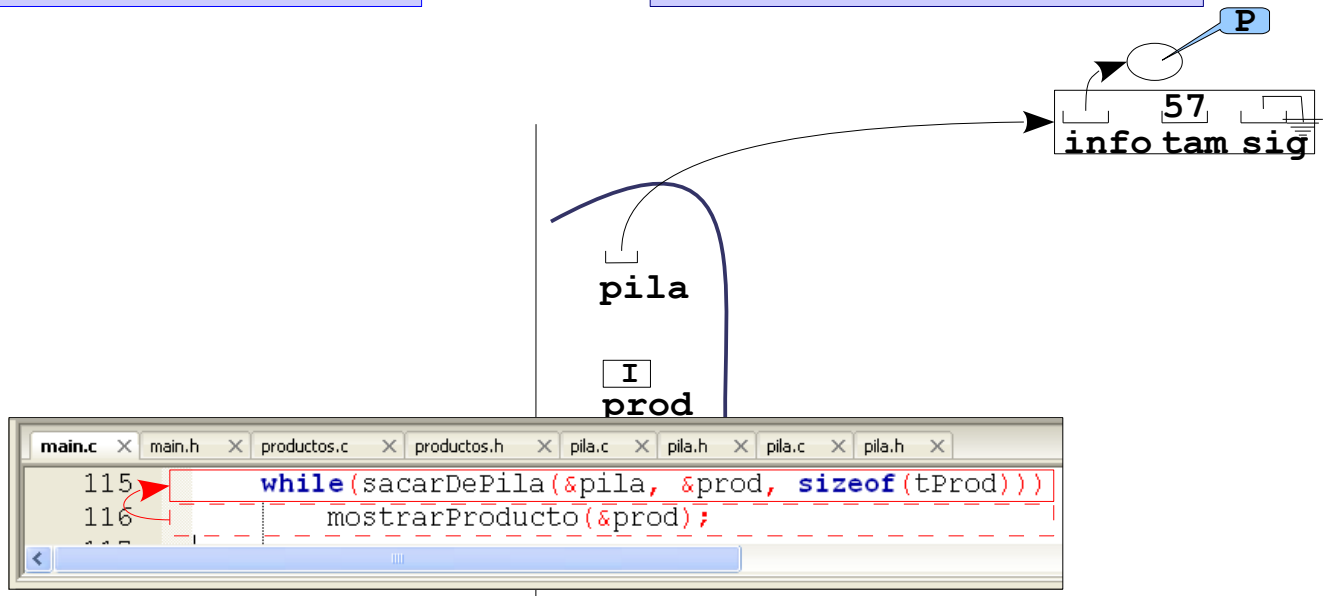
Implementación dinámica



Tipo de dato Pila.

Implementación estática

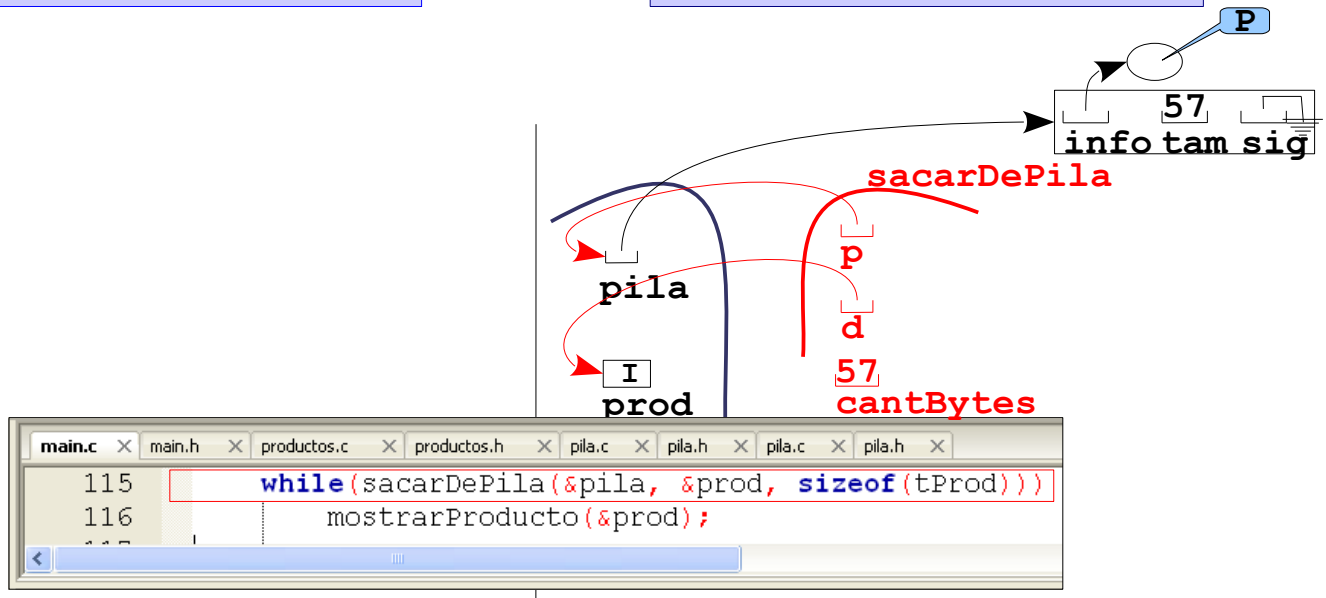
Implementación dinámica



Tipo de dato Pila.

Implementación estática

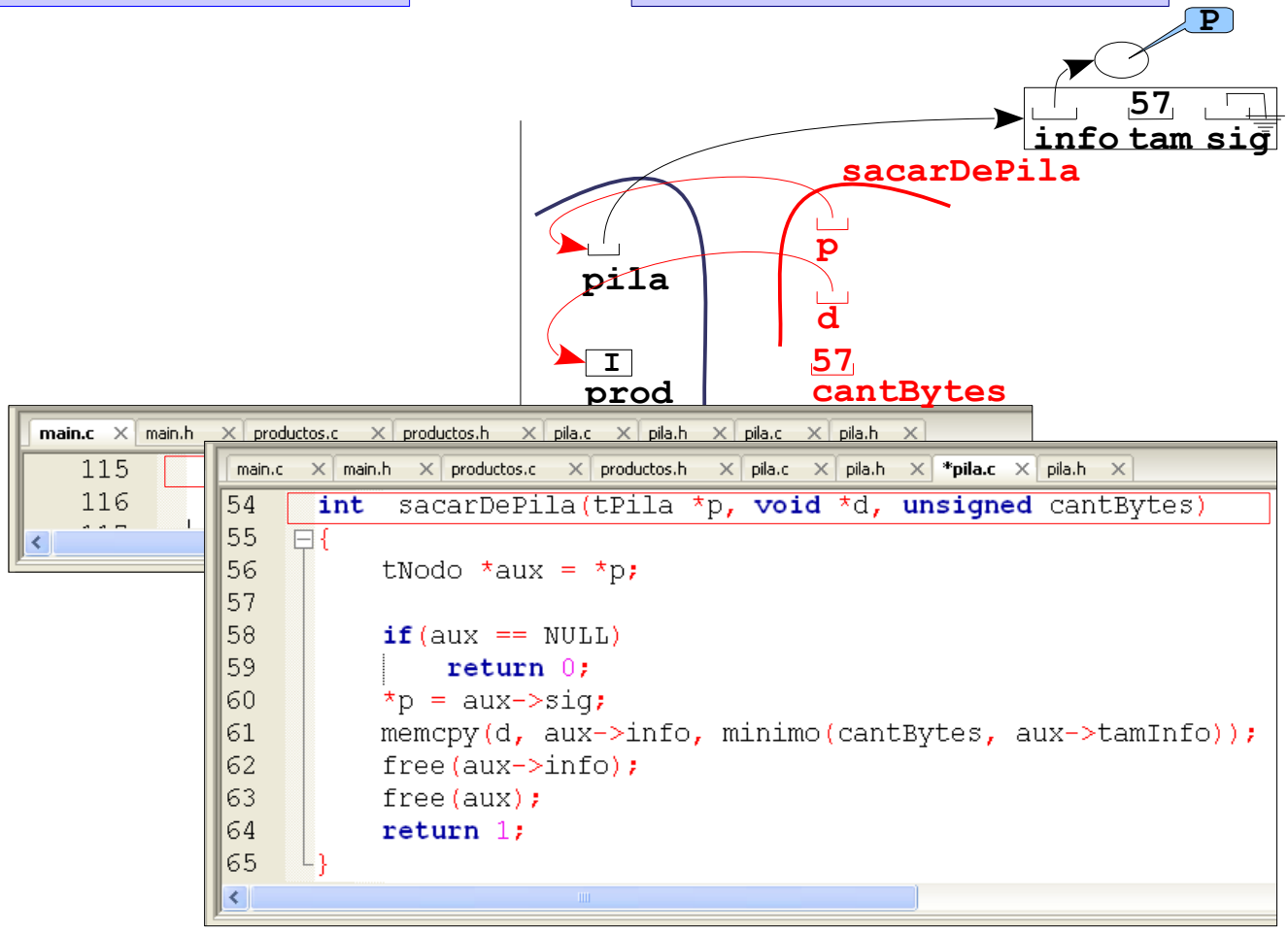
Implementación dinámica



Tipo de dato Pila.

Implementación estática

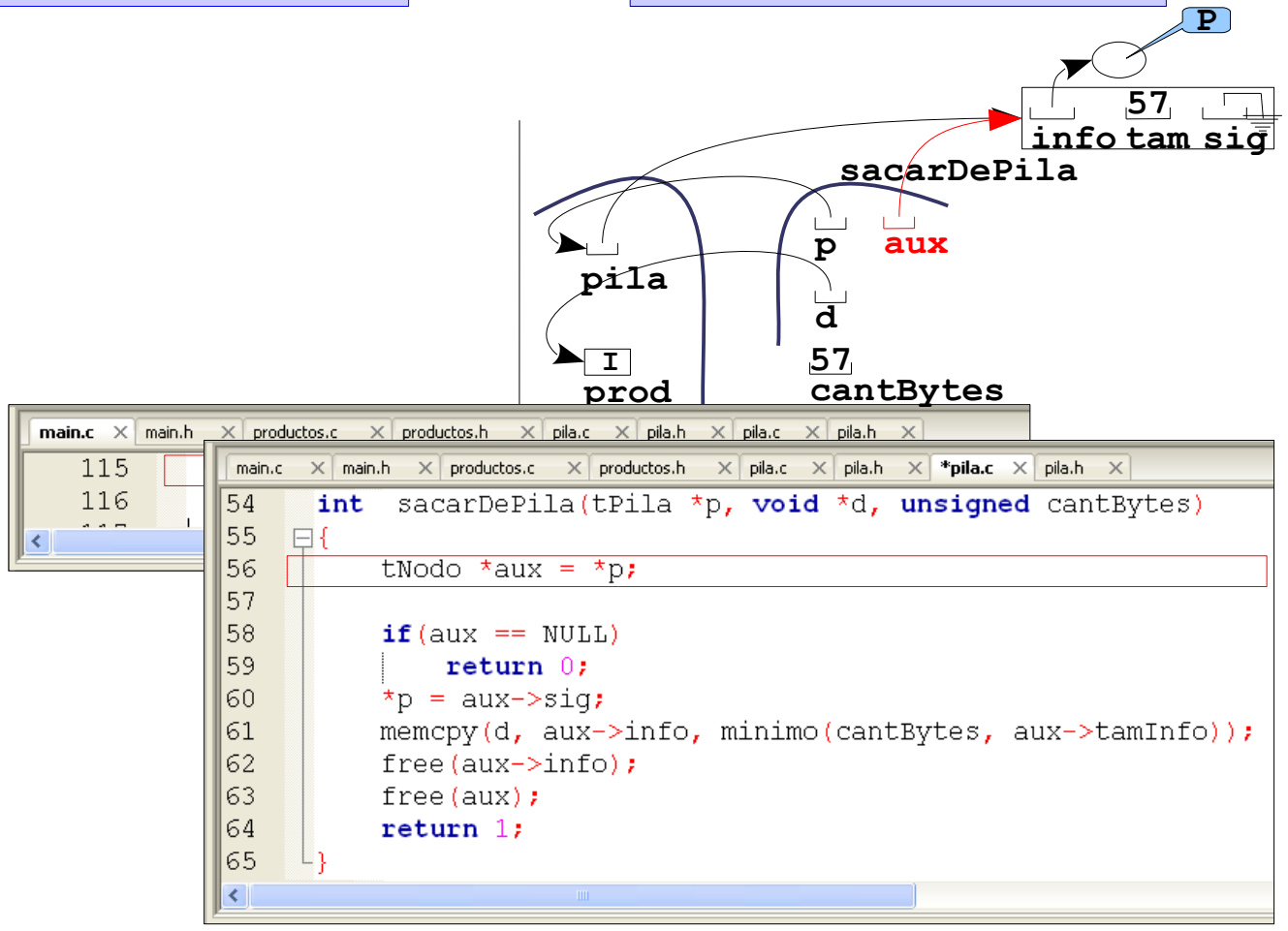
Implementación dinámica



Tipo de dato Pila.

Implementación estática

Implementación dinámica

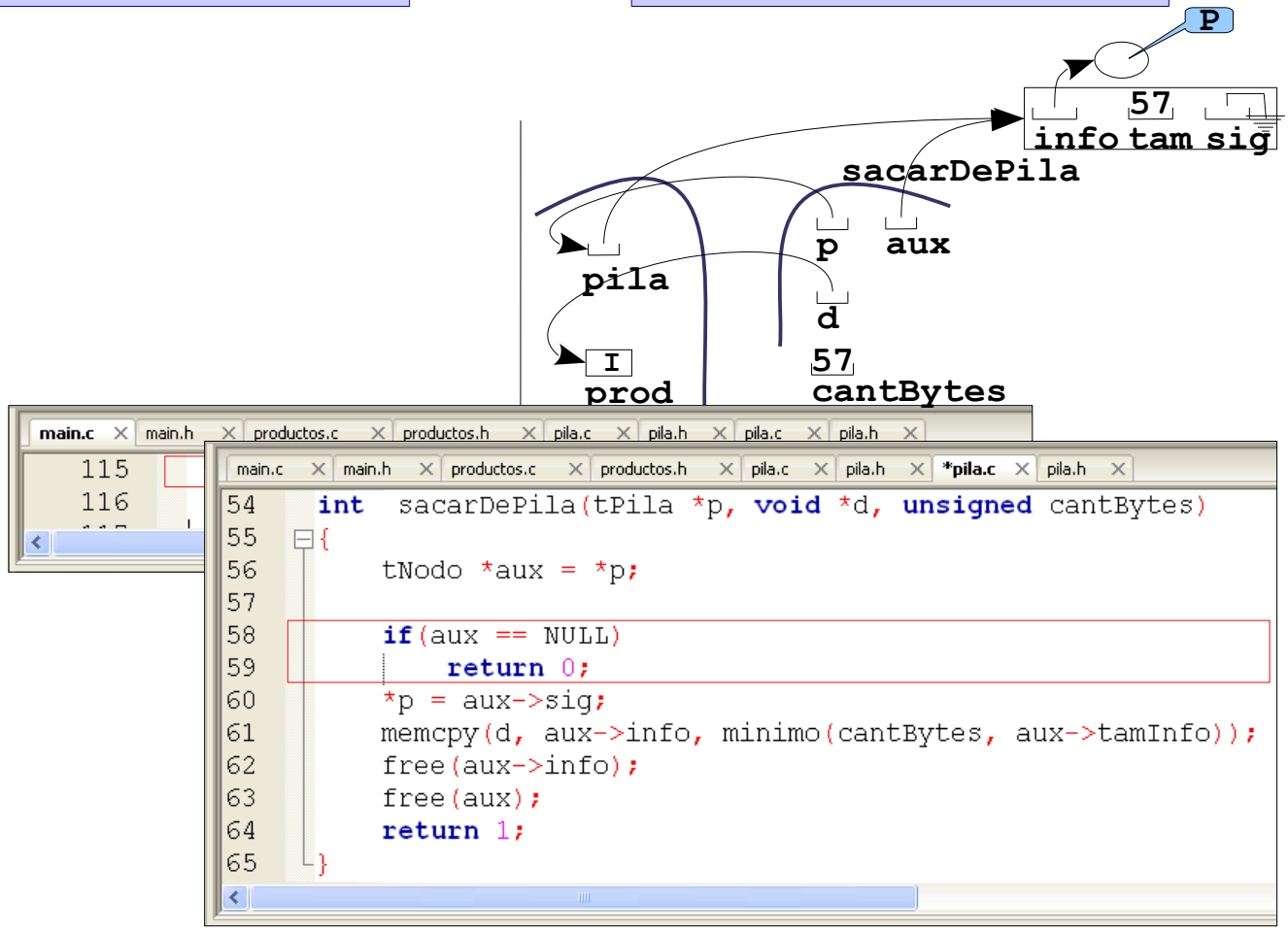



```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

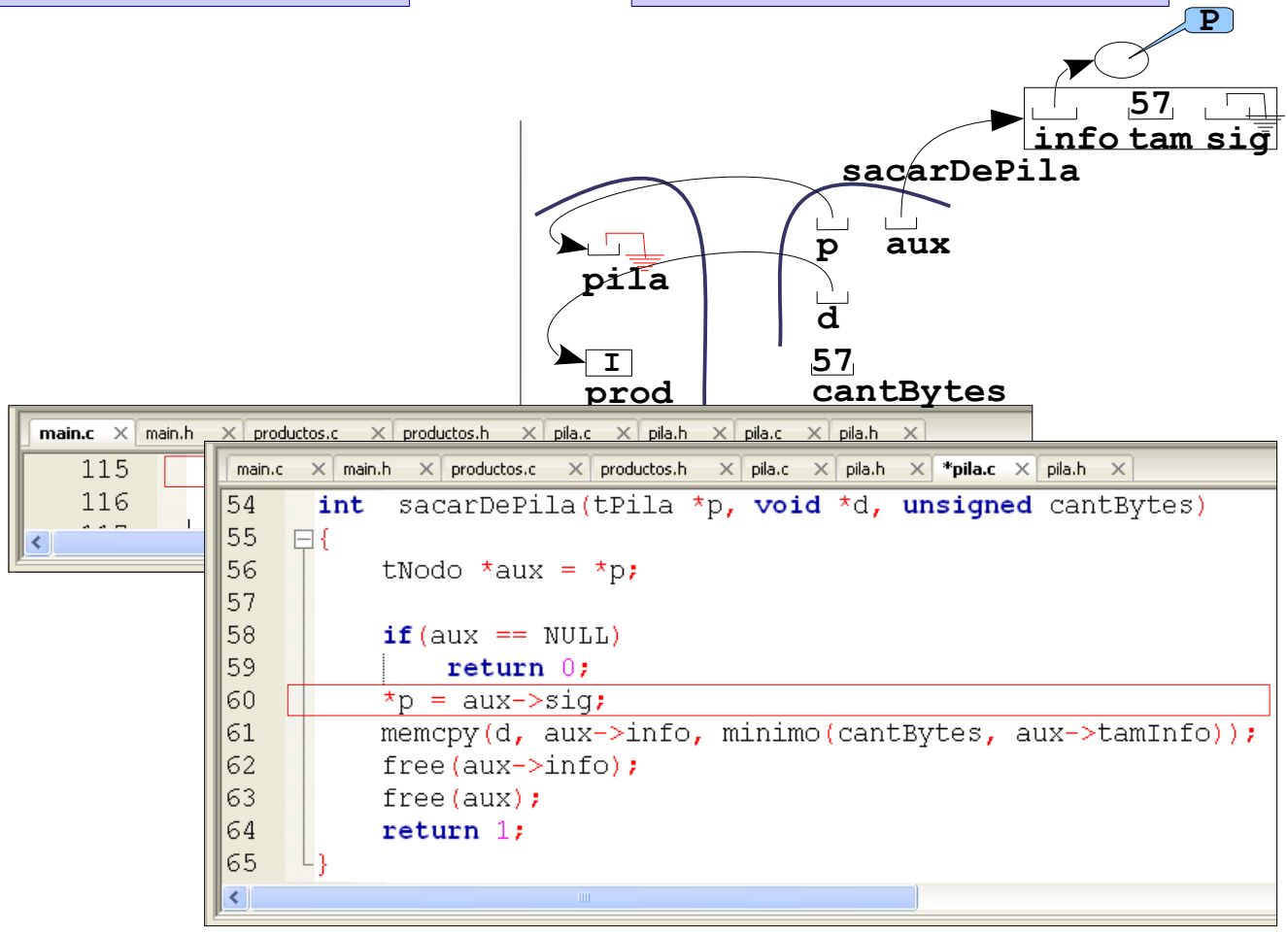
Implementación dinámica



Tipo de dato Pila.

Implementación estática

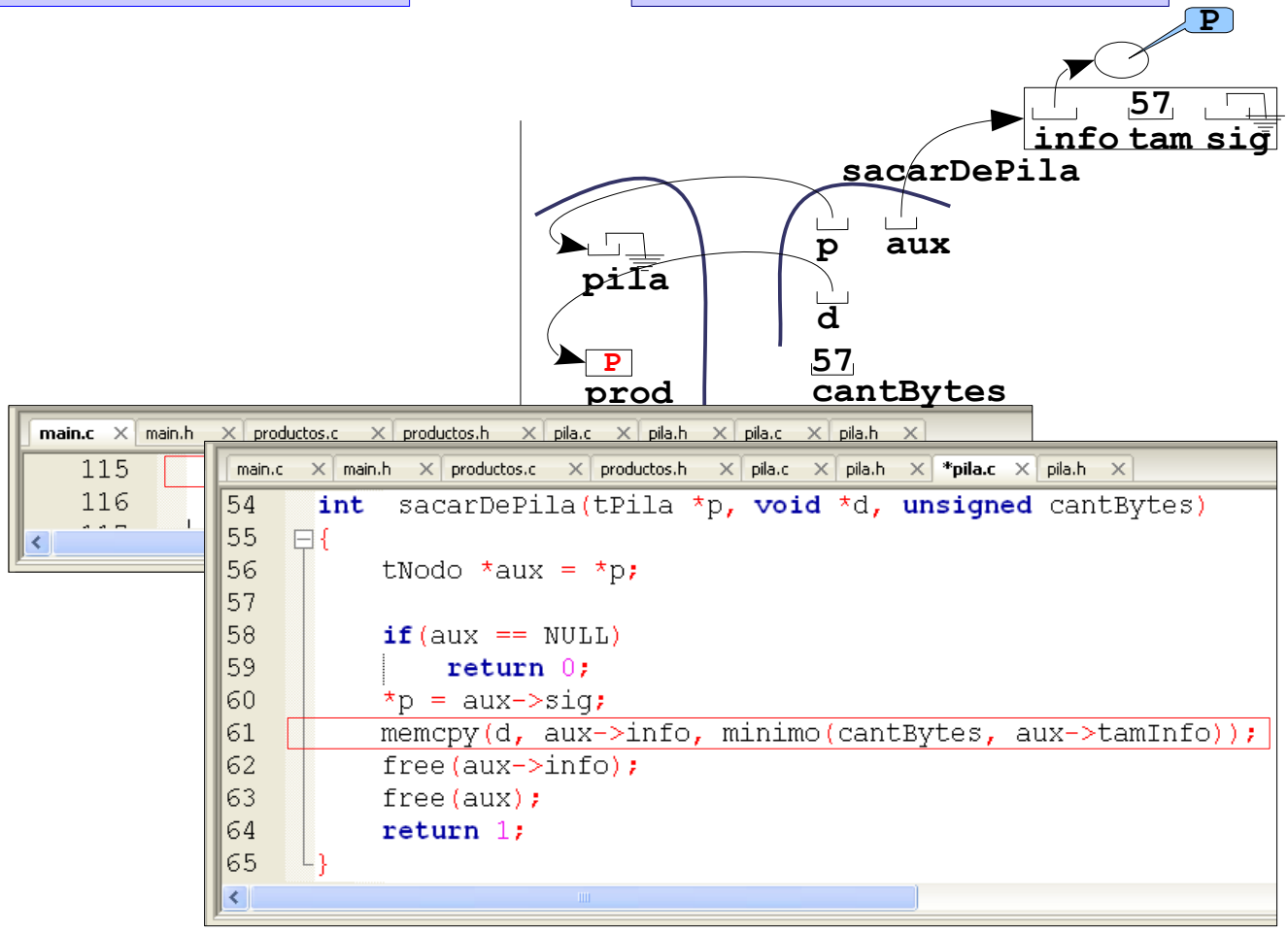
Implementación dinámica



Tipo de dato Pila.

Implementación estática

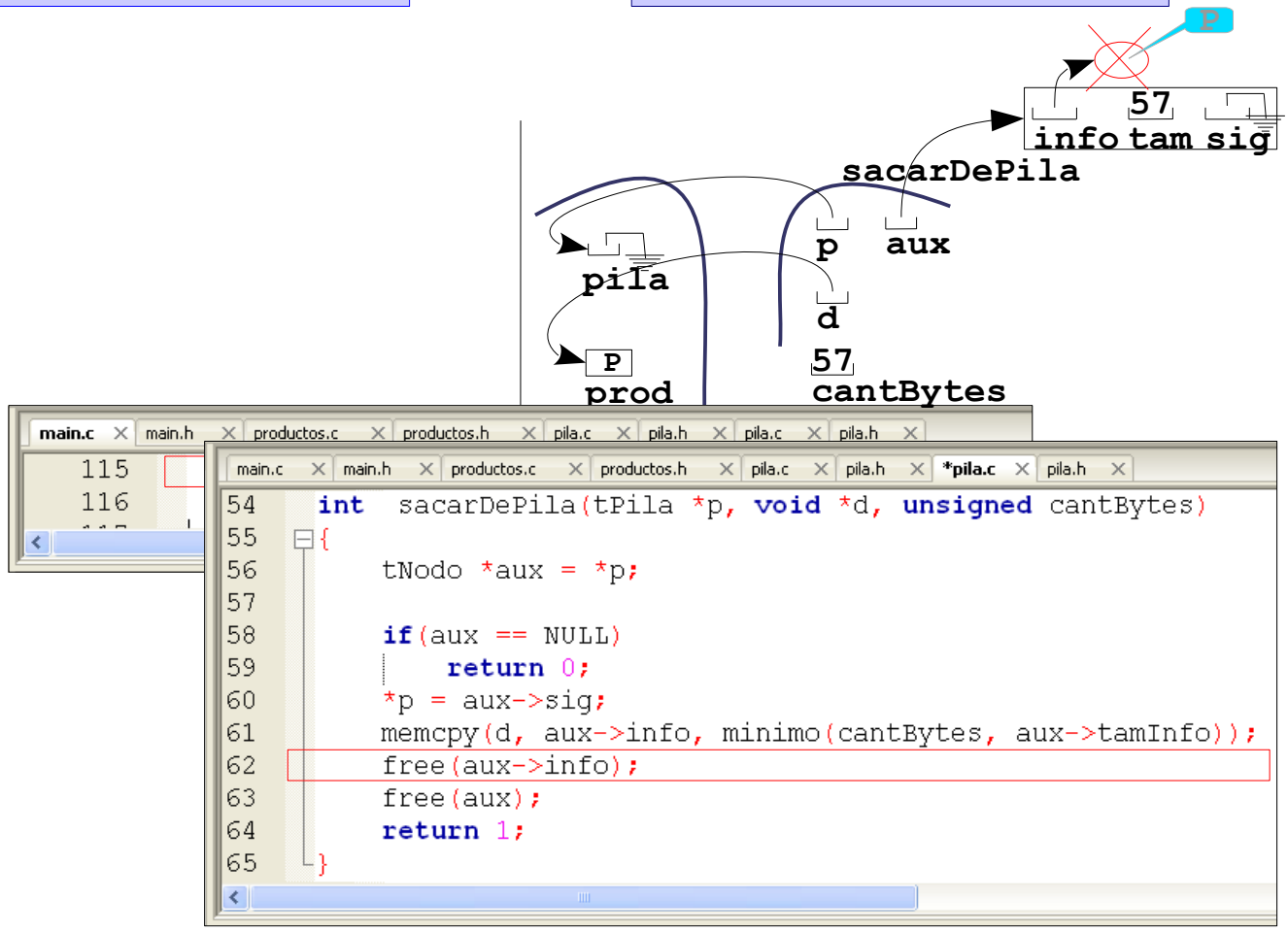
Implementación dinámica



Tipo de dato Pila.

Implementación estática

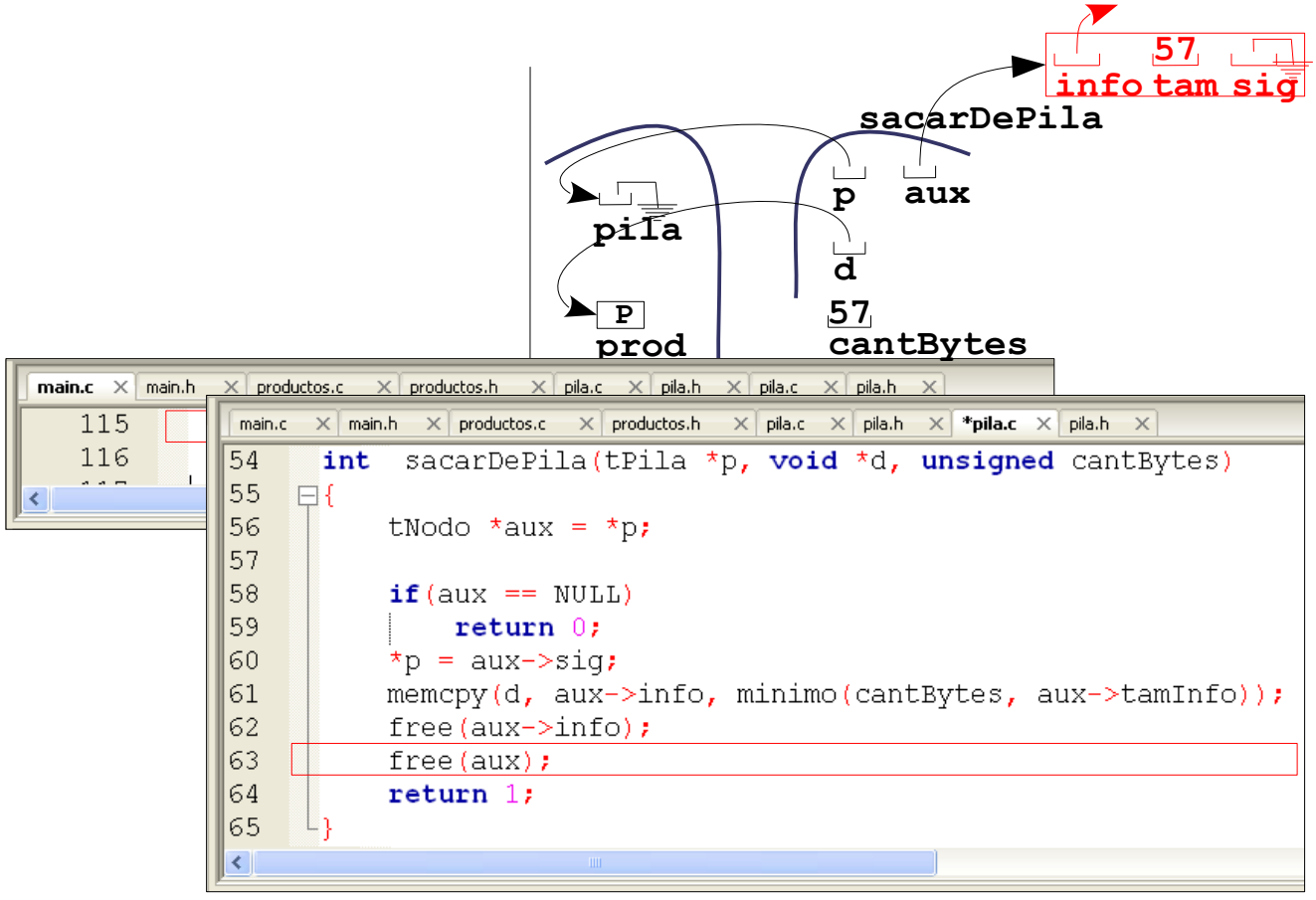
Implementación dinámica



Tipo de dato Pila.

Implementación estática

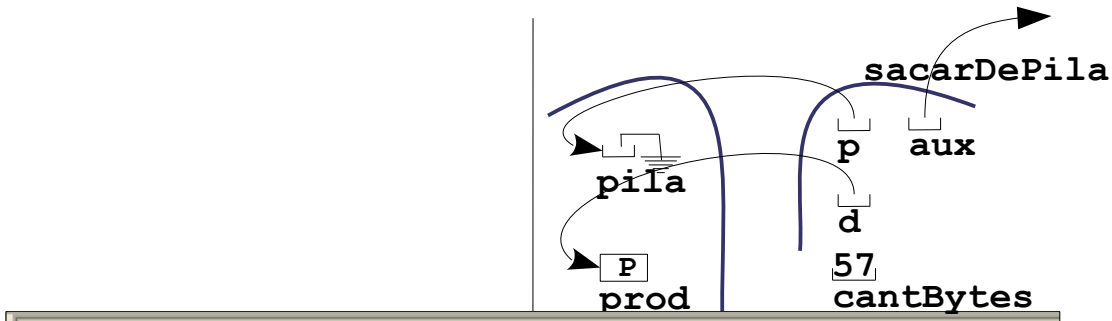
Implementación dinámica



Tipo de dato Pila.

Implementación estática

Implementación dinámica

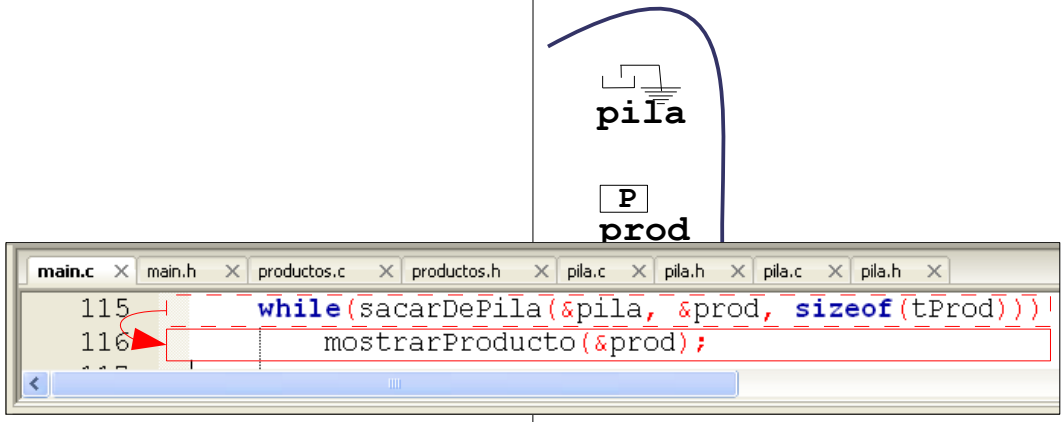



```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

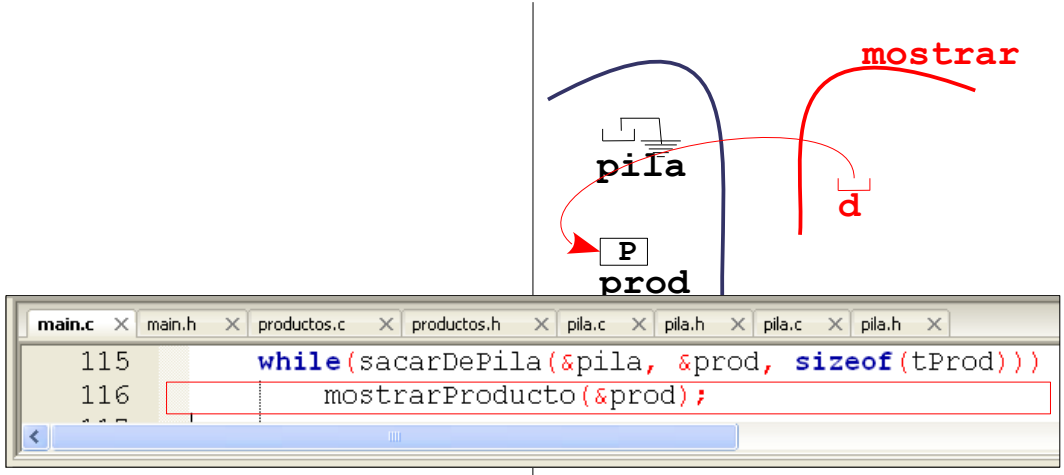
Implementación dinámica



Tipo de dato Pila.

Implementación estática

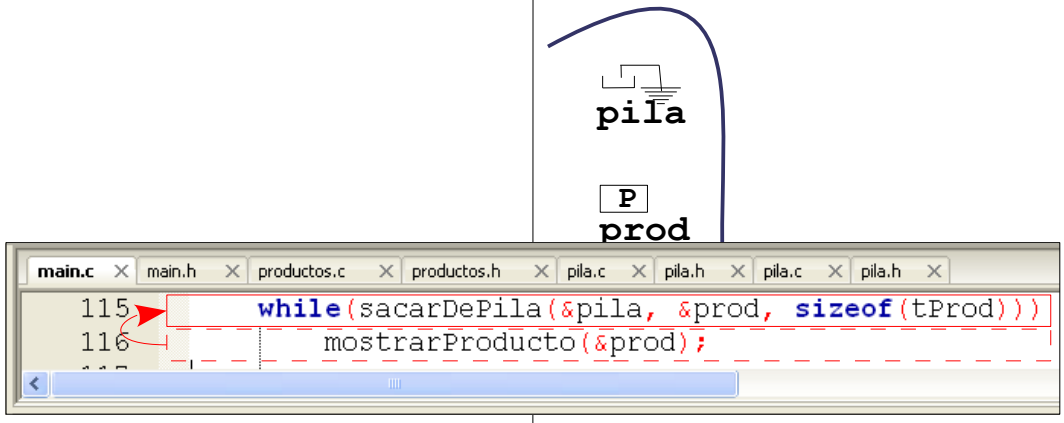
Implementación dinámica



Tipo de dato Pila.

Implementación estática

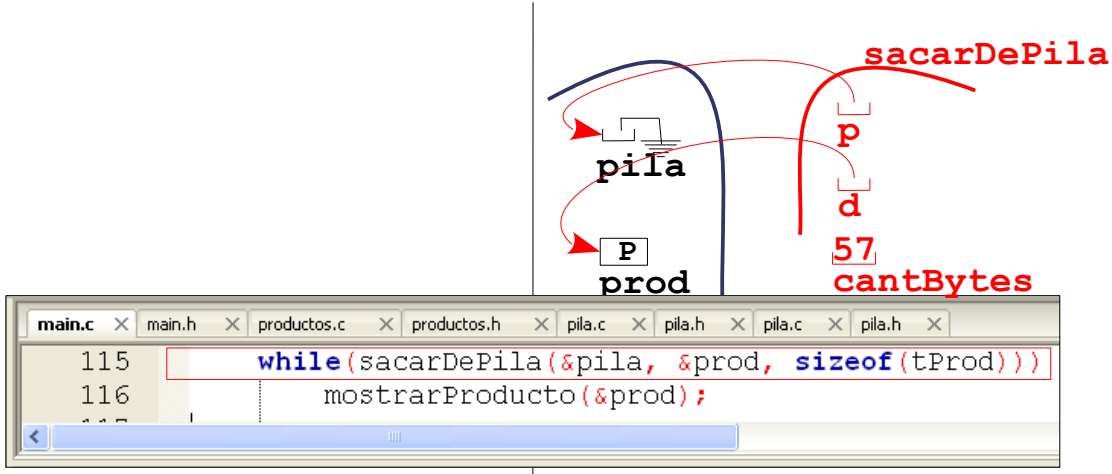
Implementación dinámica



Tipo de dato Pila.

Implementación estática

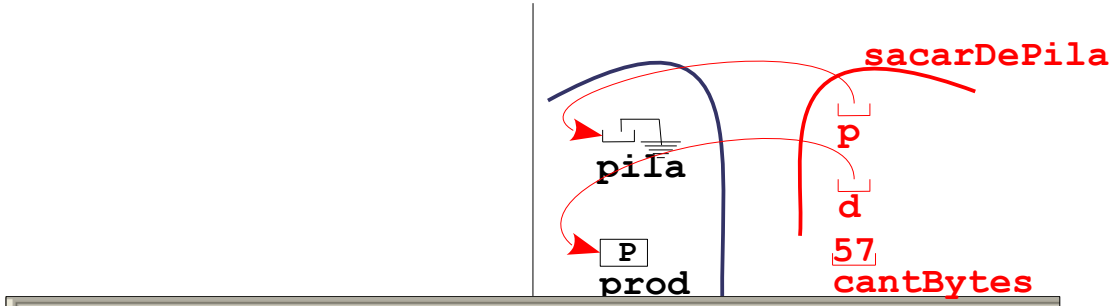
Implementación dinámica



Tipo de dato Pila.

Implementación estática

Implementación dinámica



main.c | main.h | productos.c | productos.h | pila.c | pila.h | pila.c | pila.h

```
115
116
```

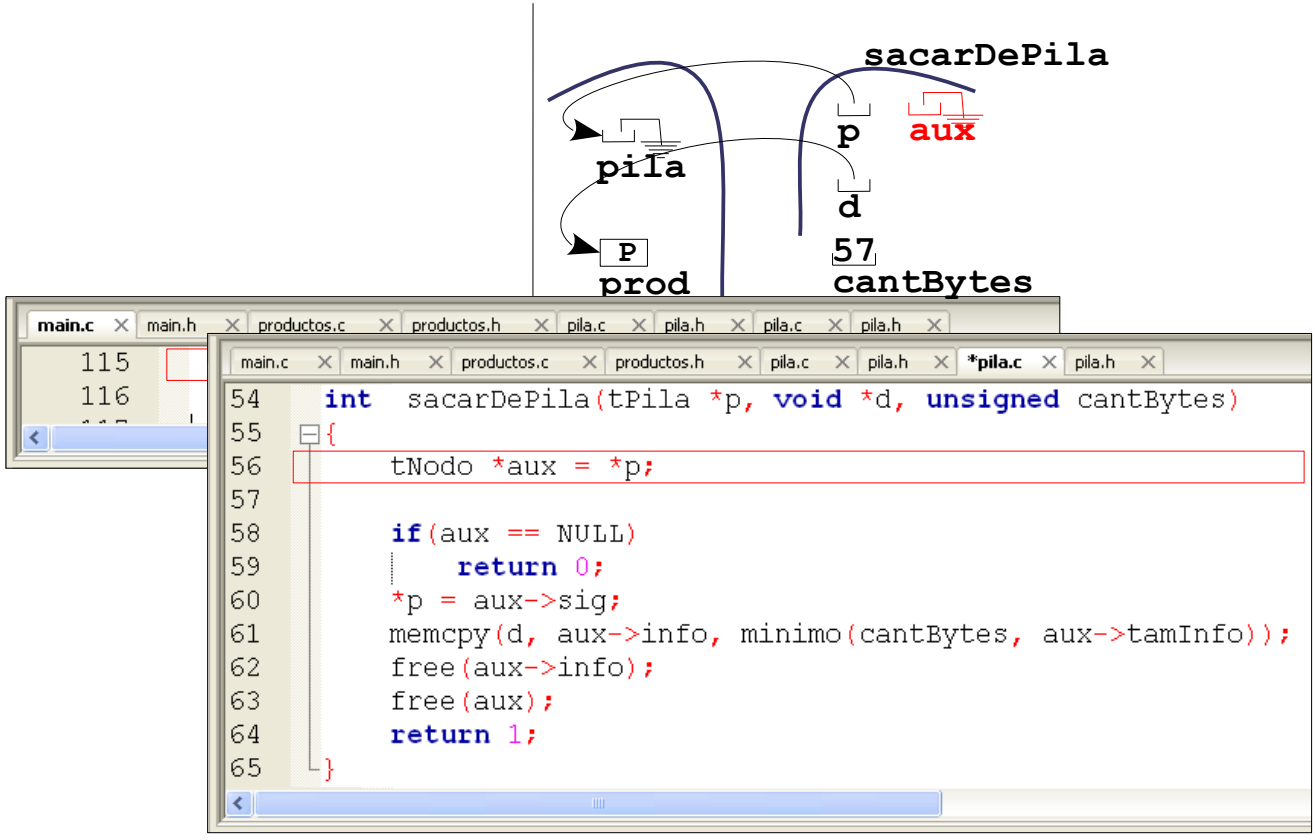
main.c | main.h | productos.c | productos.h | pila.c | pila.h | *pila.c | pila.h

```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

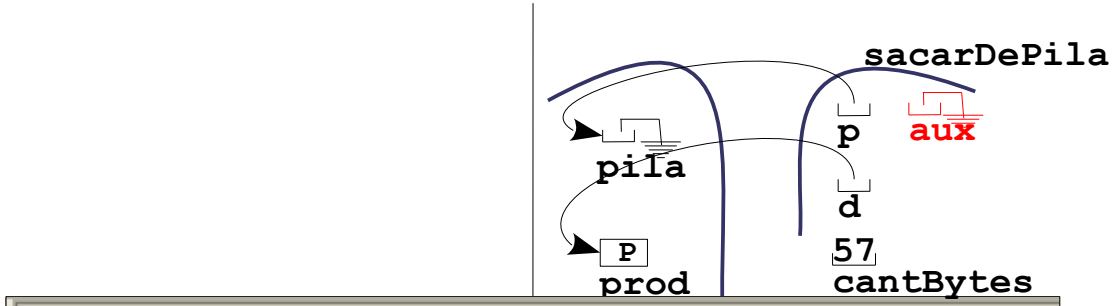
Implementación dinámica



Tipo de dato Pila.

Implementación estática

Implementación dinámica

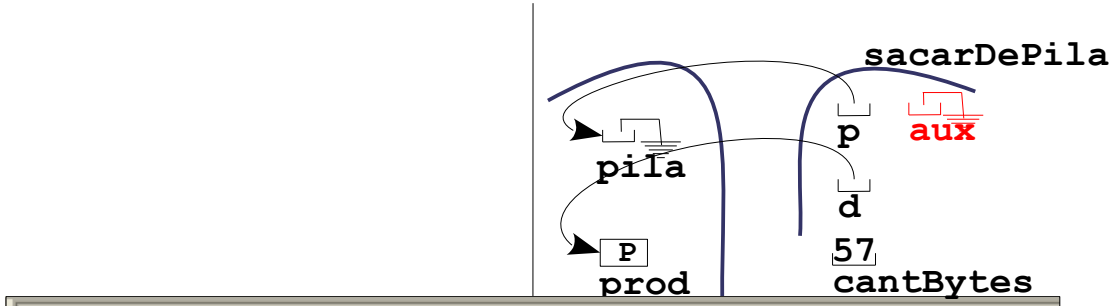



```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

Implementación estática

Implementación dinámica

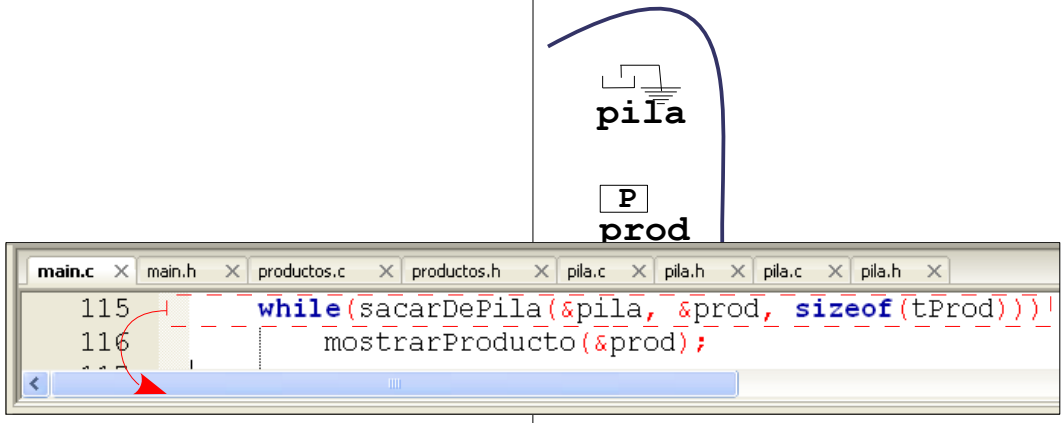



```
54  int  sacarDePila(tPila *p, void *d, unsigned cantBytes)
55  {
56      tNodo *aux = *p;
57
58      if(aux == NULL)
59          return 0;
60      *p = aux->sig;
61      memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62      free(aux->info);
63      free(aux);
64      return 1;
65  }
```

Tipo de dato Pila.

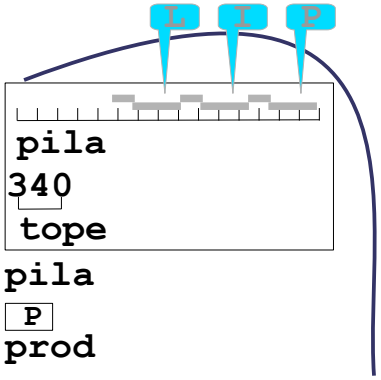
Implementación estática

Implementación dinámica



Tipo de dato Pila.

Implementación estática



Implementación dinámica