



# Tipos de Datos Abstractos

Tipo de dato Pila.

# Tipos de Datos Abstractos

Tipo de dato Pila.

Implementación estática

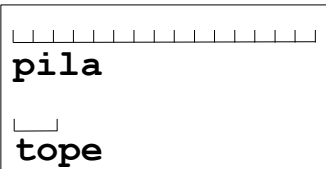
Implementación dinámica

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



tPila

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



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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

|||||  
pila  
?  
tope  
pila

?  
pila

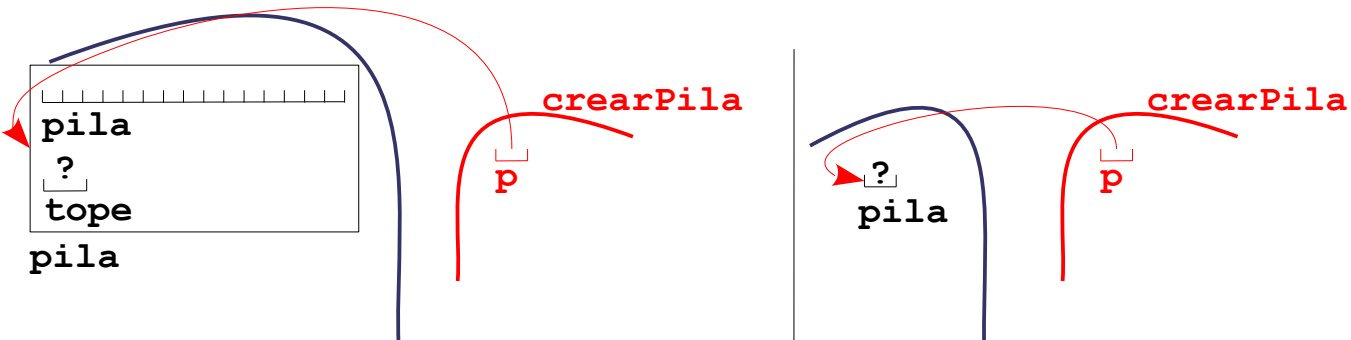
```
86 void probarPonerYSacarDePila(void)
87 {
88
89     tPila pila;
90
91
92
93
94
95
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



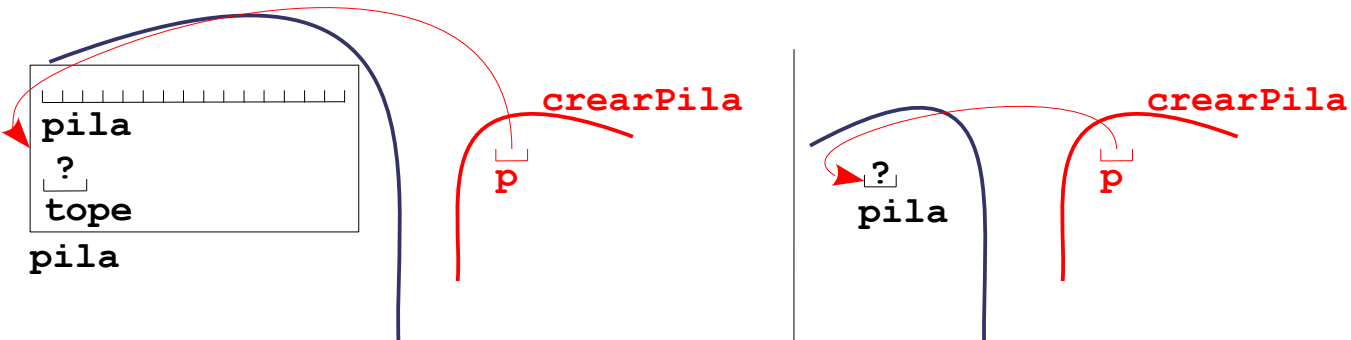
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88
89     tPila pila;
90
91     crearPila(&pila);
92
93
94
95
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88
89     tPila pila;
90
91     crearPila(&pila);
92
93 }
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x
9 void crearPila(tPila *p)
10 {
11     p->tope = TAM_PILA;
12 }
13
```

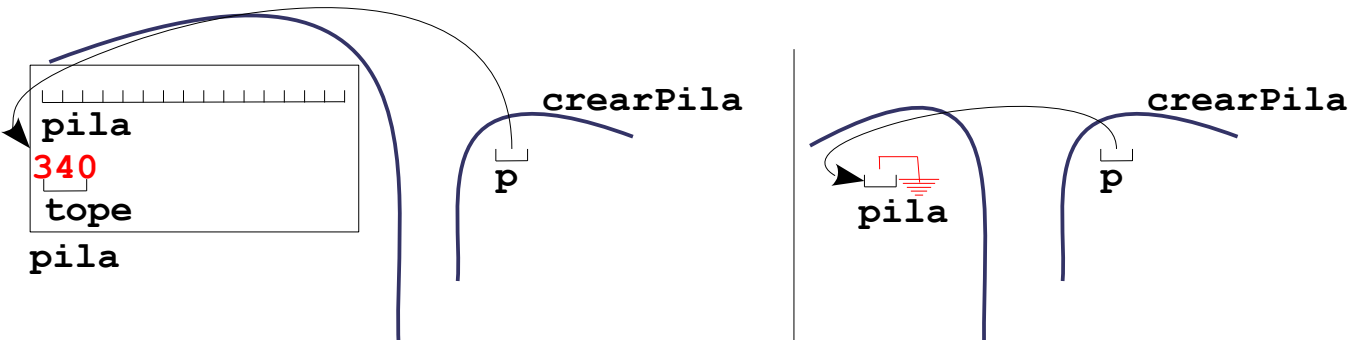
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x
9 void crearPila(tPila *p)
10 {
11     *p = NULL;
12 }
13
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88
89     tPila pila;
90
91     crearPila(&pila);
92
93 }
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x
9 void crearPila(tPila *p)
10 {
11     p->tope = TAM_PILA;
12 }
13
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x
9 void crearPila(tPila *p)
10 {
11     *p = NULL;
12 }
13
```

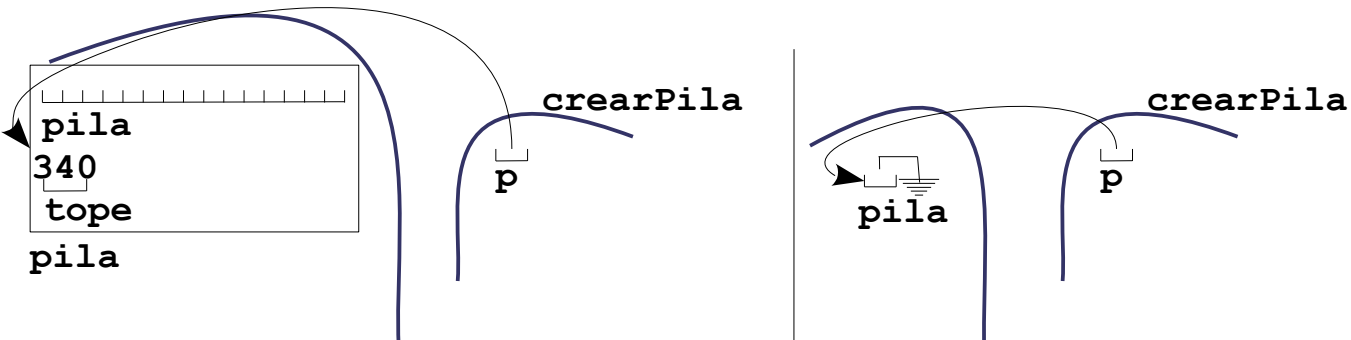


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



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

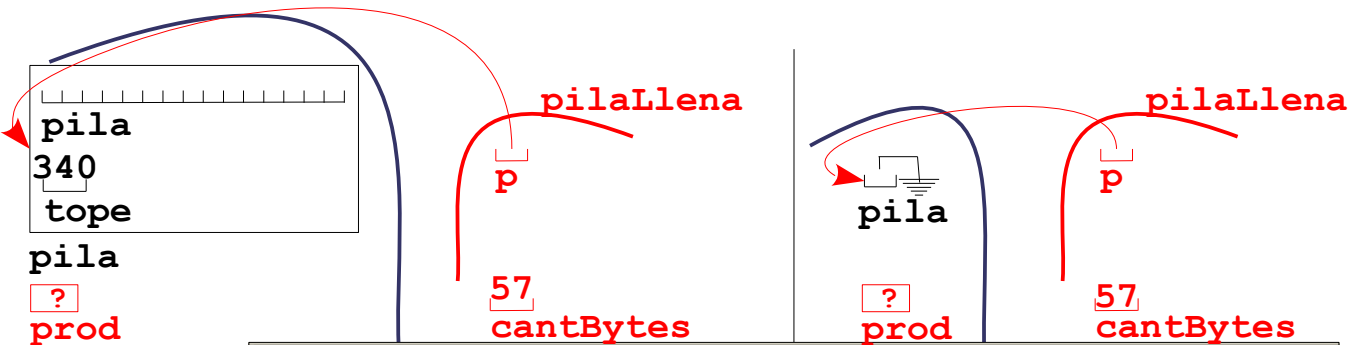


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



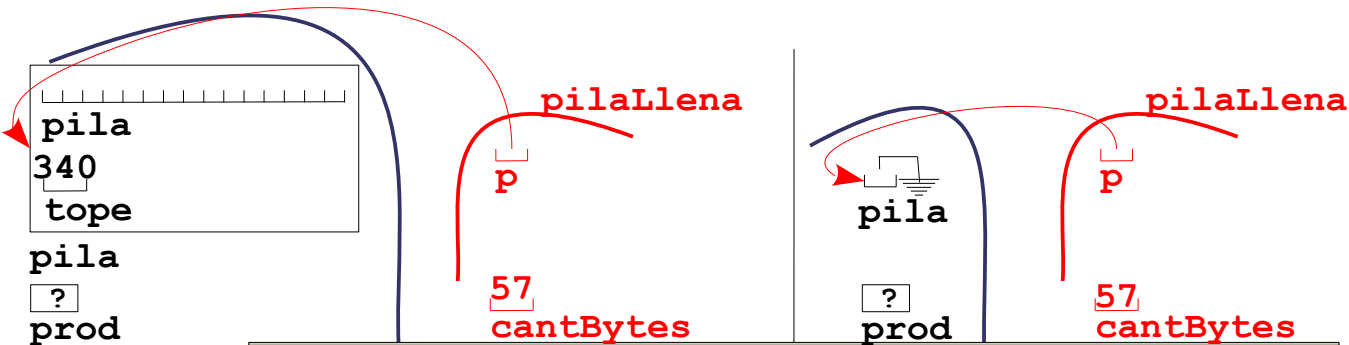
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd prod;
89     tPila pila;
90
91     crearPila(&pila);
92     while (!pilaLlena(&pila, sizeof(prod)) &&
93           !finDeProductos(&prod))
94         ingresarProducto(&prod);
95 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

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

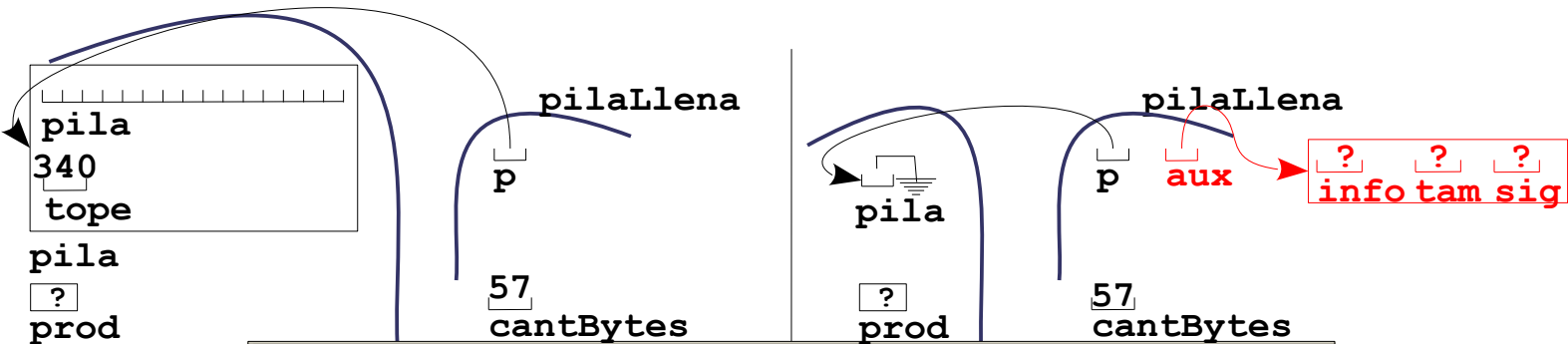
    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

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

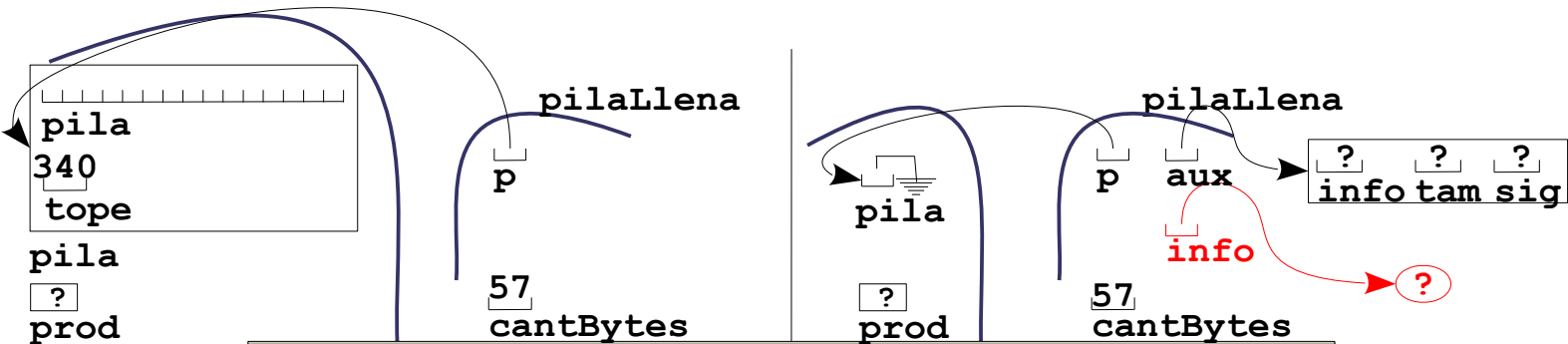
    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

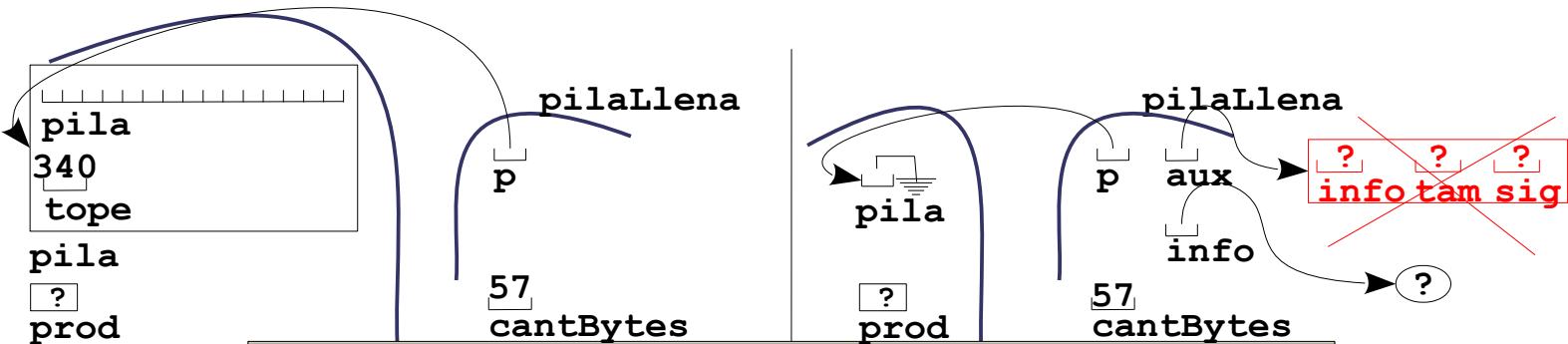
```
main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
19 int pilaLlena(const tPila *p, unsigned cantBytes)
20 {
21     tNodo *aux = (tNodo *)malloc(sizeof(tNodo));
22     void *info = malloc(cantBytes);
23
24     free(aux);
25     free(info);
26     return aux == NULL || info == NULL;
27 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

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

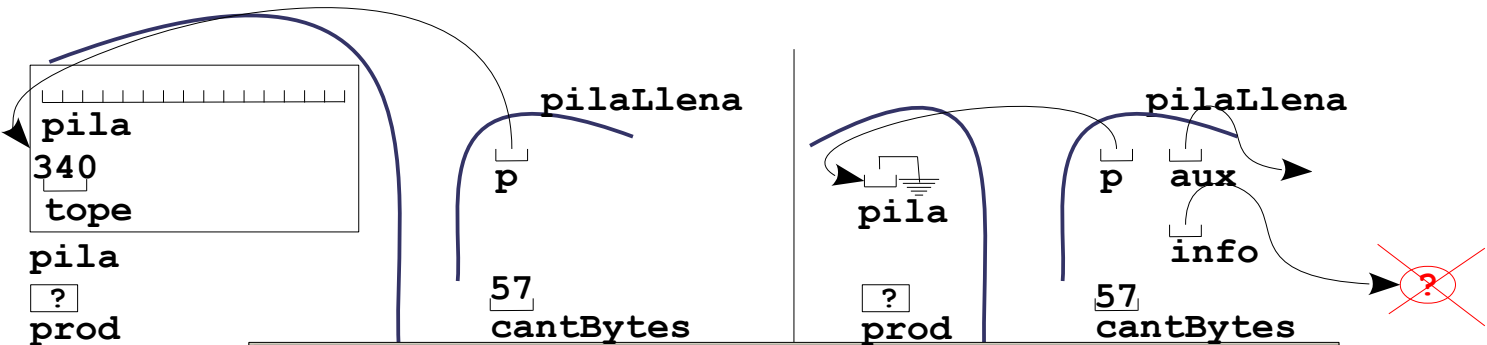
    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

```
main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
19 int pilaLlena(const tPila *p, unsigned cantBytes)
20 {
21     tNodo *aux = (tNodo *)malloc(sizeof(tNodo));
22     void *info = malloc(cantBytes);
23     free(aux);
24     free(info);
25     return aux == NULL || info == NULL;
26 }
```

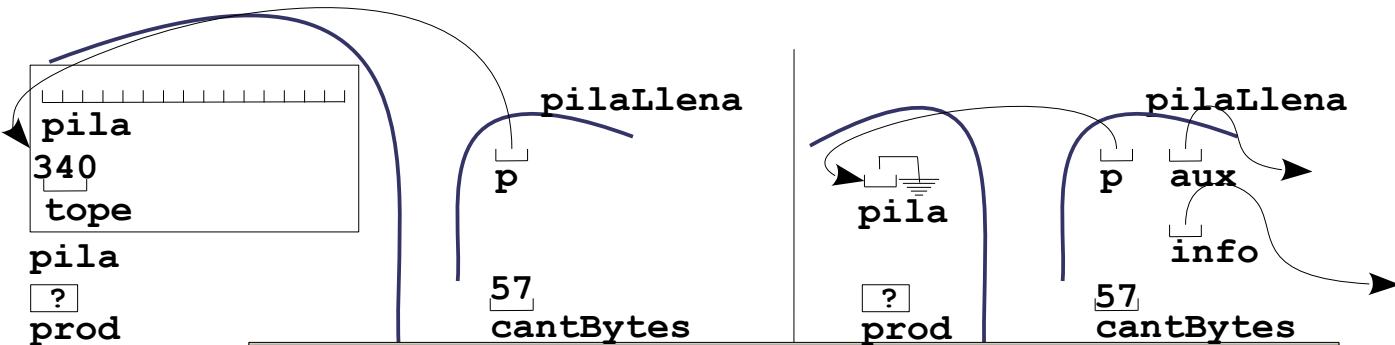


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

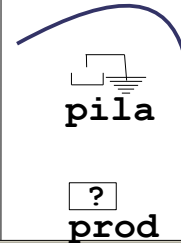
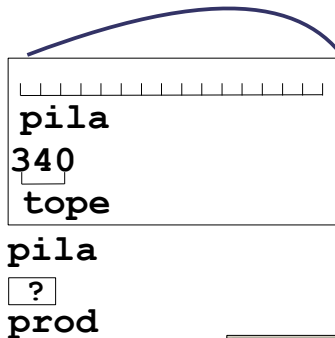
```
main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
19 int pilaLlena(const tPila *p, unsigned cantBytes)
20 {
21     tNodo *aux = (tNodo *)malloc(sizeof(tNodo));
22     void *info = malloc(cantBytes);
23
24     free(aux);
25     free(info);
26     return aux == NULL || info == NULL;
27 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



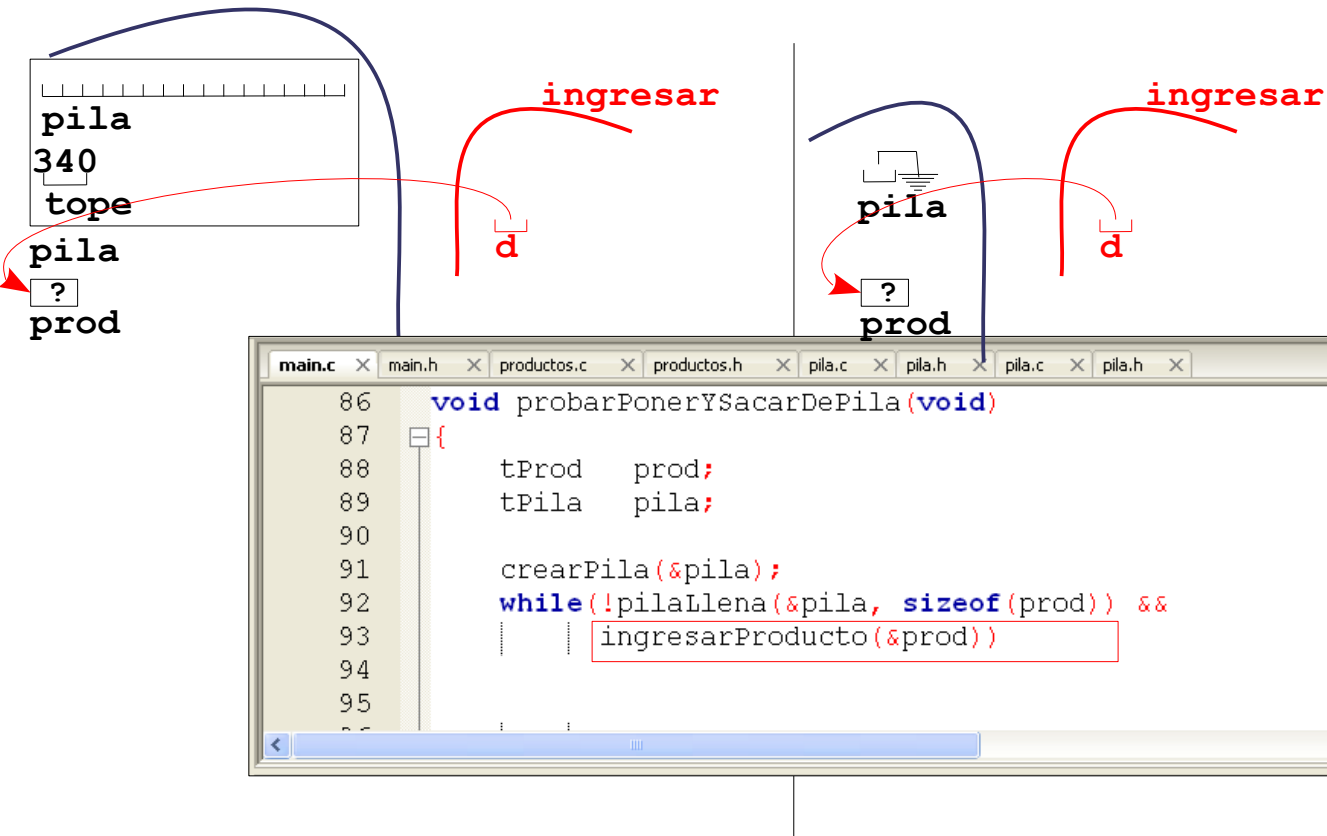
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd prod;
89     tPila pila;
90
91     crearPila(&pila);
92     while (!pilaLlena(&pila, sizeof(prod))) &&
93         ingresarProducto(&prod);
94
95     ...
96 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

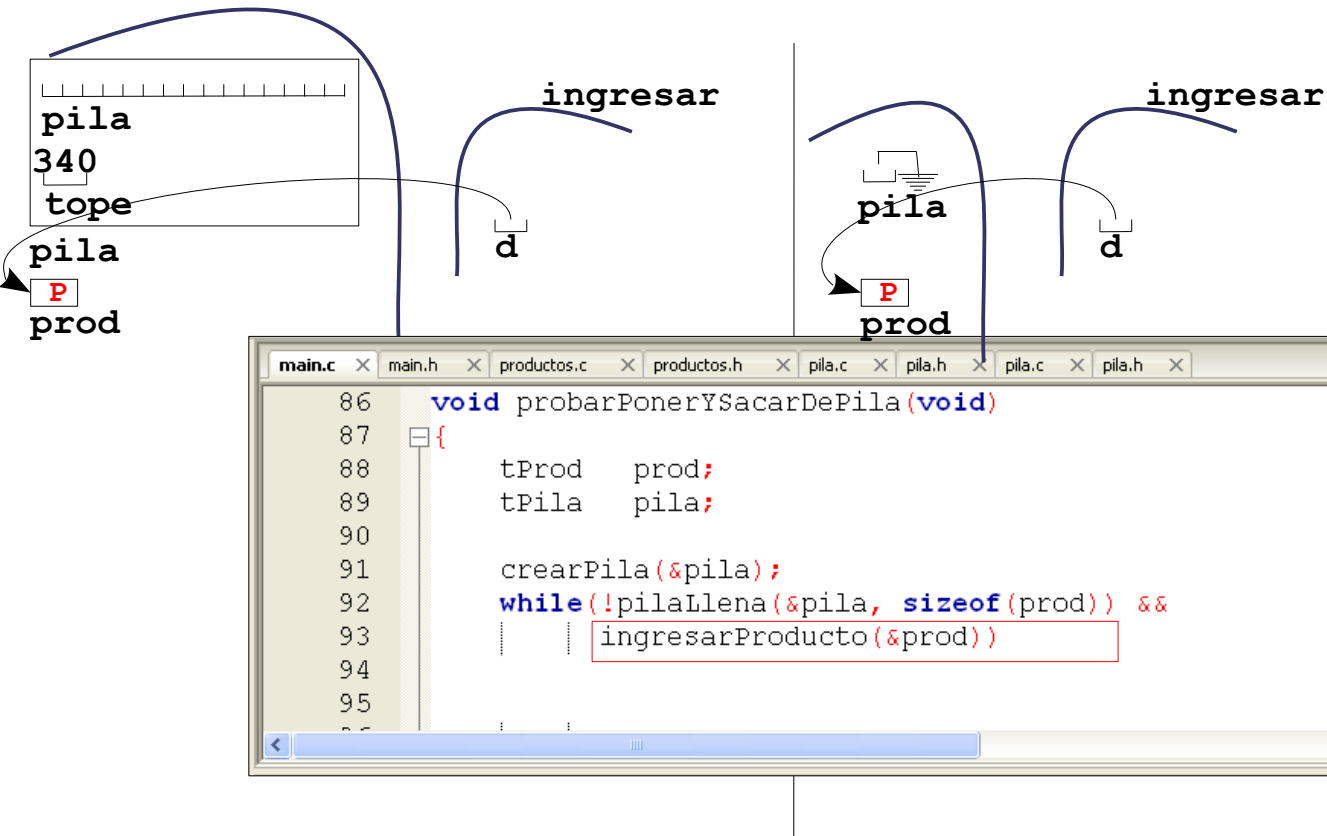


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

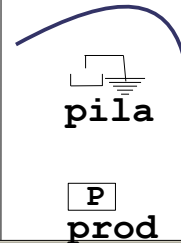
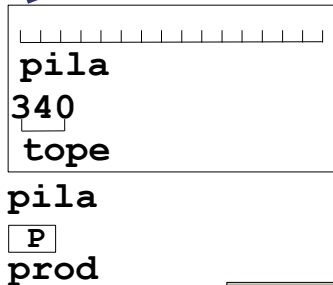


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



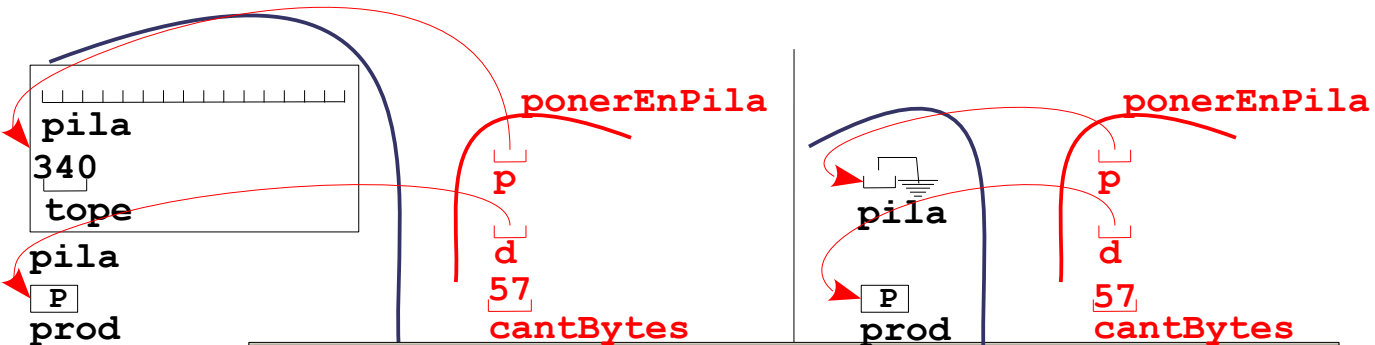
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd  prod;
89     tPila  pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93         ingresarProducto(&prod))
94         if(!ponerEnPila(&pila,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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



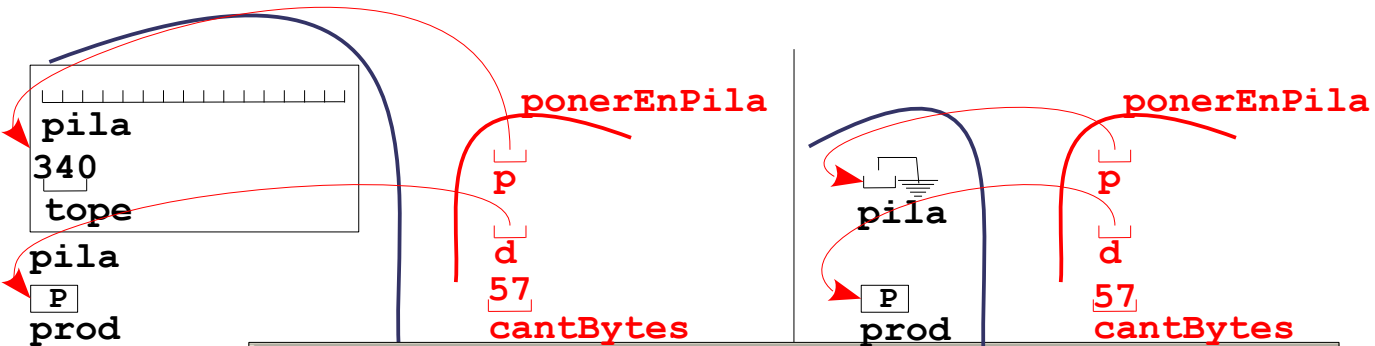
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd prod;
89     tPila pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93         ingresarProducto(&prod))
94         if(!ponerEnPila(&pila,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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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;
}
```

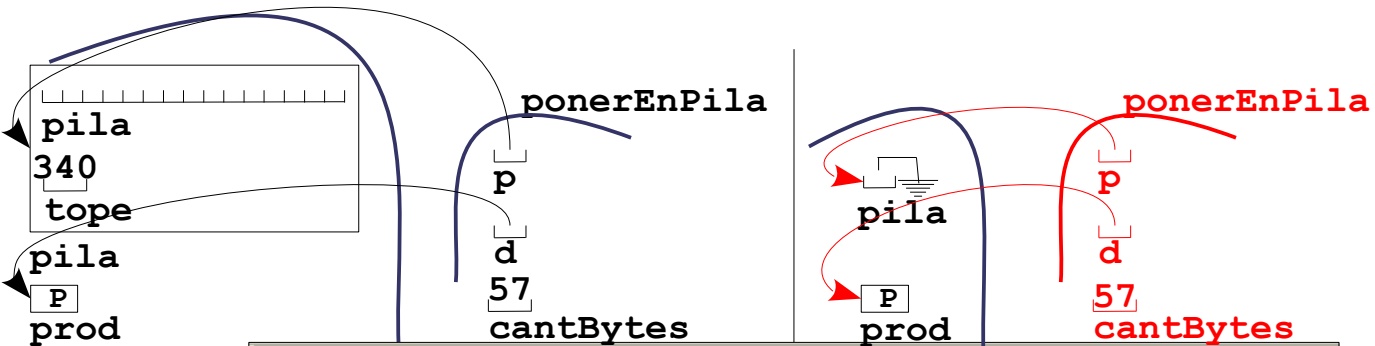
```
) &&
sizeof(prod))
lla llena\n");
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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;
}
```

```
) &&
sizeof(prod))
lla llena\n");
```

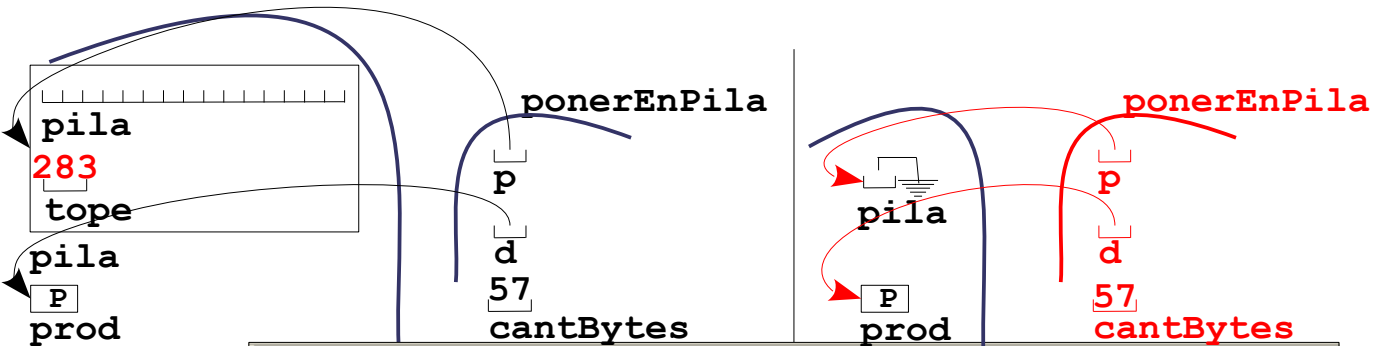


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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;
}
```

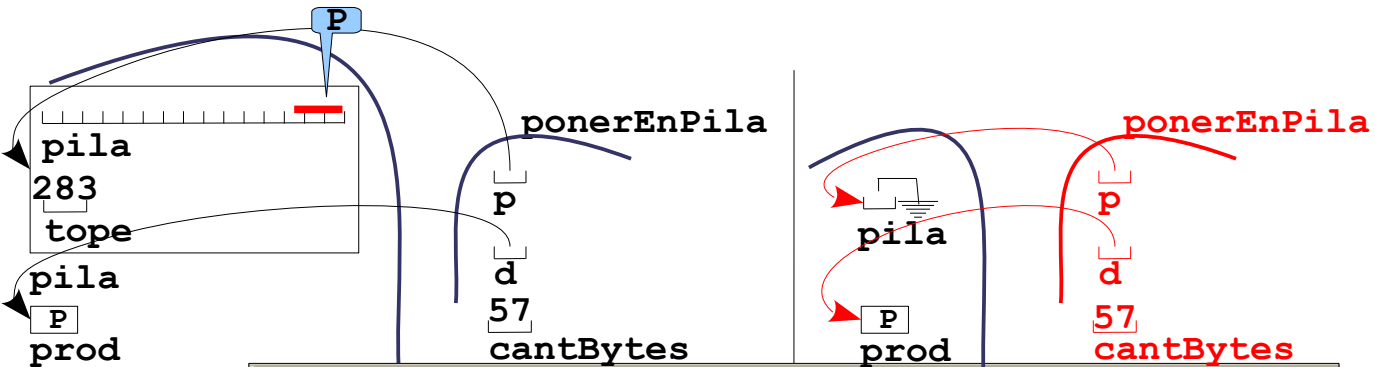
```
) &&
sizeof(prod))
la llena\n");
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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;
}
```

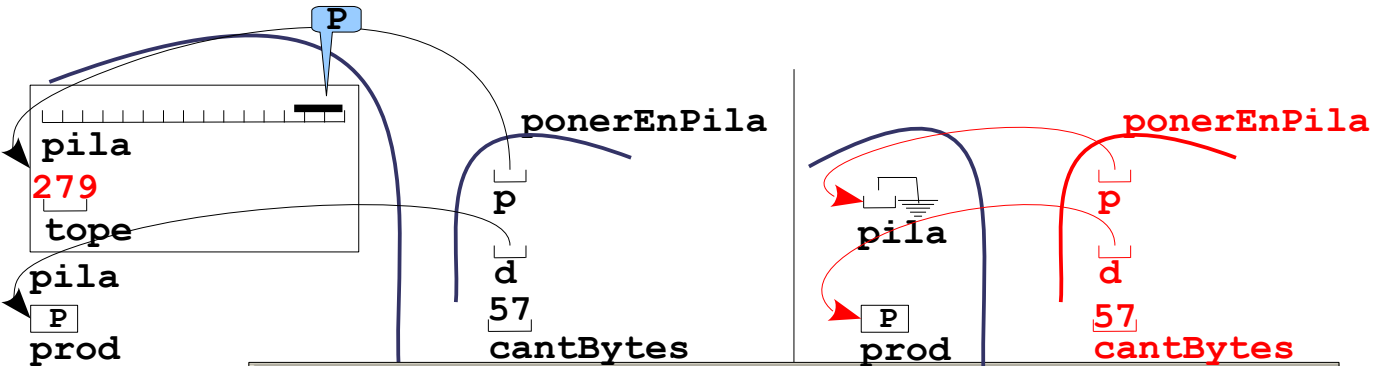
```
) &&
sizeof(prod))
la llena\n");
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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;
}
```

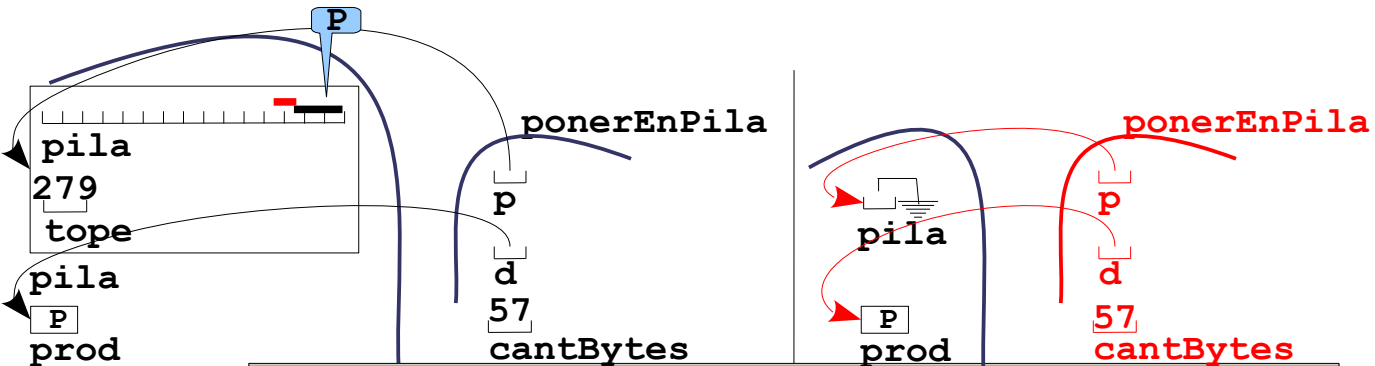
```
) &&
sizeof(prod))
la llena\n");
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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;
}
```

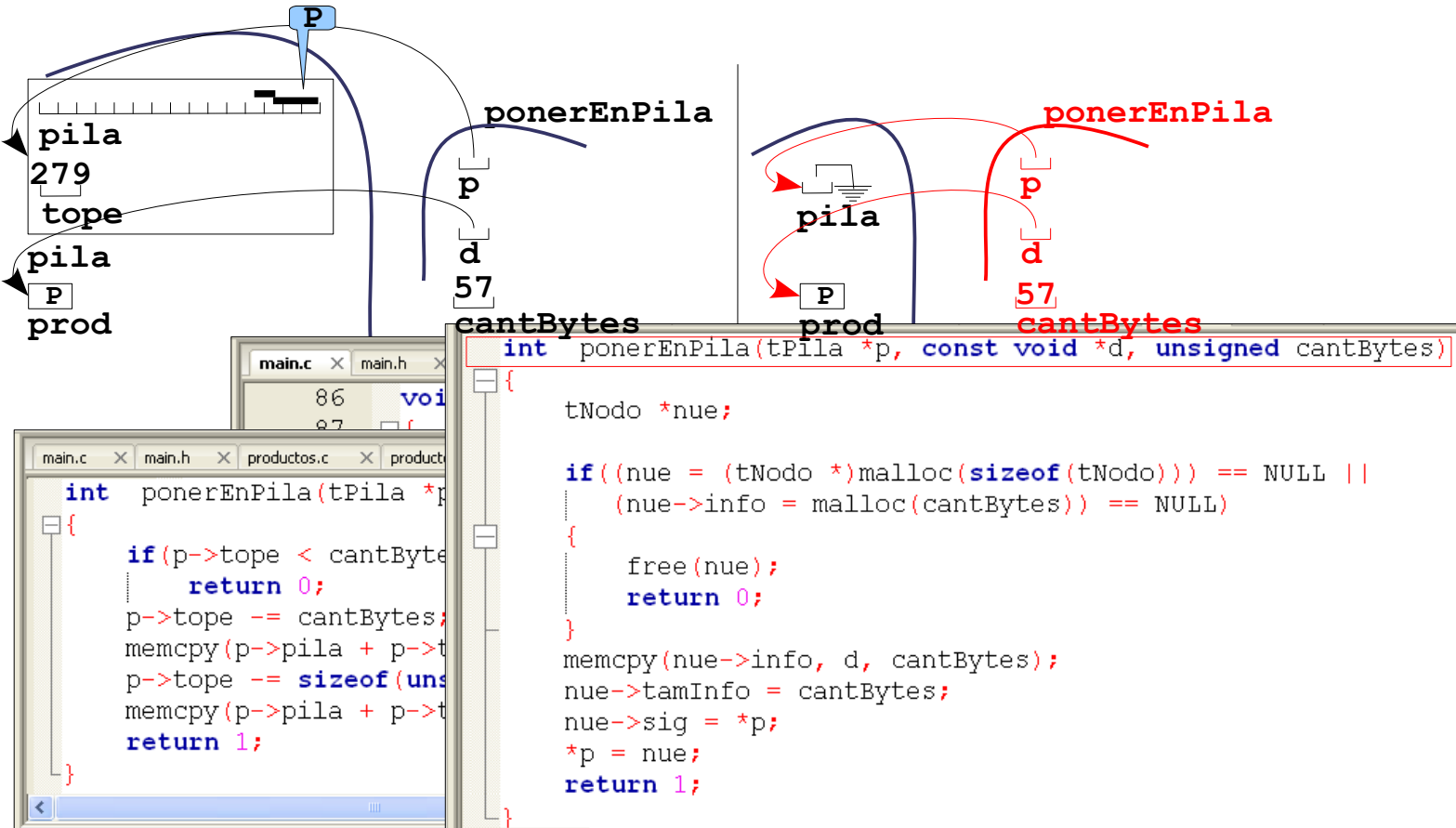
```
) &&
sizeof(prod))
la llena\n");
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica

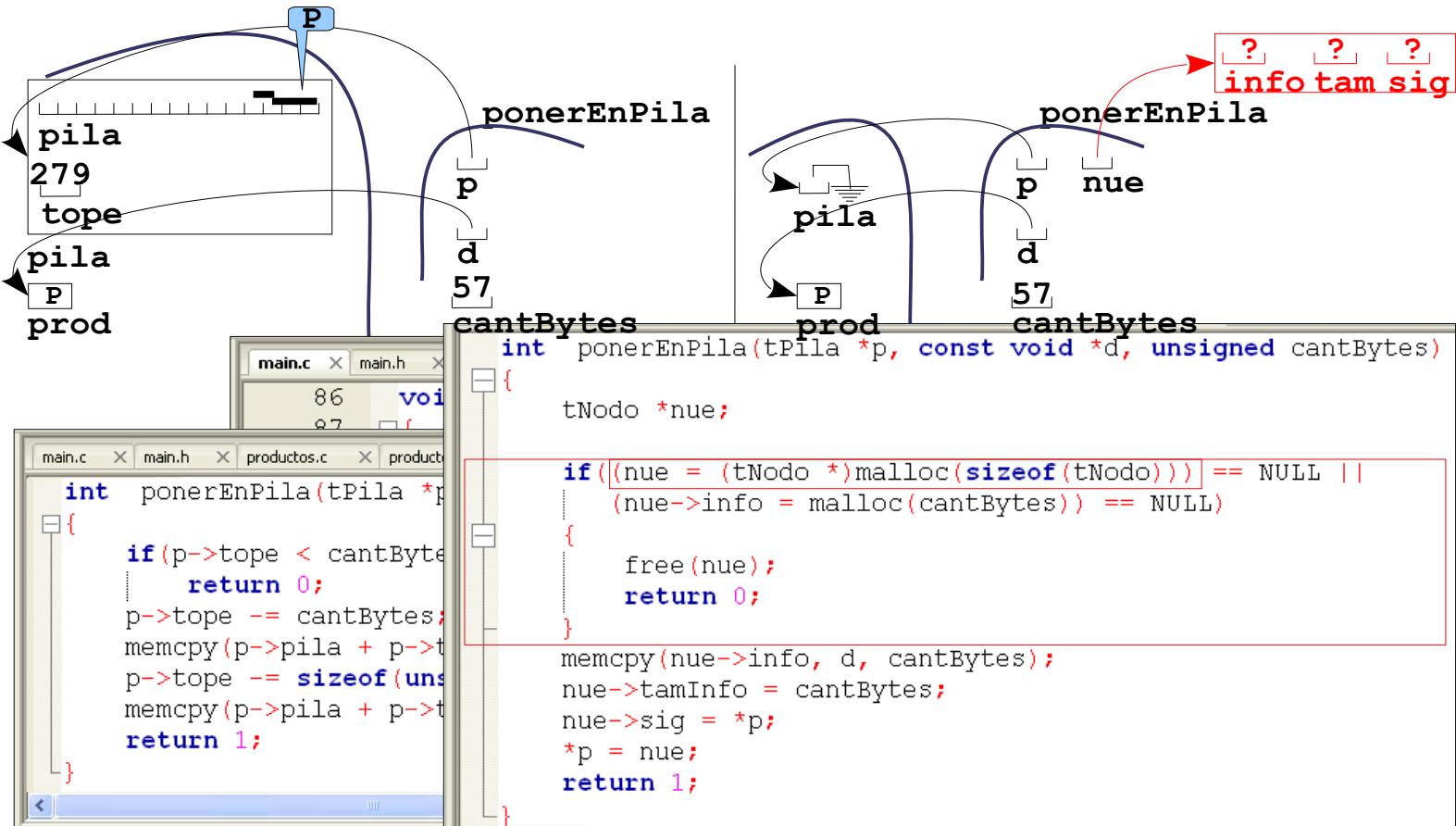


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica

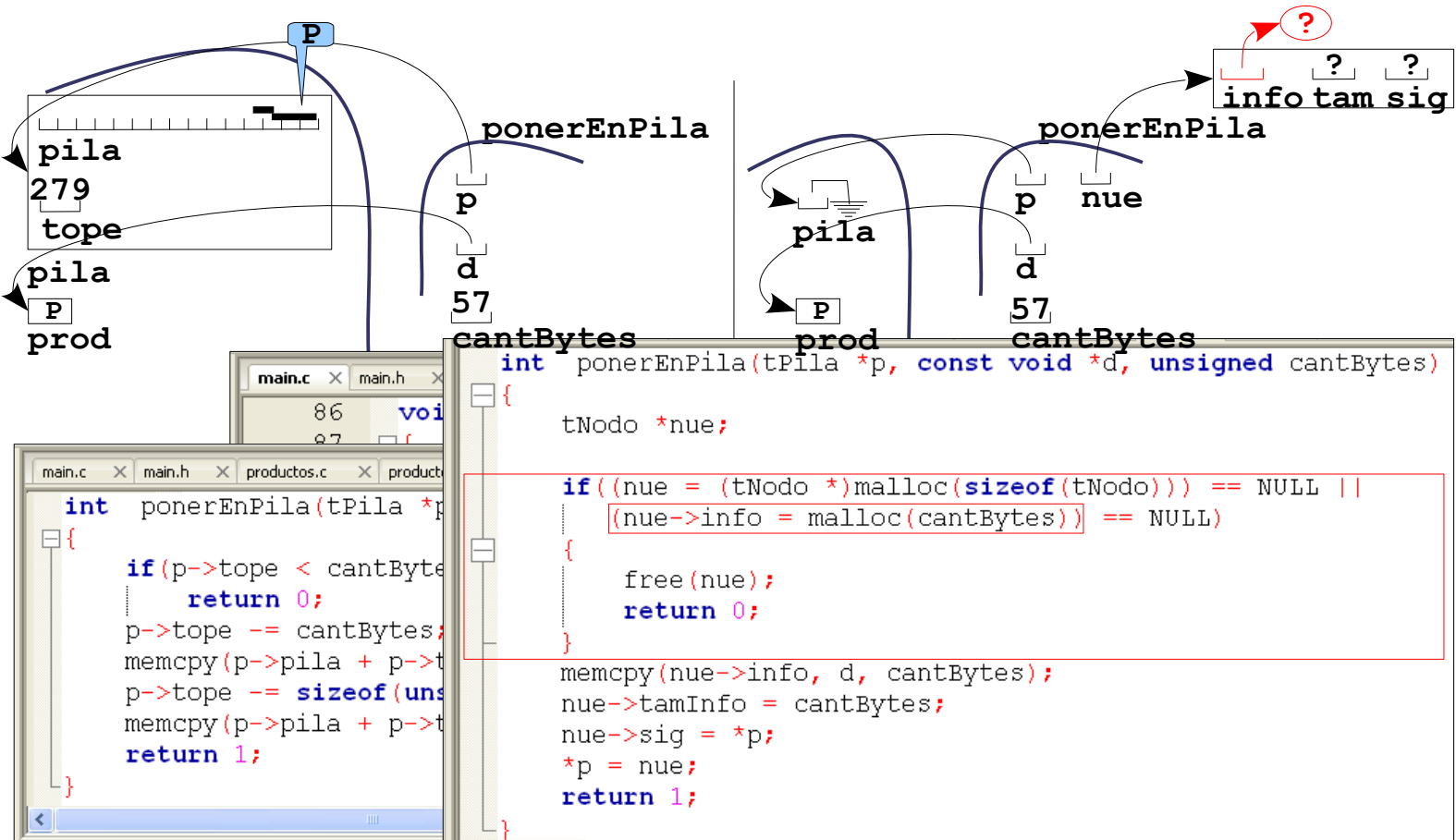


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica

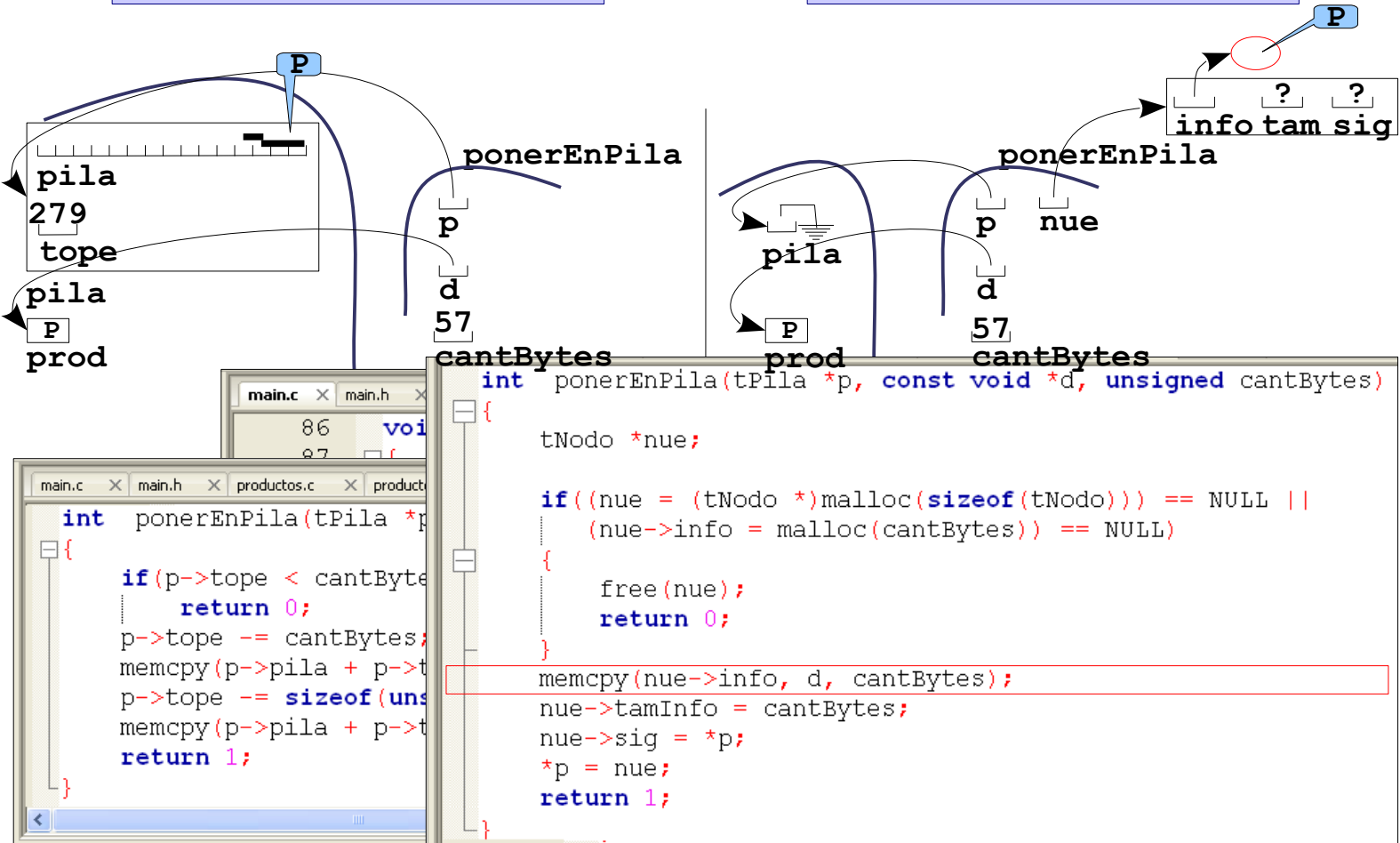


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



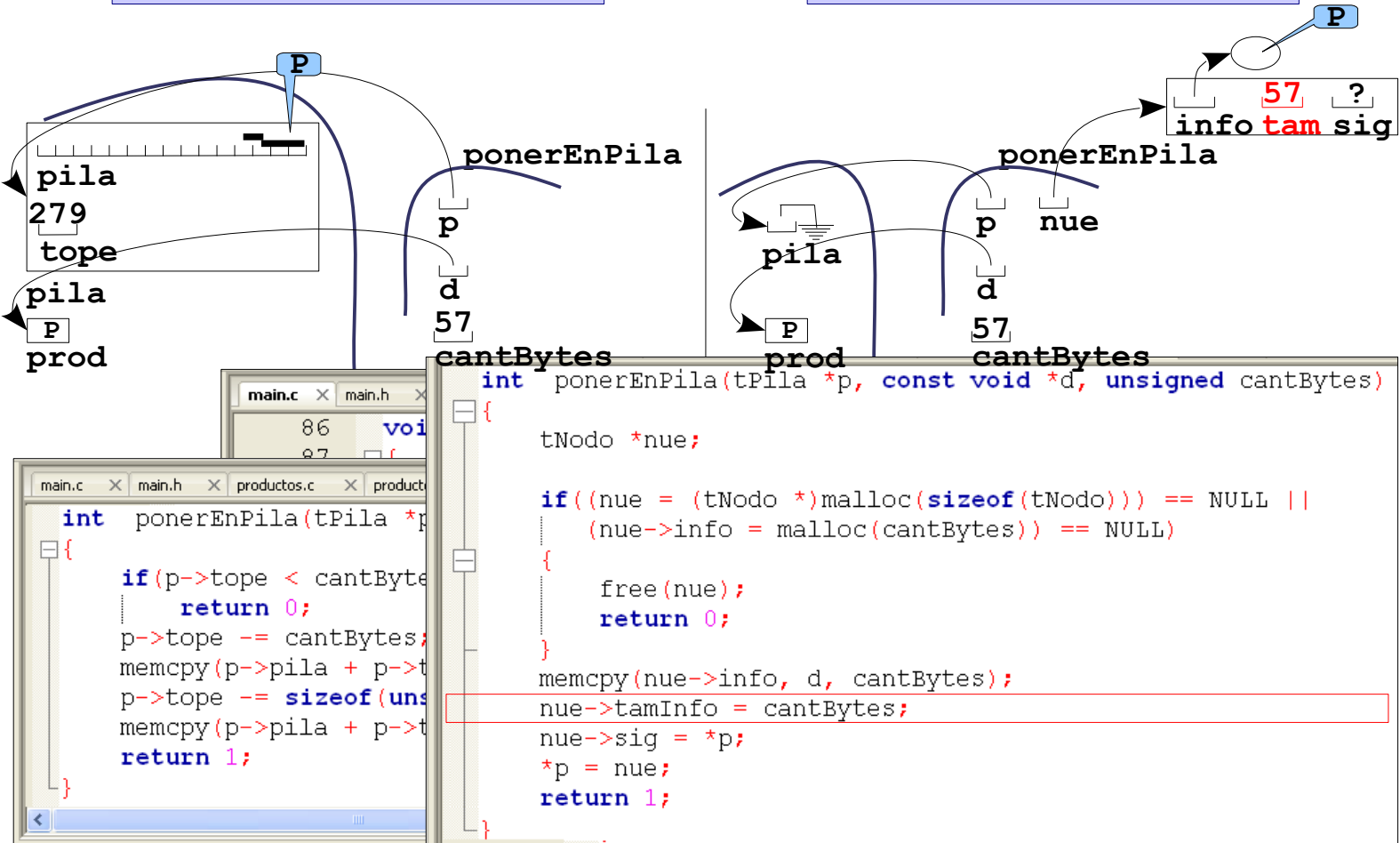


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica

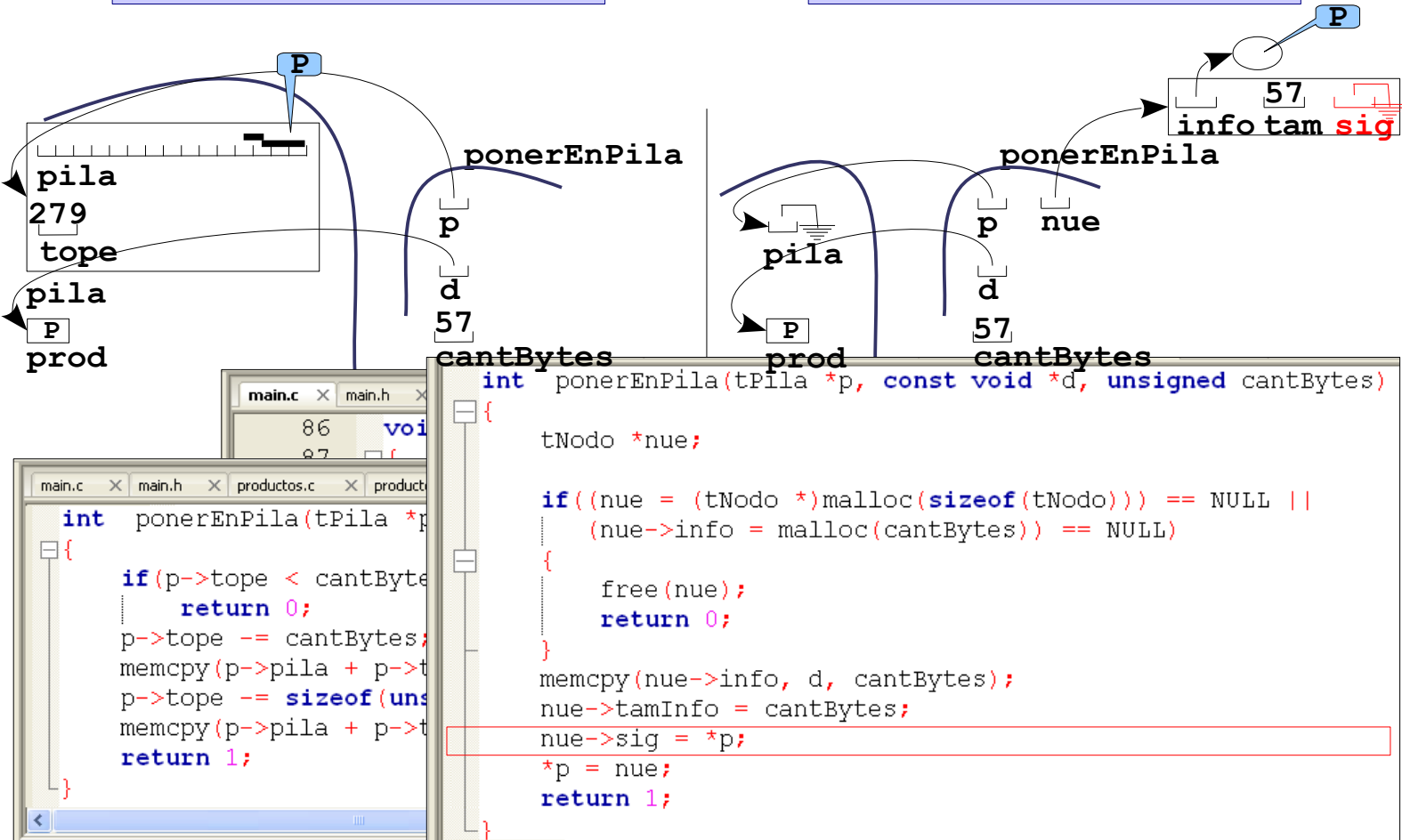


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica

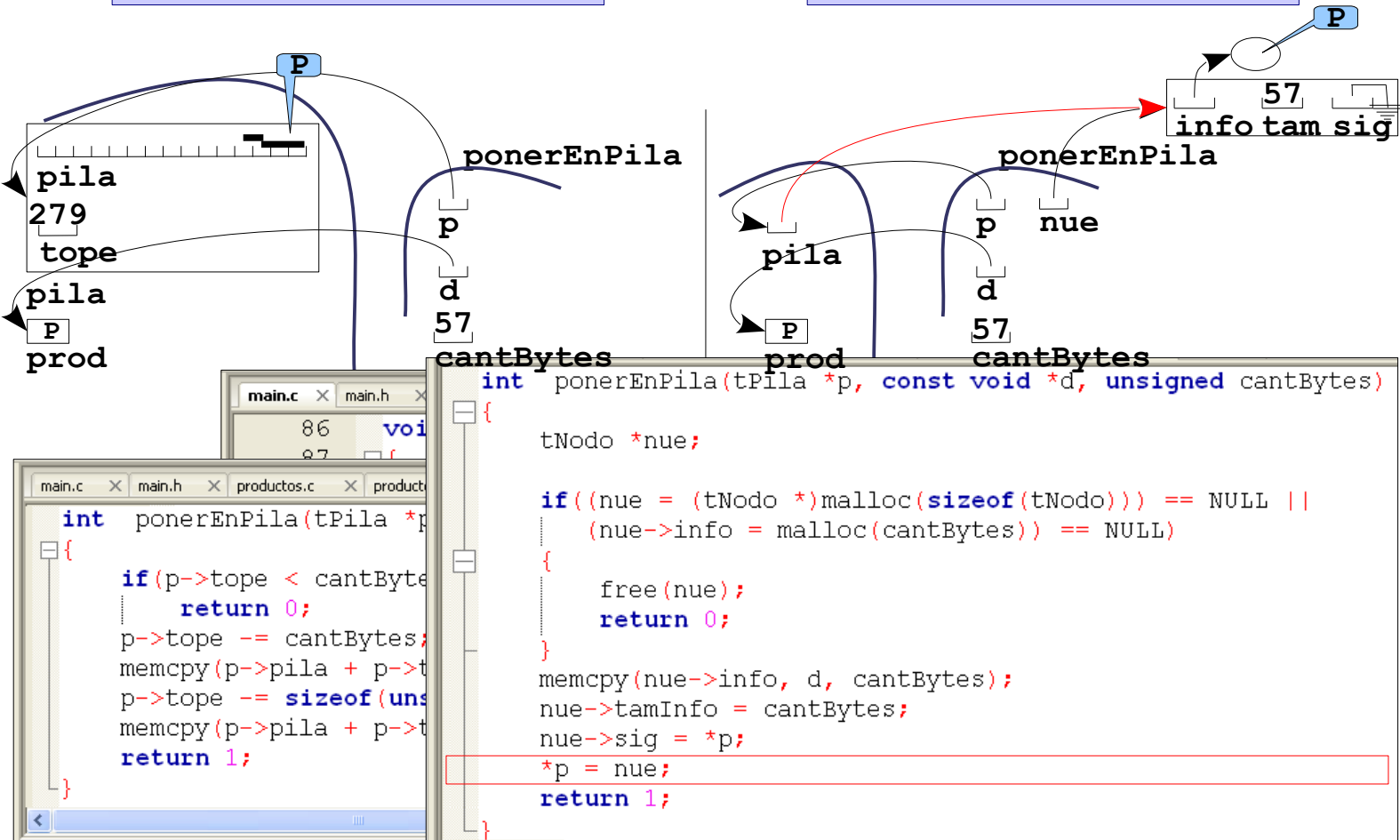


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica

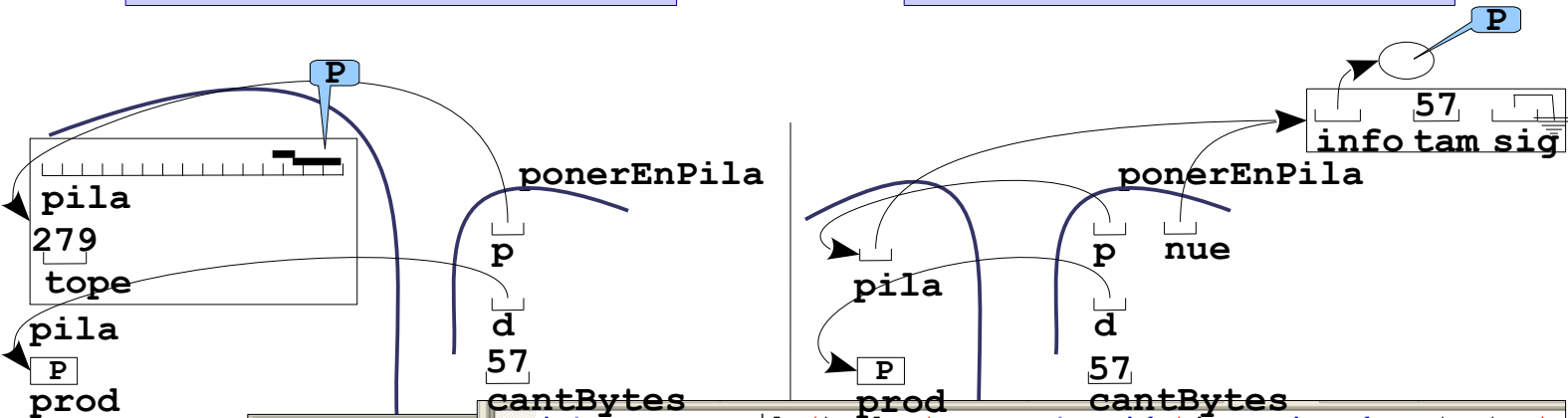


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



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

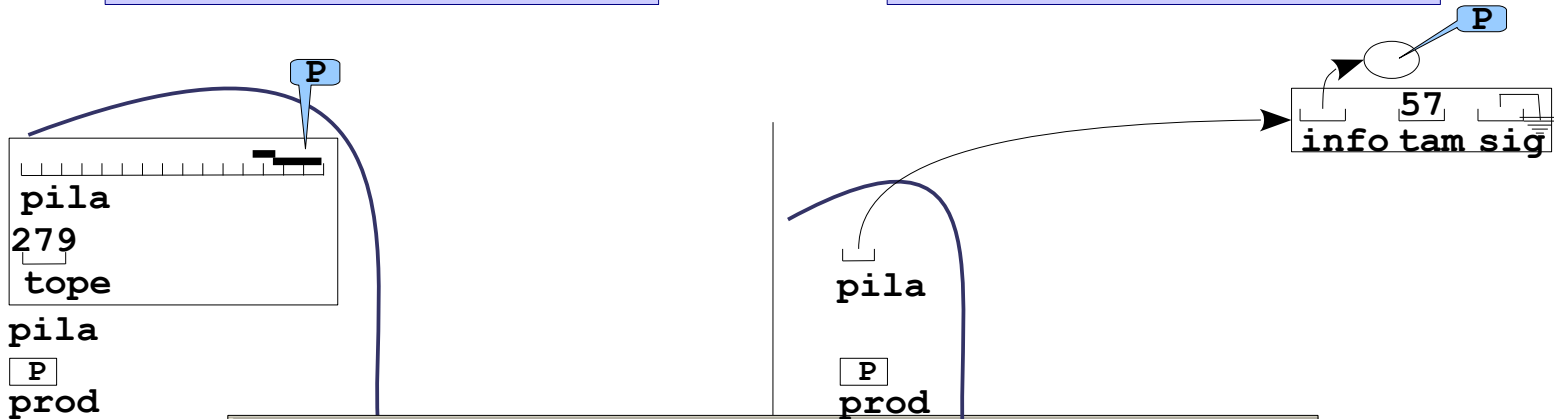
    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
        (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



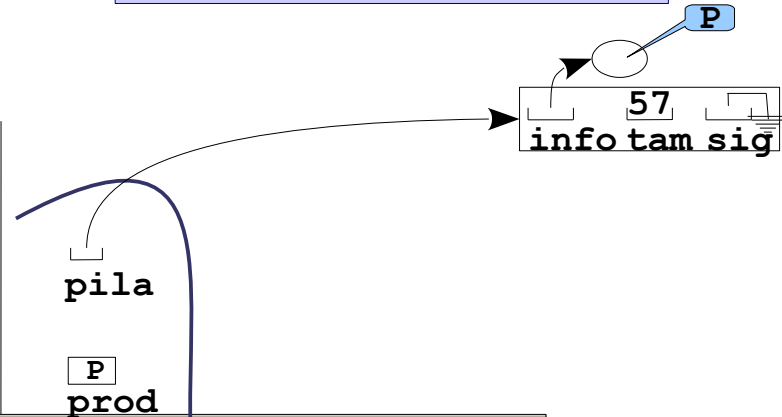
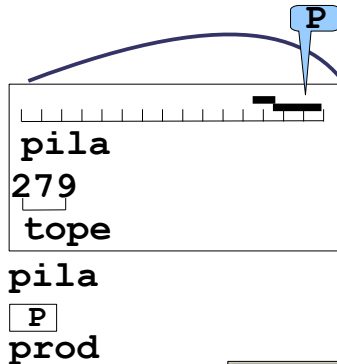
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd  prod;
89     tPila  pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93         ingresarProducto(&prod))
94         if(!ponerEnPila(&pila,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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



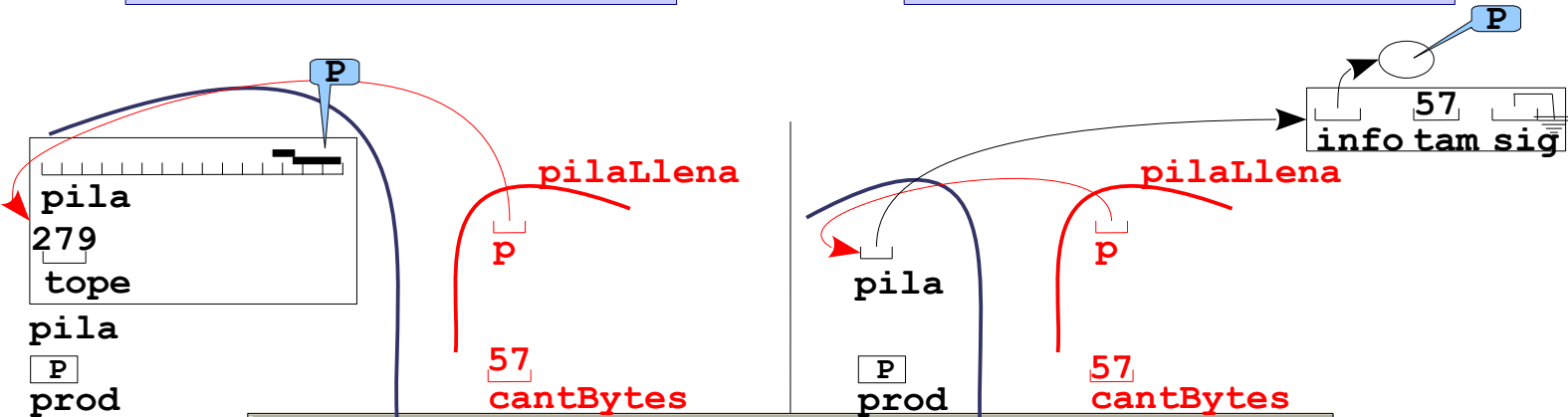
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd  prod;
89     tPila  pila;
90
91     crearPila(&pila);
92     while (!pilaLlena(&pila, sizeof(prod)) &&
93           ingresarProducto(&prod))
94         if (!ponerEnPila(&pila,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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



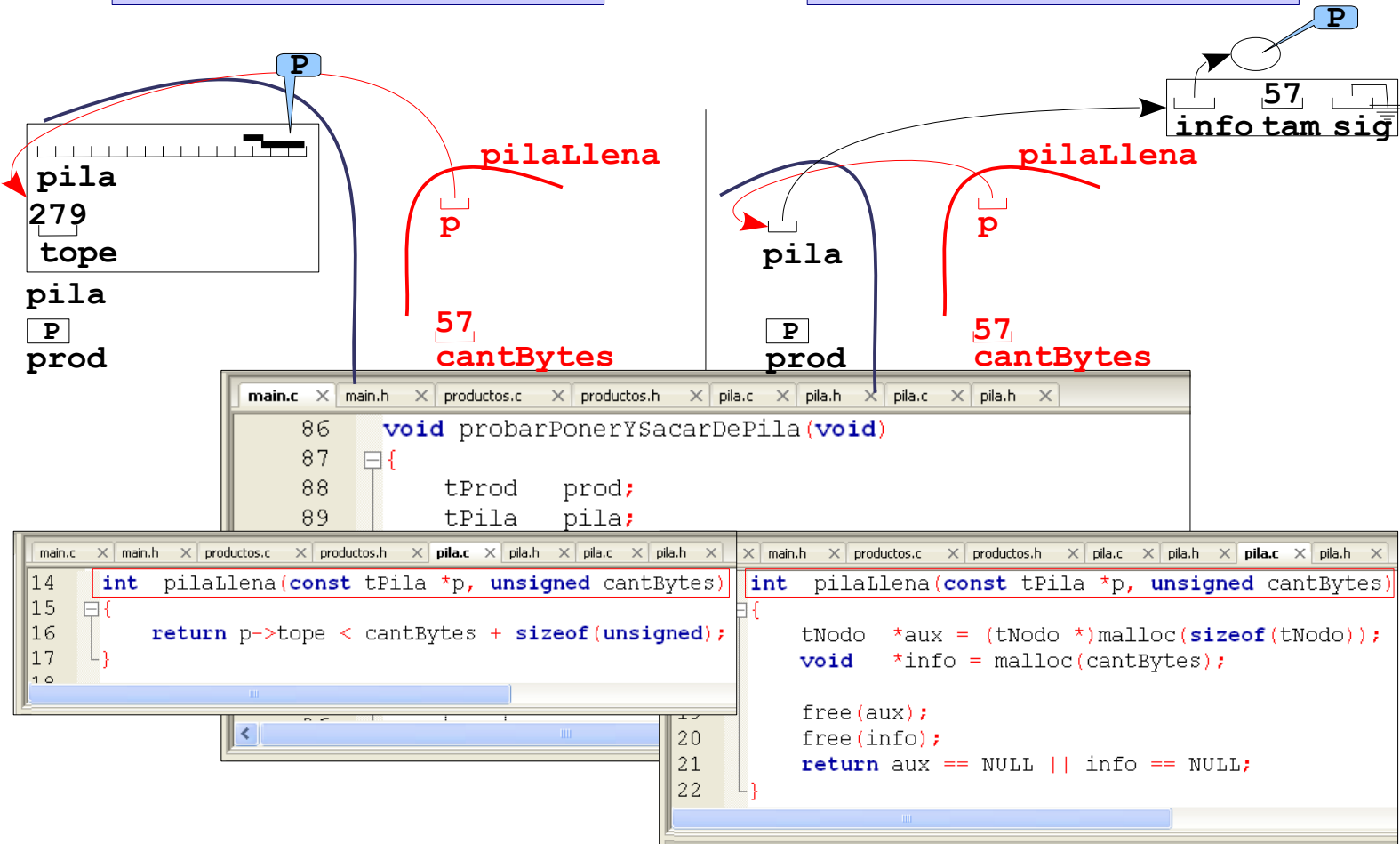
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd prod;
89     tPila pila;
90
91     crearPila(&pila);
92     while (!pilaLlena(&pila, sizeof(prod)) &&
93           ingresarProducto(&prod))
94         if (!ponerEnPila(&pila,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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



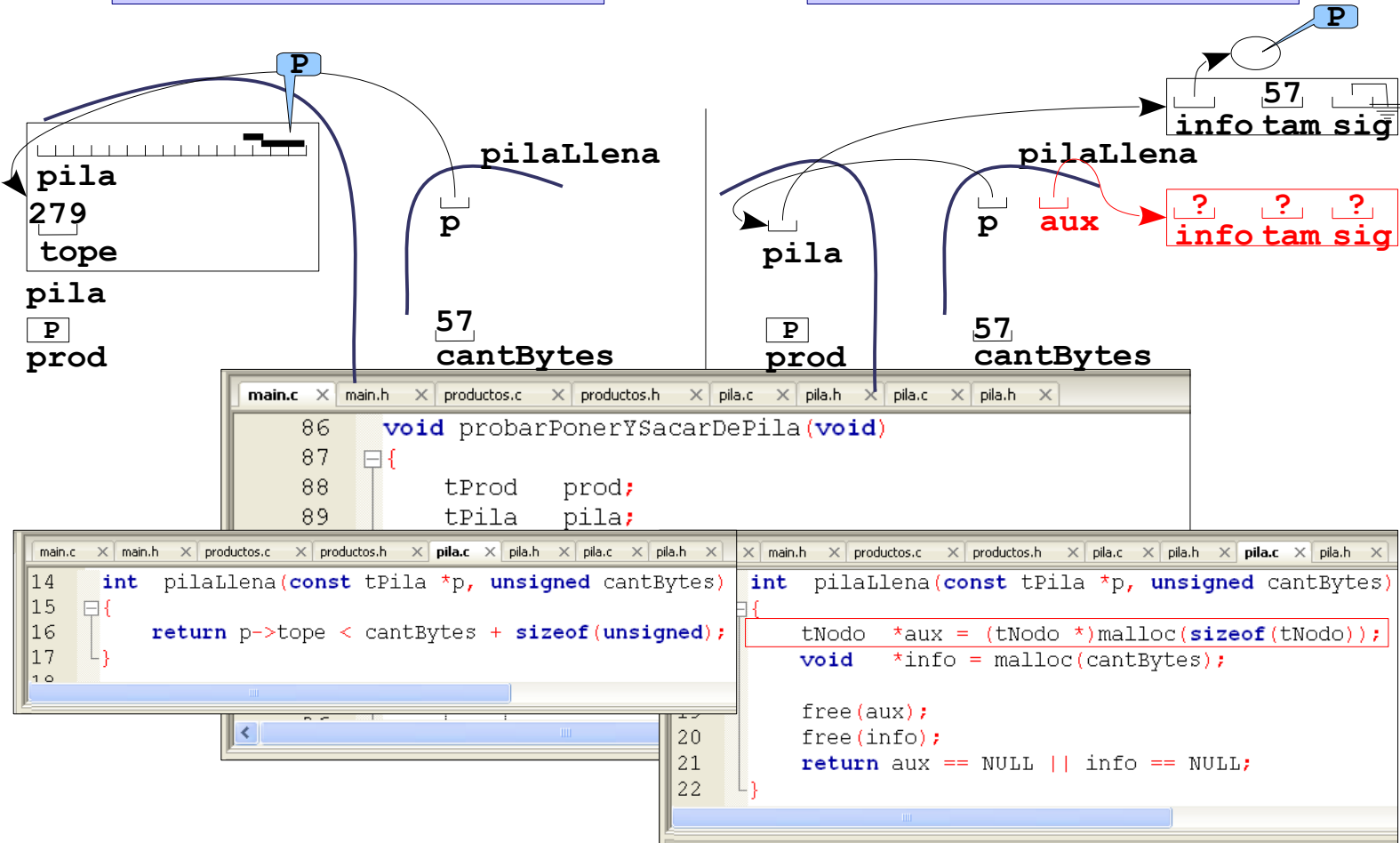


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica

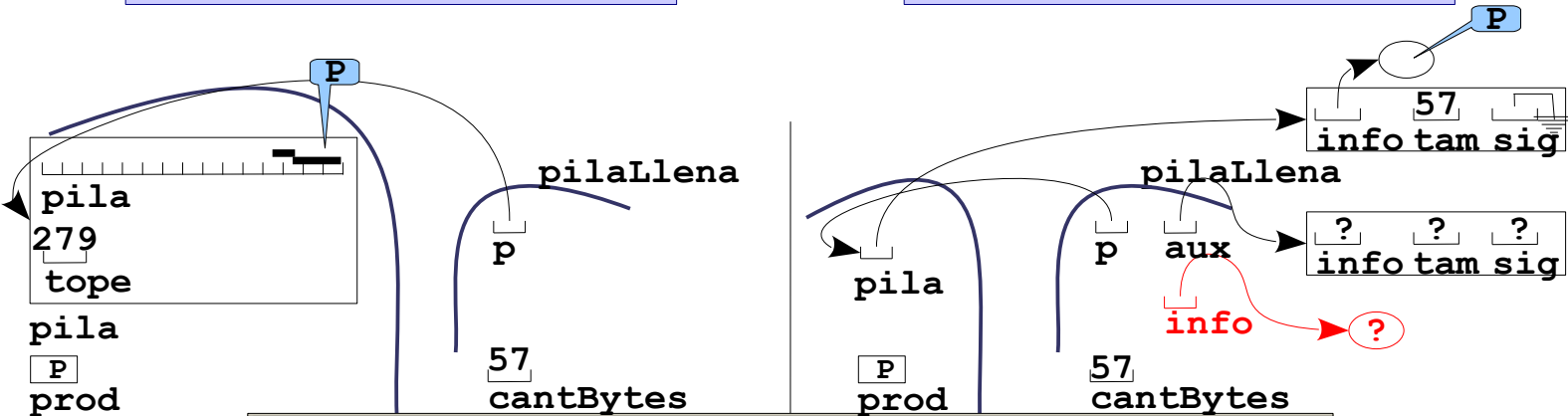


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

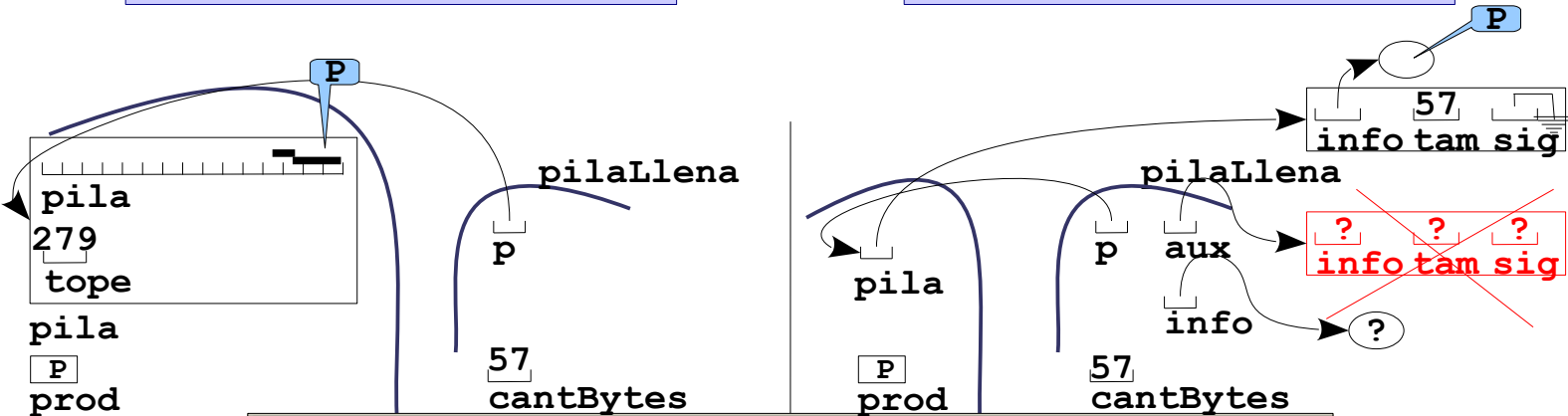
```
main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
19 int pilaLlena(const tPila *p, unsigned cantBytes)
20 {
21     tNodo *aux = (tNodo *)malloc(sizeof(tNodo));
22     void *info = malloc(cantBytes);
23
24     free(aux);
25     free(info);
26     return aux == NULL || info == NULL;
27 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

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

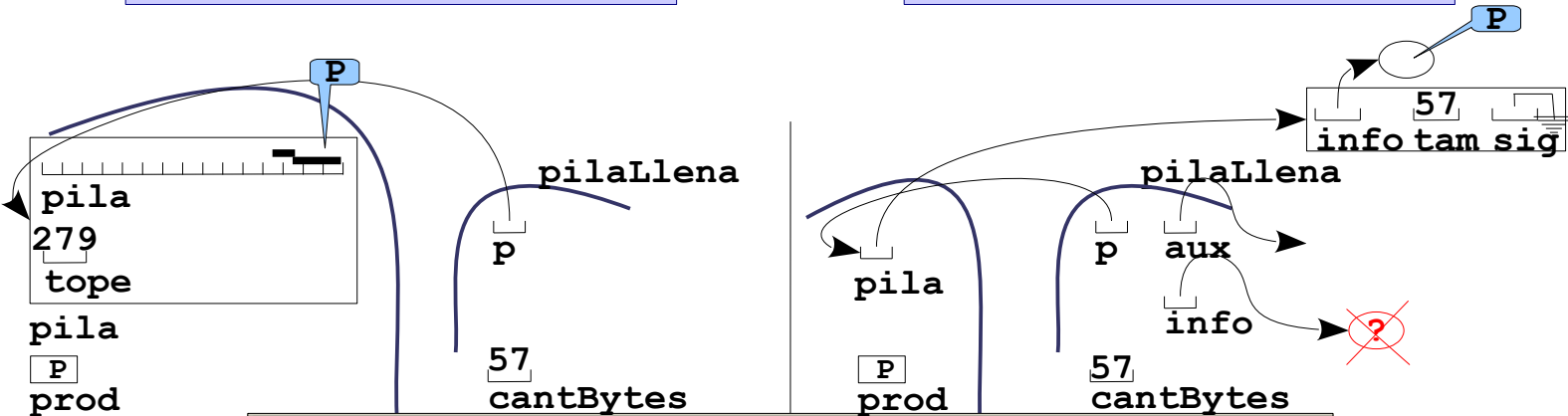
    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

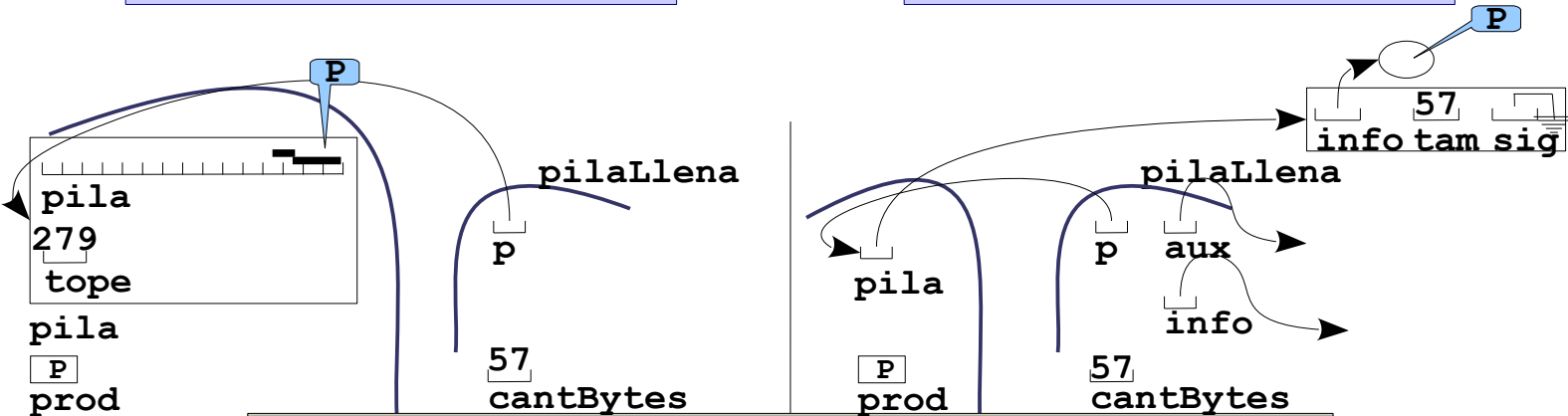
```
main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
19 int pilaLlena(const tPila *p, unsigned cantBytes)
20 {
21     tNodo *aux = (tNodo *)malloc(sizeof(tNodo));
22     void *info = malloc(cantBytes);
23
24     free(aux);
25     free(info);
26     return aux == NULL || info == NULL;
27 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

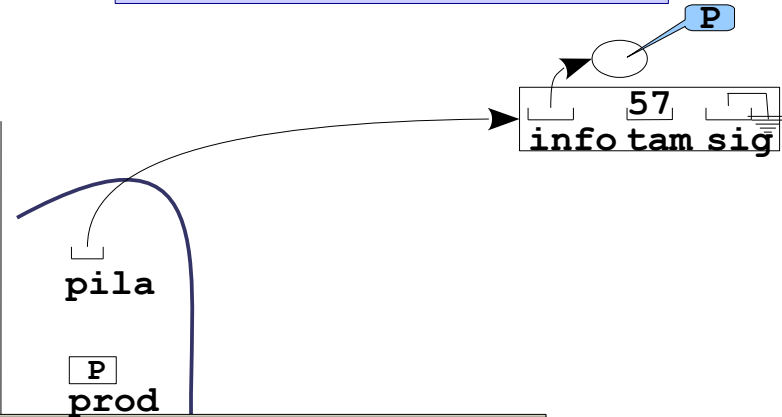
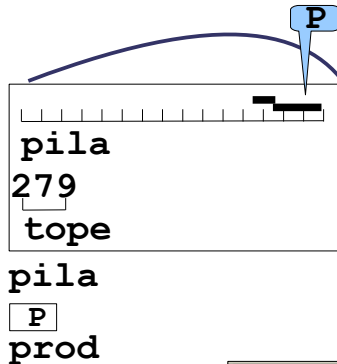
```
main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
19 int pilaLlena(const tPila *p, unsigned cantBytes)
20 {
21     tNodo *aux = (tNodo *)malloc(sizeof(tNodo));
22     void *info = malloc(cantBytes);
23
24     free(aux);
25     free(info);
26     return aux == NULL || info == NULL;
27 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



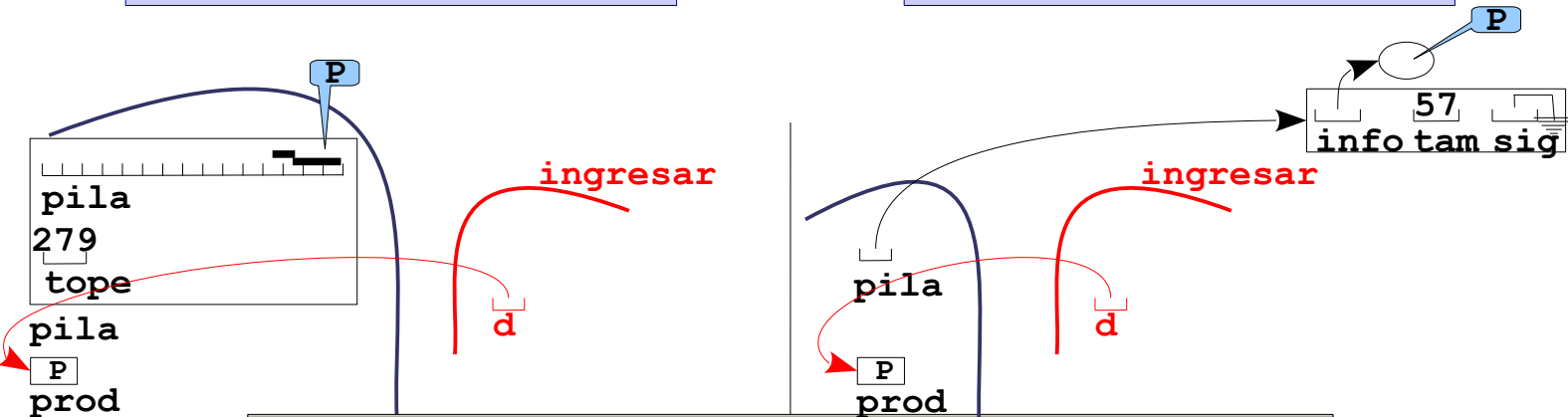
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd  prod;
89     tPila  pila;
90
91     crearPila(&pila);
92     while (!pilaLlena(&pila, sizeof(prod))) &&
93         ingresarProducto(&prod)
94         if (!ponerEnPila(&pila,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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



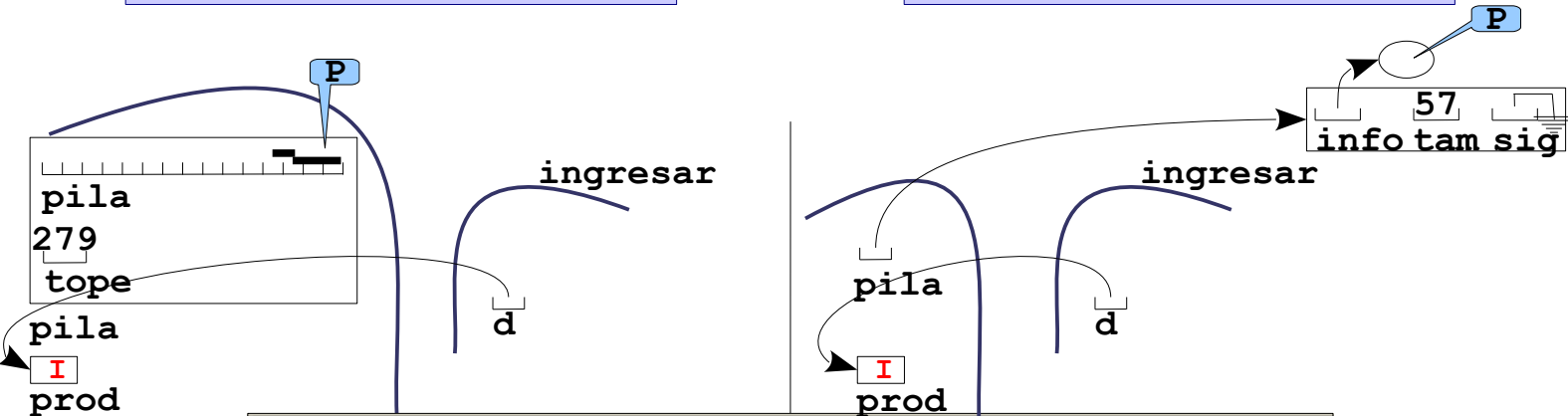
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd  prod;
89     tPila  pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93           ingresarProducto(&prod))
94         if(!ponerEnPila(&pila,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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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");
```

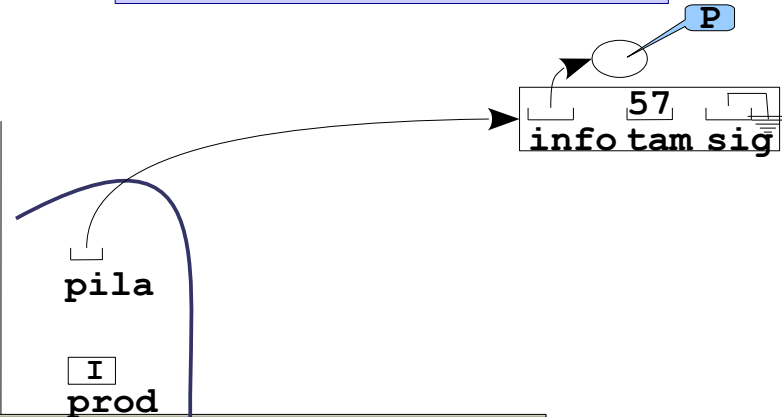
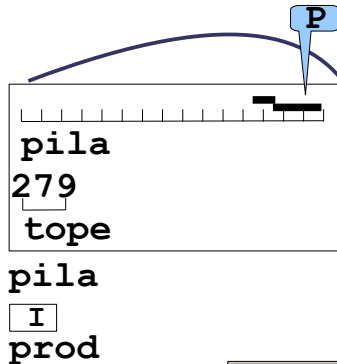


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



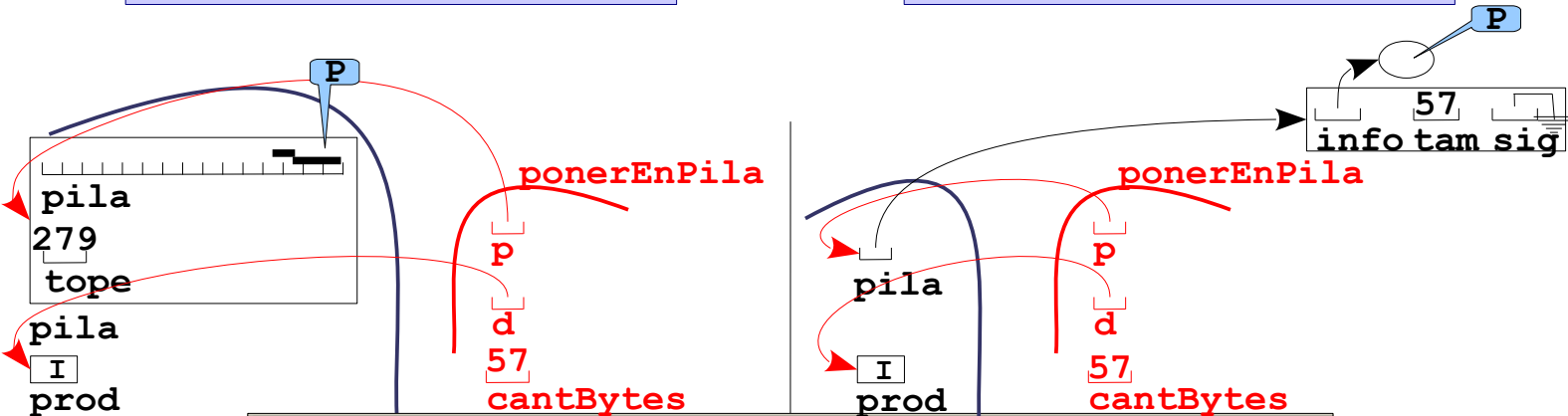
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd  prod;
89     tPila  pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93         ingresarProducto(&prod))
94         if(!ponerEnPila(&pila,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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



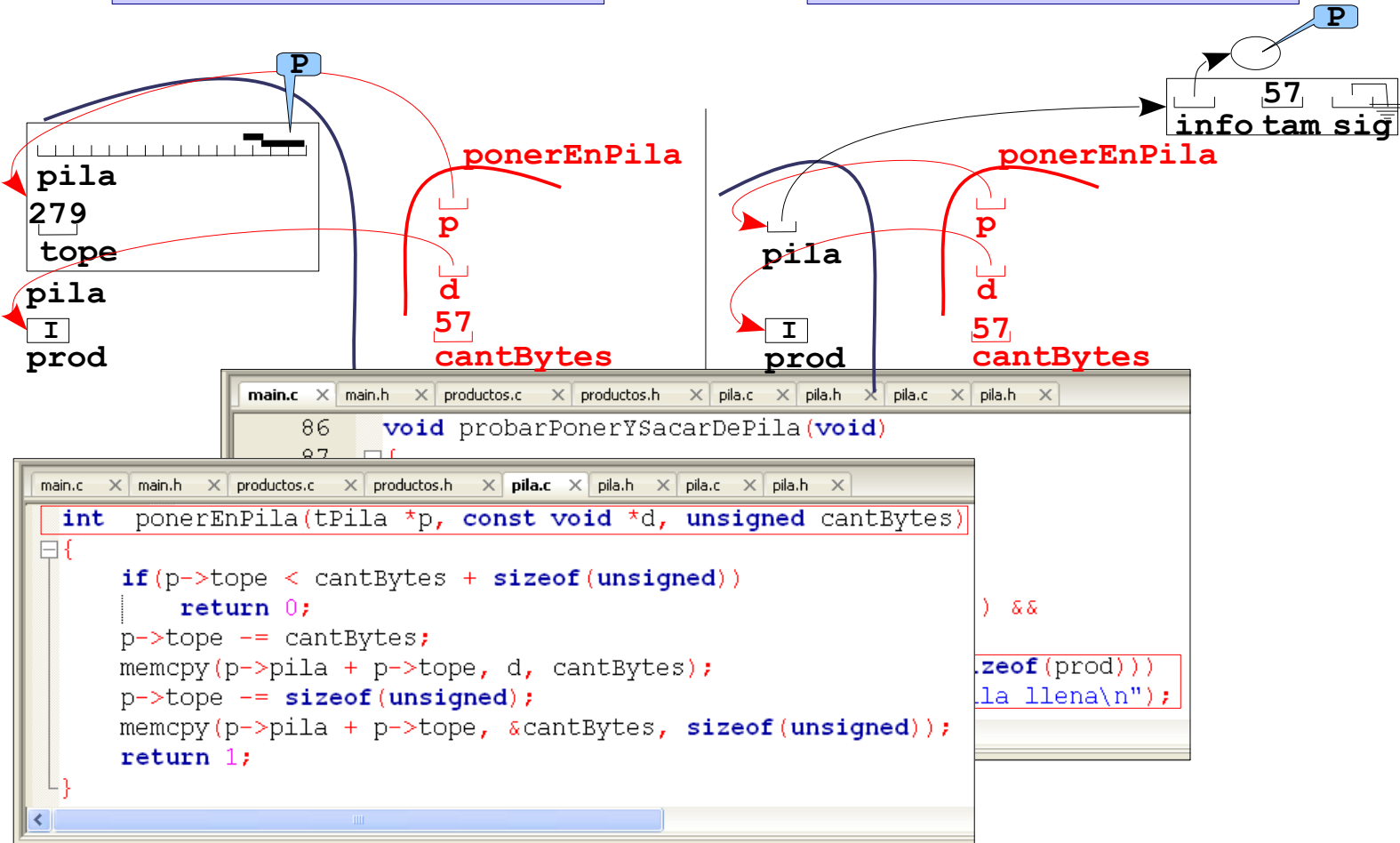
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd prod;
89     tPila pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93         ingresarProducto(&prod))
94         if(!ponerEnPila(&pila,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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica

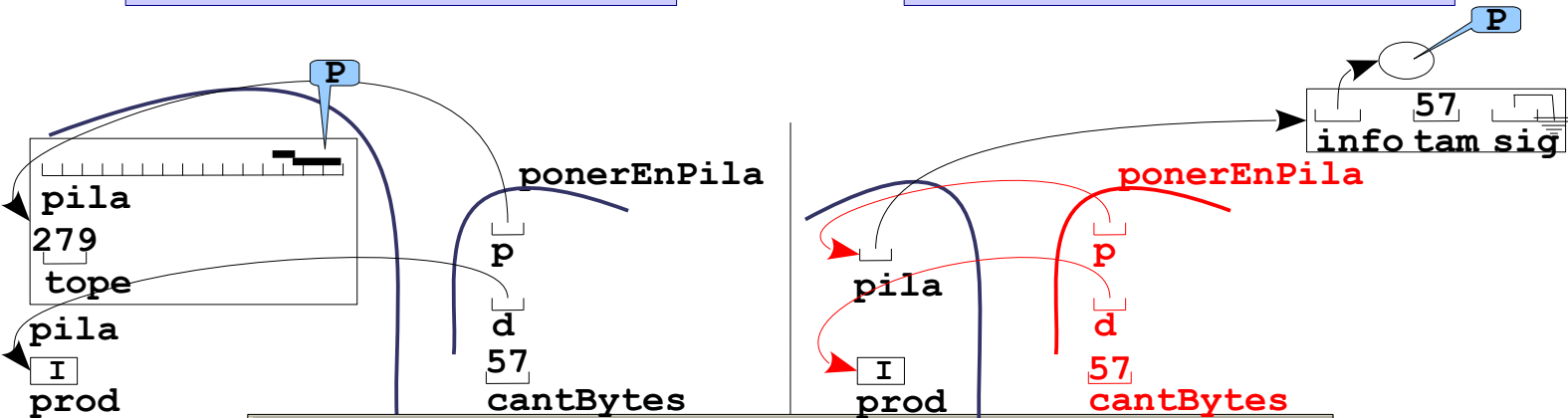


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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;
}
```

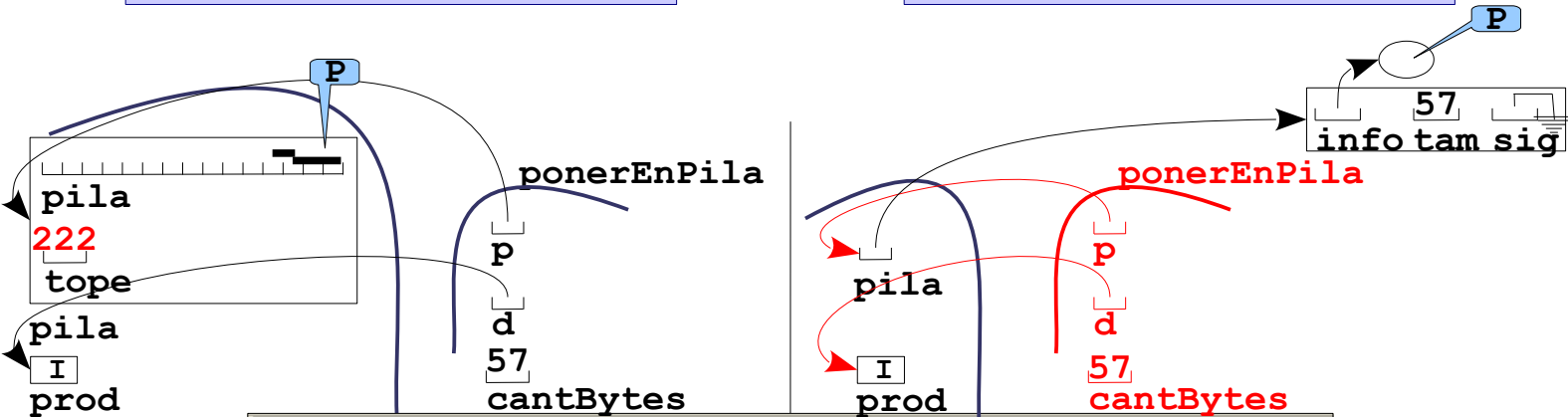
```
) &&
sizeof(prod))
lla llena\n");
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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;
}
```

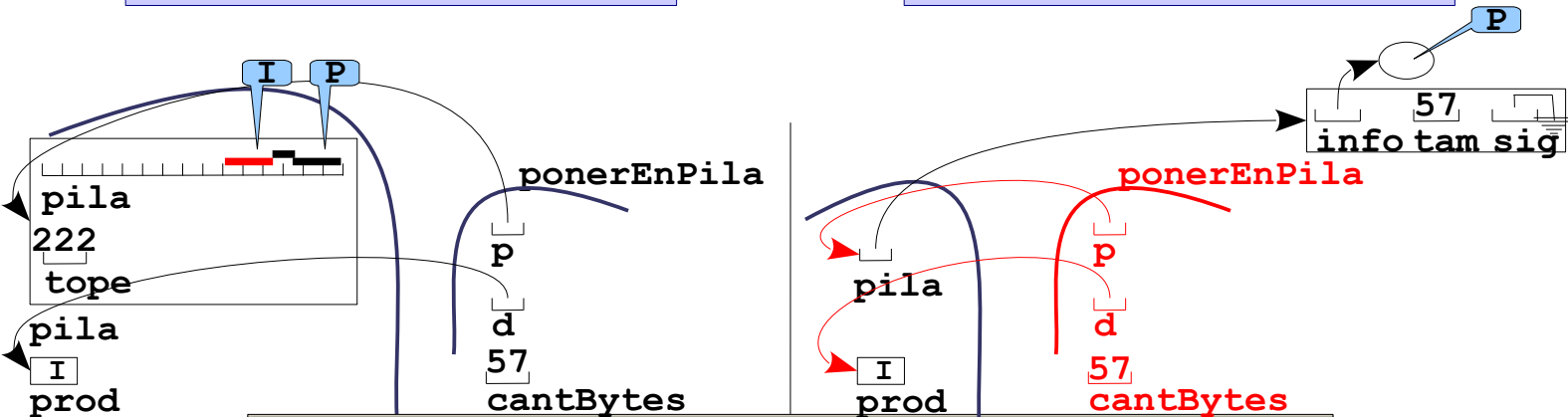
```
) &&
sizeof(prod)))
la llena\n");
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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;
}
```

```
) &&
sizeof(prod)))
la llena\n");
```

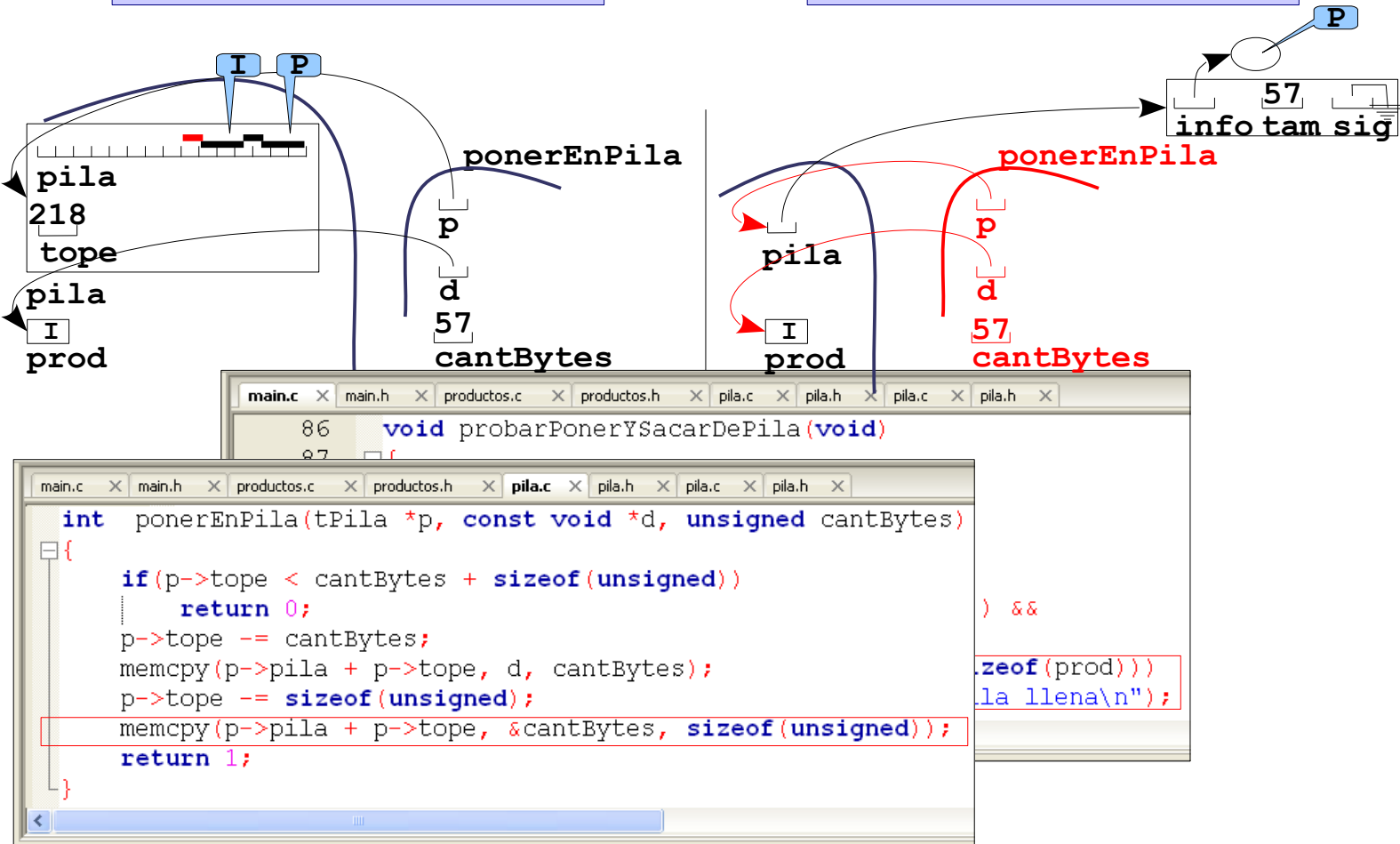


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



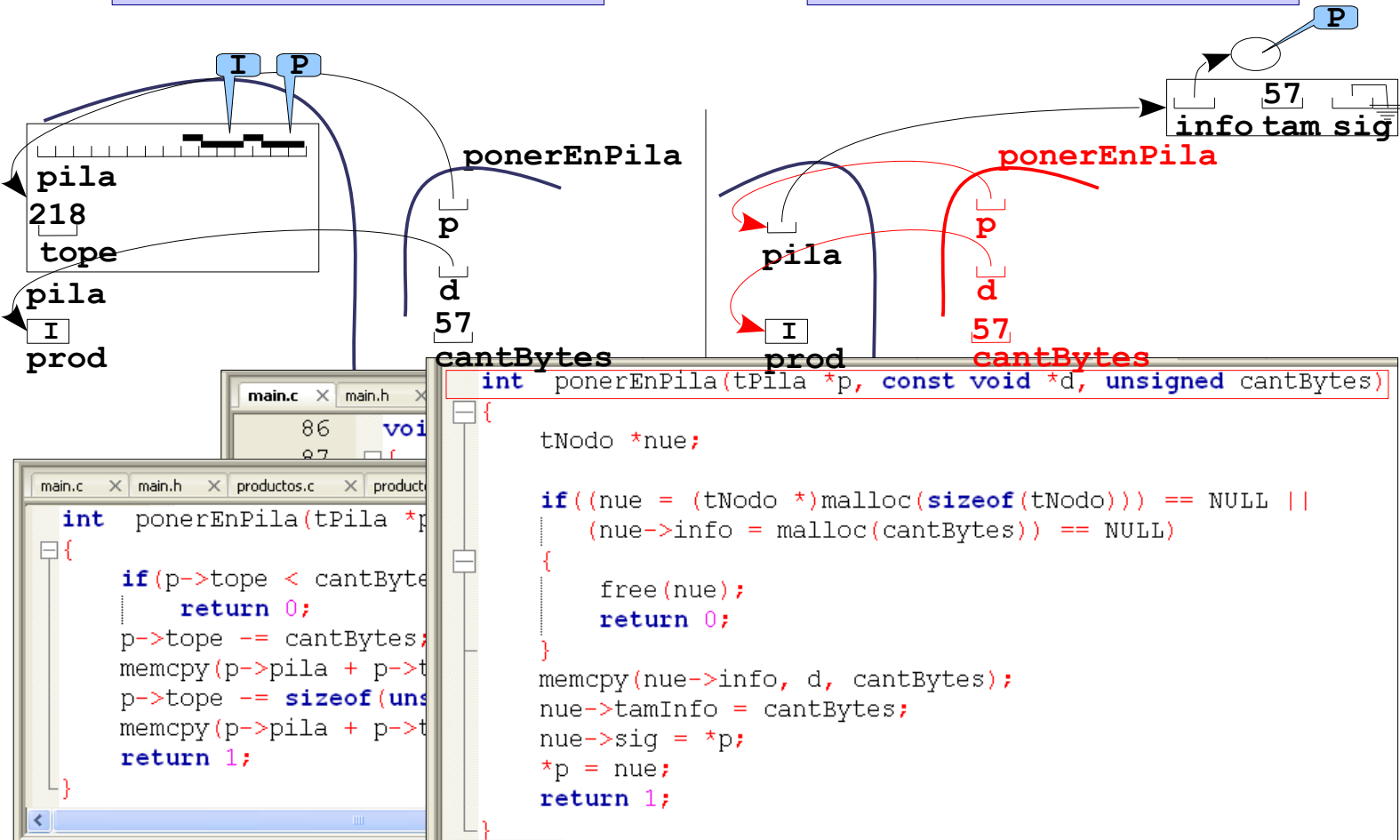


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica

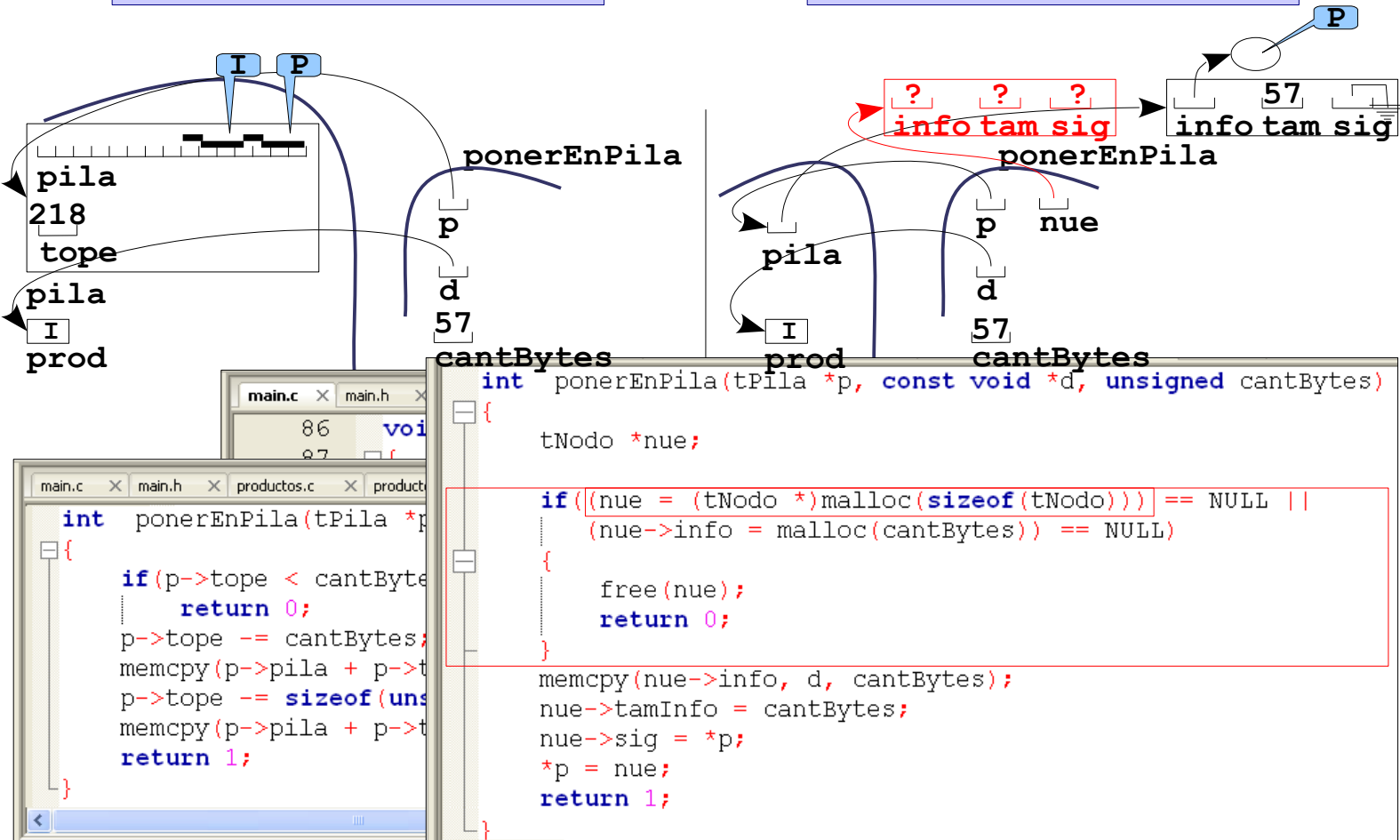


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica

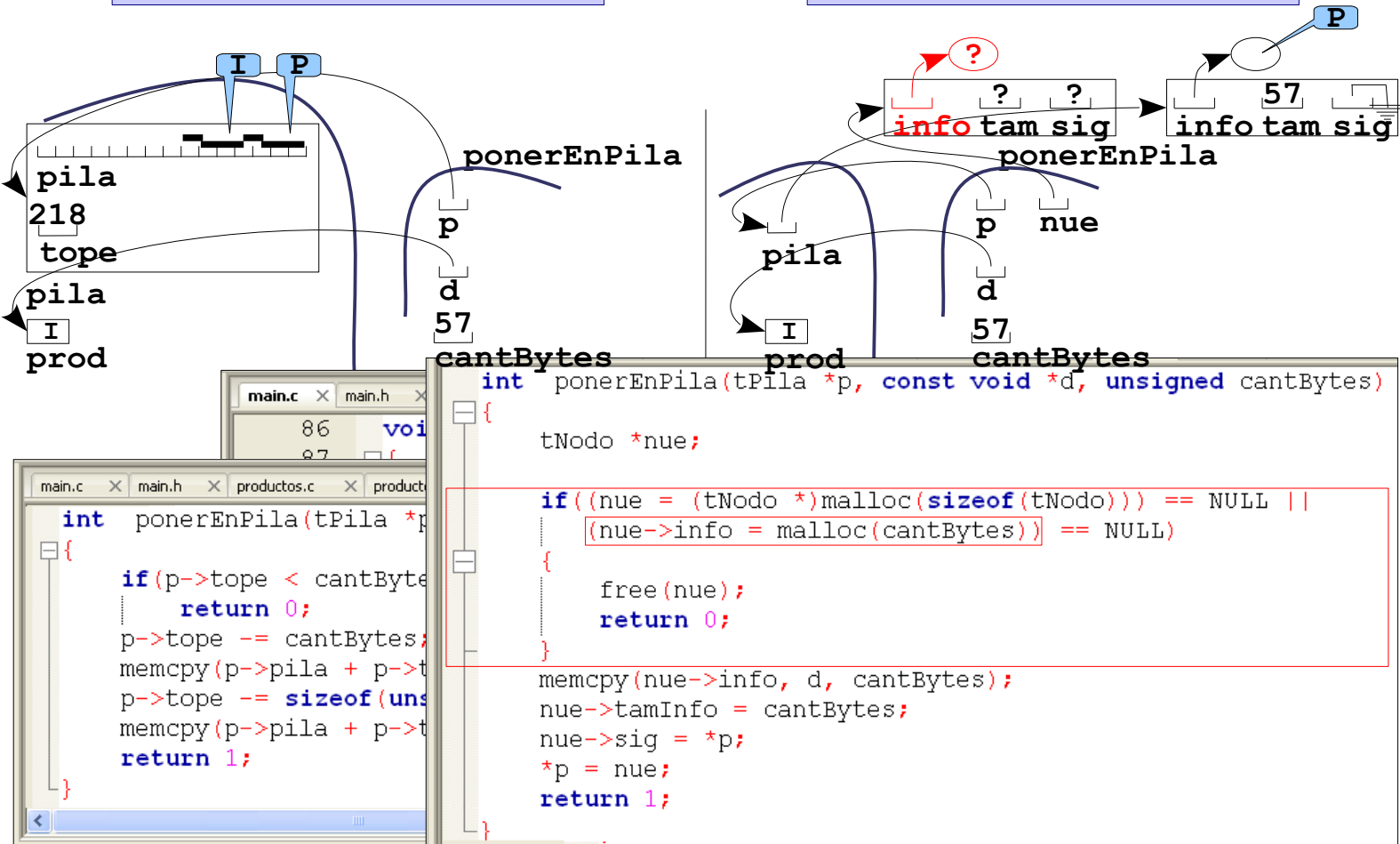


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

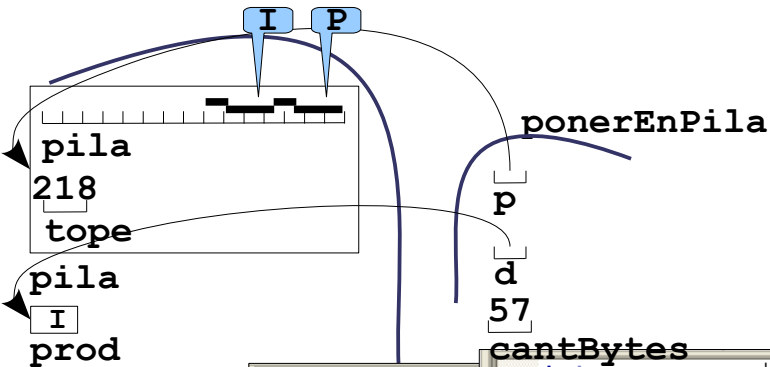
### Implementación dinámica



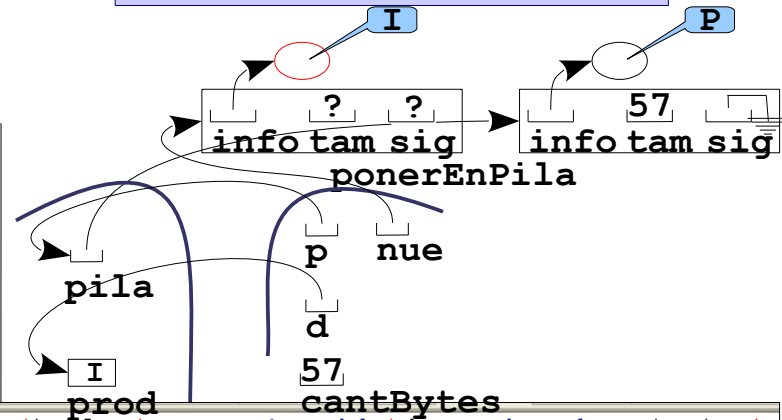
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



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

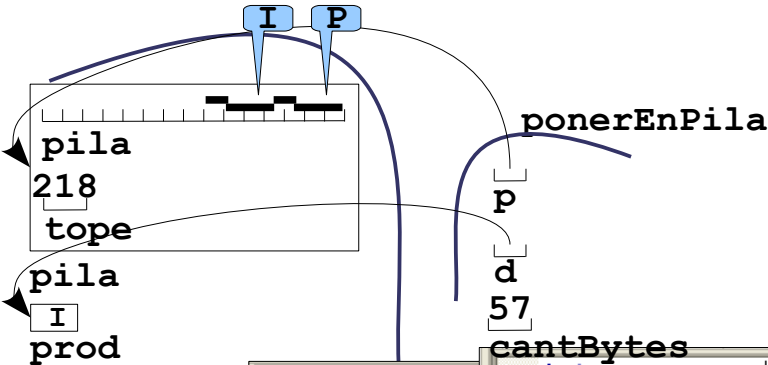
    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }

    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

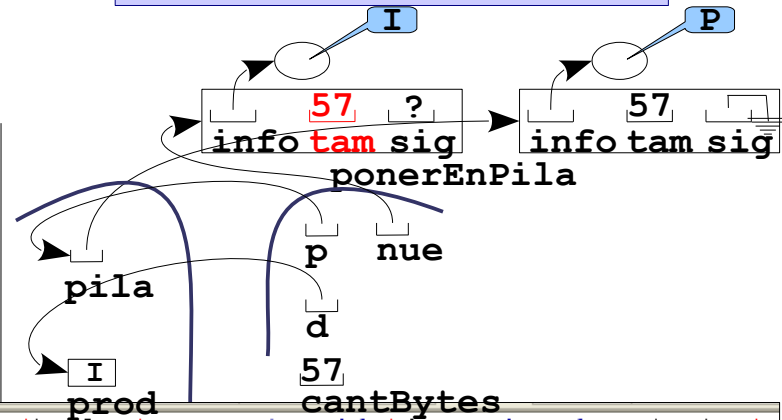
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



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

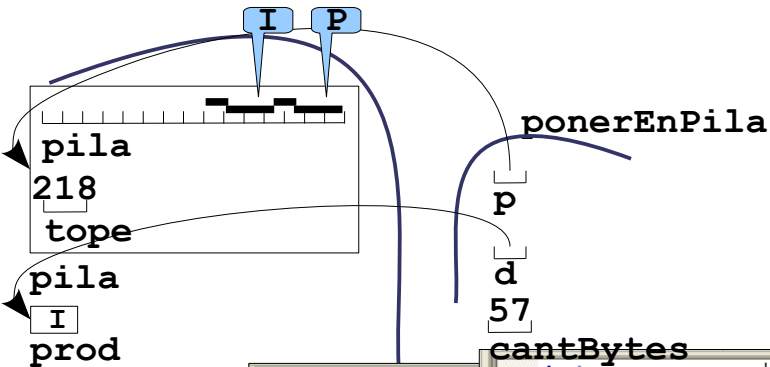
    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }

    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

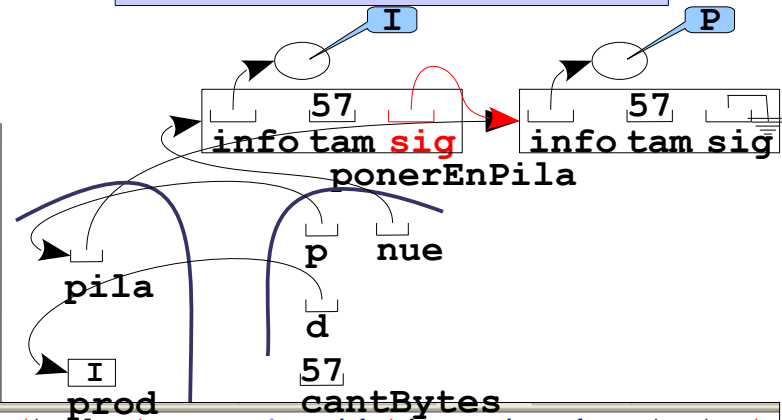
# Tipos de Datos Abstractos

## 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;
}
```



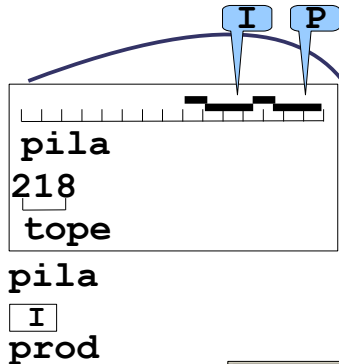




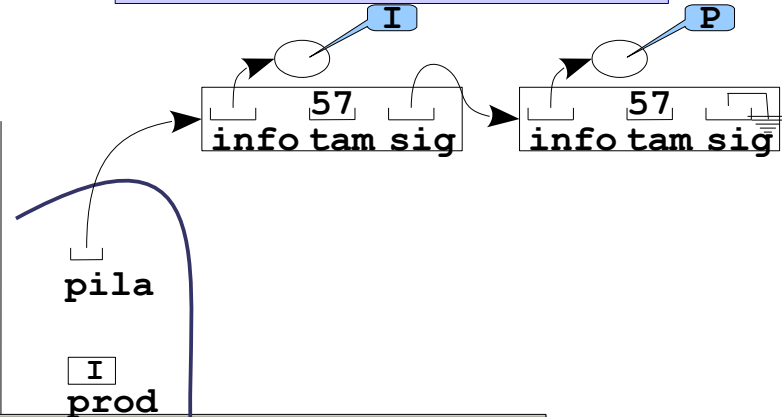
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica

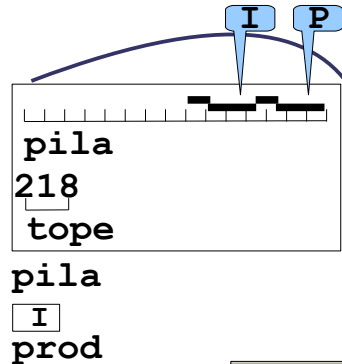


```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd    prod;
89     tPila    pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93           ingresarProducto(&prod))
94         if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95             puts("ERROR - inesperado: pila llena\n");
96 }
```

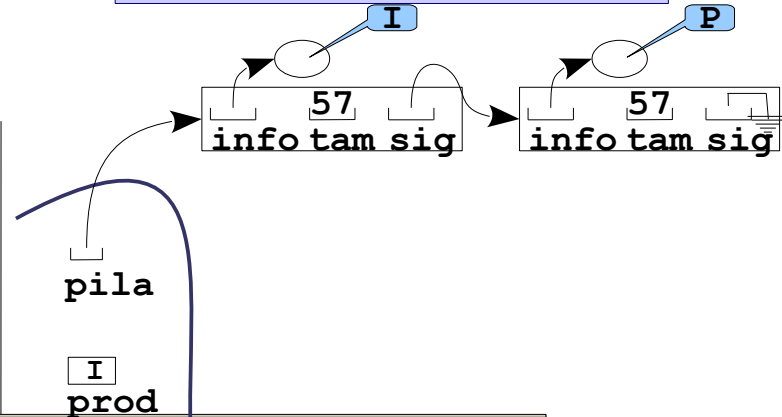
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica

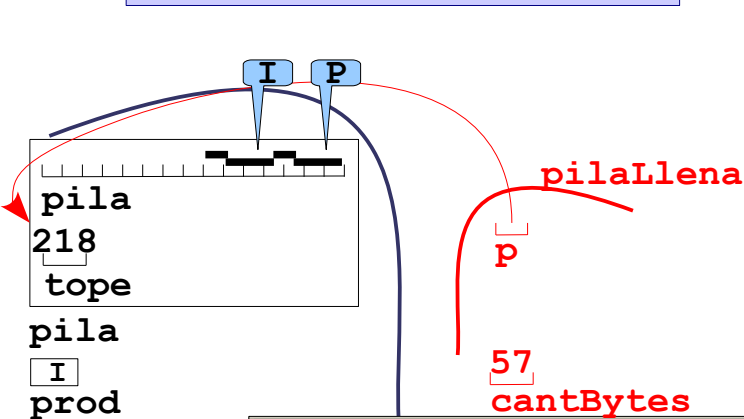


```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd    prod;
89     tPila    pila;
90
91     crearPila(&pila);
92     while (!pilaLlena(&pila, sizeof(prod)) &&
93           ingresarProducto(&prod))
94         if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

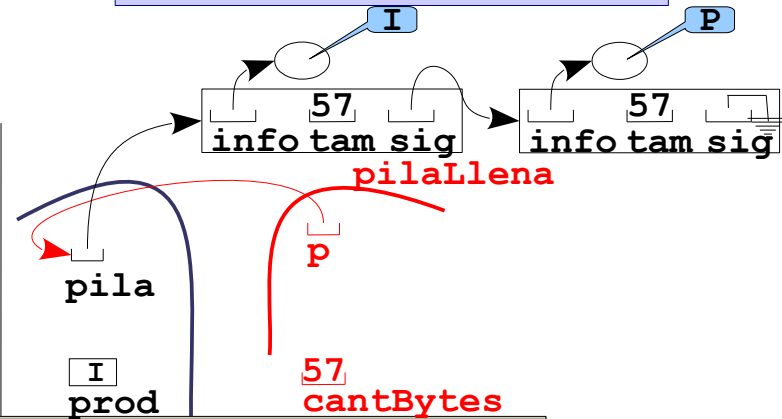
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica

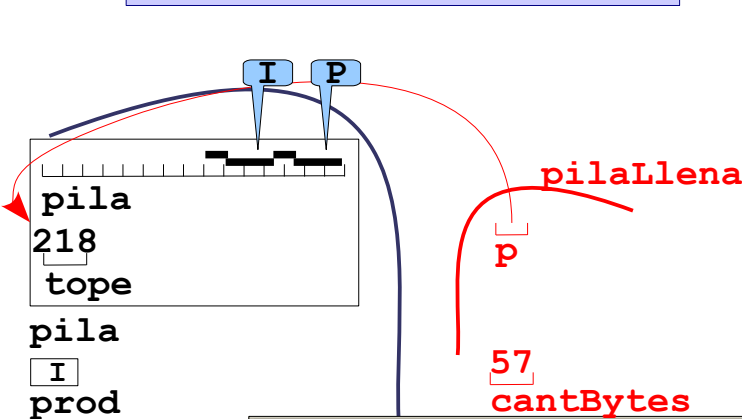


```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd prod;
89     tPila pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93         ingresarProducto(&prod))
94         if(!ponerEnPila(&pila,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

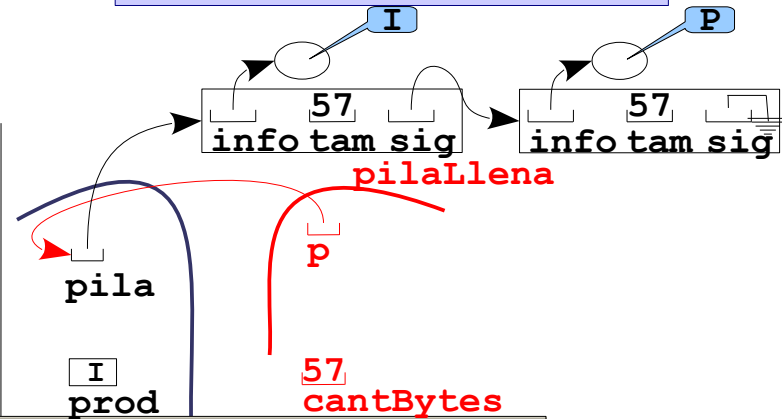
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

```
main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
19 int pilaLlena(const tPila *p, unsigned cantBytes)
20 {
21     tNodo *aux = (tNodo *)malloc(sizeof(tNodo));
22     void *info = malloc(cantBytes);
23
24     free(aux);
25     free(info);
26     return aux == NULL || info == NULL;
27 }
```



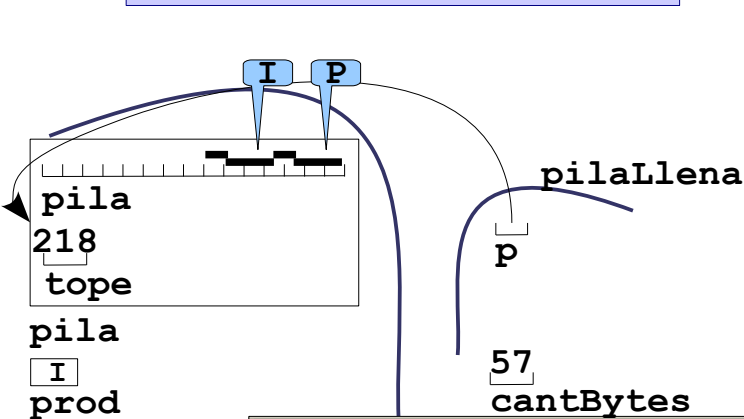




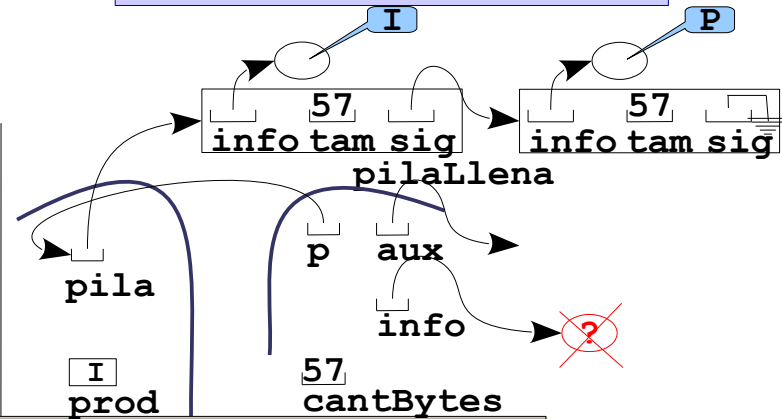
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

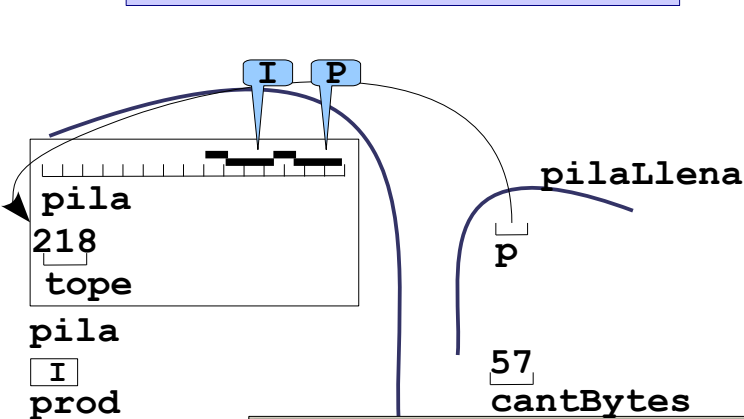
```
main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
19 int pilaLlena(const tPila *p, unsigned cantBytes)
20 {
21     tNodo *aux = (tNodo *)malloc(sizeof(tNodo));
22     void *info = malloc(cantBytes);
23     free(aux);
24     free(info);
25     return aux == NULL || info == NULL;
26 }
```



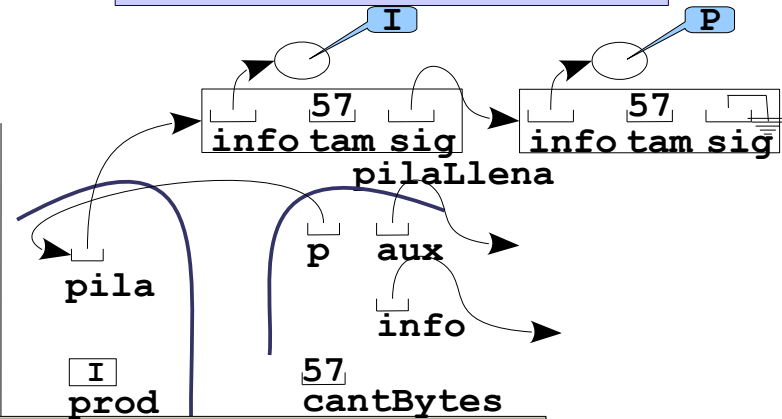
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



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

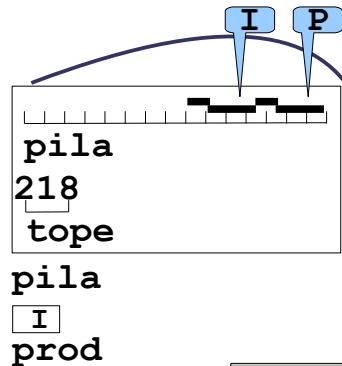
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
14 int pilaLlena(const tPila *p, unsigned cantBytes)
15 {
16     return p->tope < cantBytes + sizeof(unsigned);
17 }
18
```

```
main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
19 int pilaLlena(const tPila *p, unsigned cantBytes)
20 {
21     tNodo *aux = (tNodo *)malloc(sizeof(tNodo));
22     void *info = malloc(cantBytes);
23     free(aux);
24     free(info);
25     return aux == NULL || info == NULL;
26 }
```

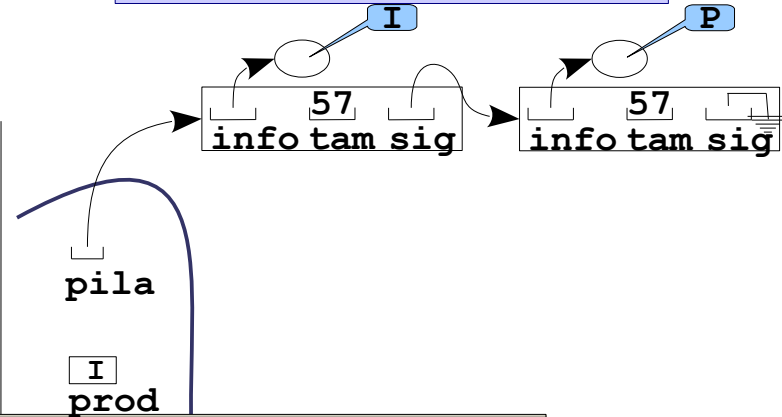
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica

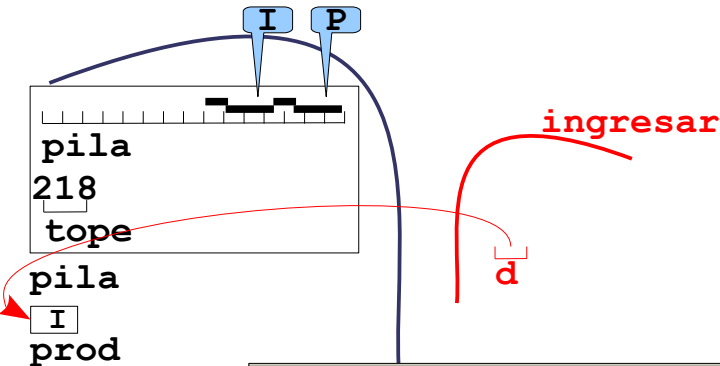


```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd  prod;
89     tPila  pila;
90
91     crearPila(&pila);
92     while (!pilaLlena(&pila, sizeof(prod))) {
93         ingresarProducto(&prod);
94         if (!ponerEnPila(&pila, &prod, sizeof(prod)))
95             puts("ERROR - inesperado: pila llena\n");
96     }
```

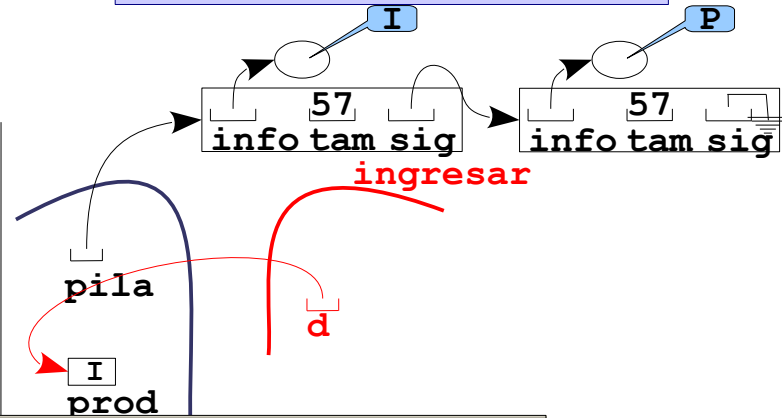
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica

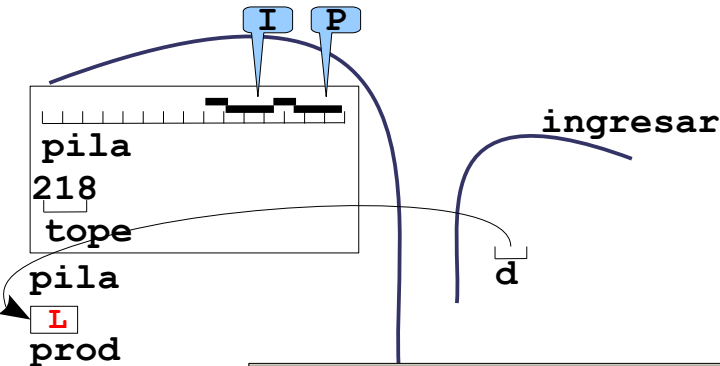


```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd  prod;
89     tPila  pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93         ingresarProducto(&prod))
94         if(!ponerEnPila(&pila,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

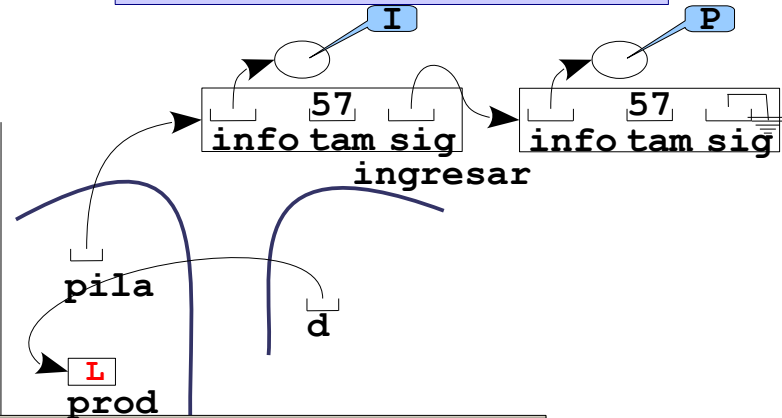
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica

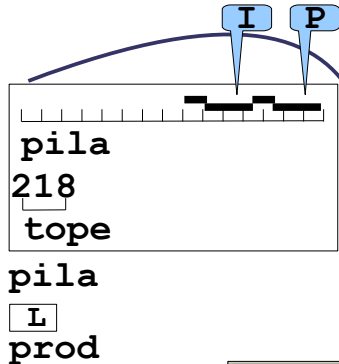


```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd prod;
89     tPila pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93         ingresarProducto(&prod))
94         if(!ponerEnPila(&pila,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

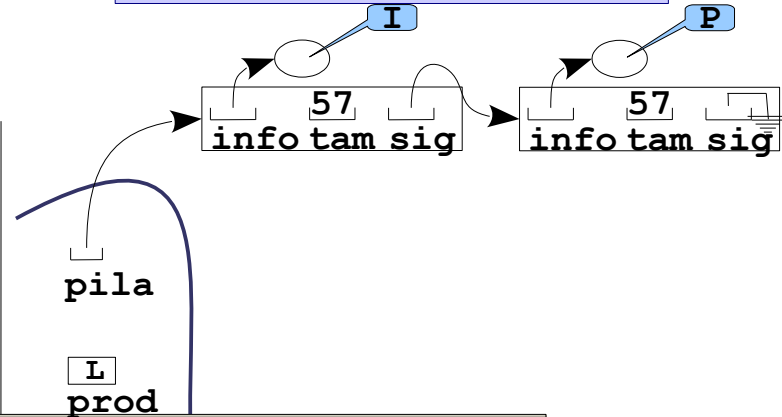
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica

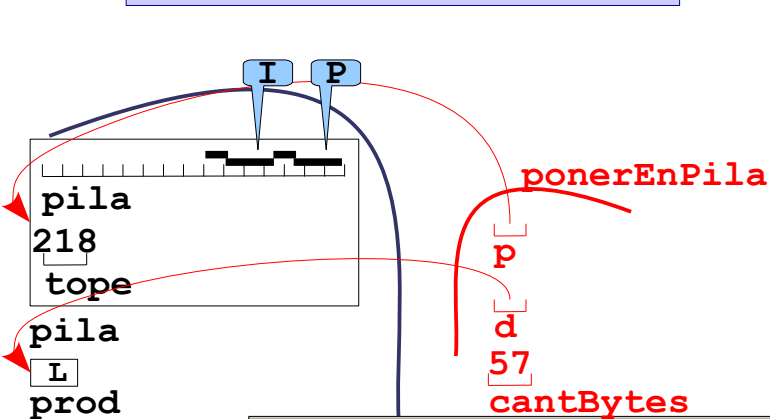


```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd prod;
89     tPila pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93         ingresarProducto(&prod))
94         if(!ponerEnPila(&pila,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

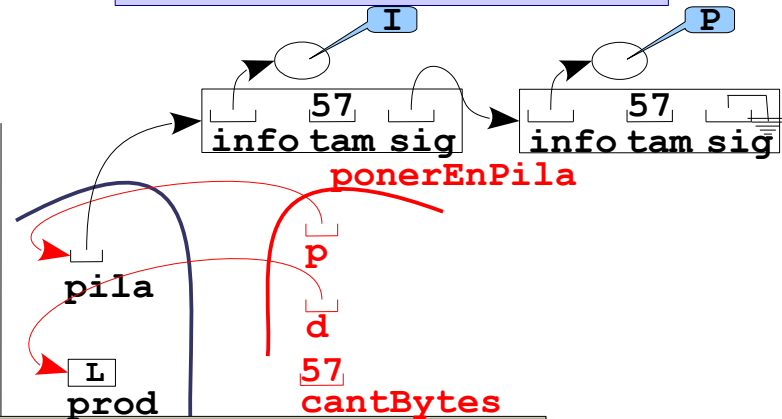
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica

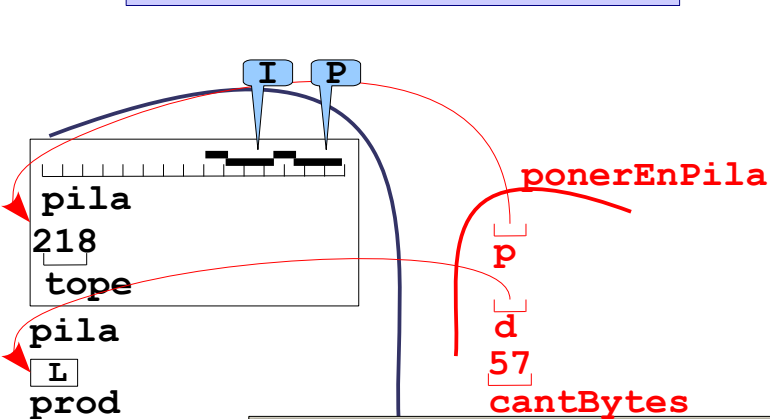


```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd prod;
89     tPila pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93         ingresarProducto(&prod))
94         if(!ponerEnPila(&pila,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

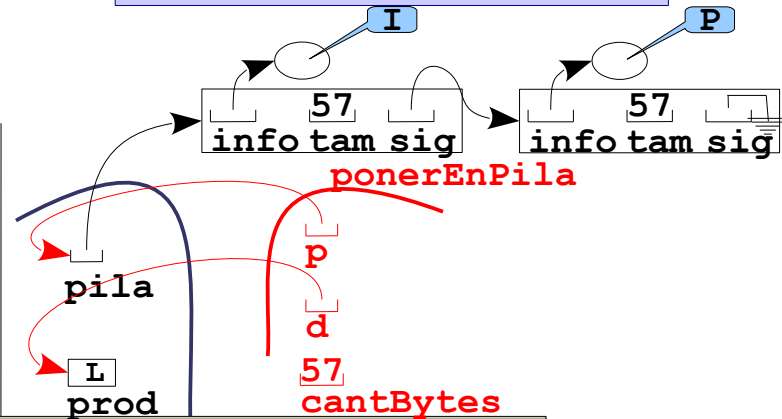
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

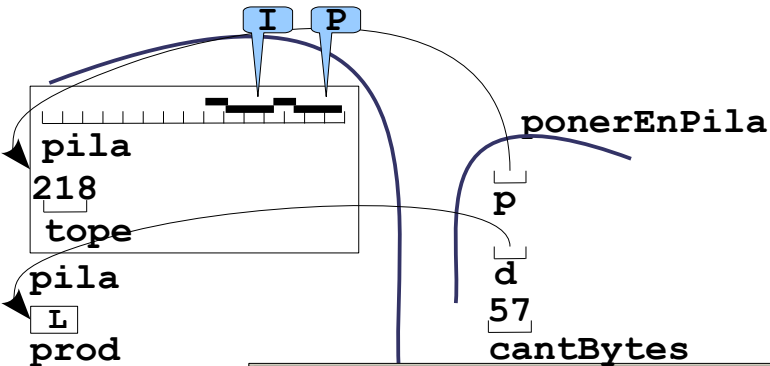
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
int ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);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

```
) &&
sizeof(prod))
la llena\n");
```

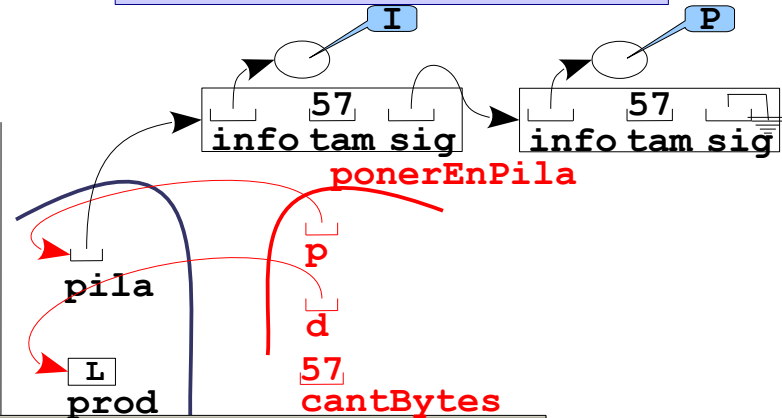
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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;
}
```

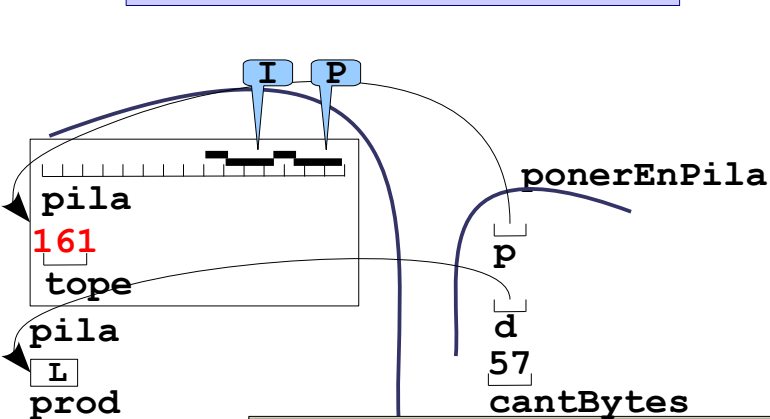
```
) &&
sizeof(prod))
la llena\n");
```



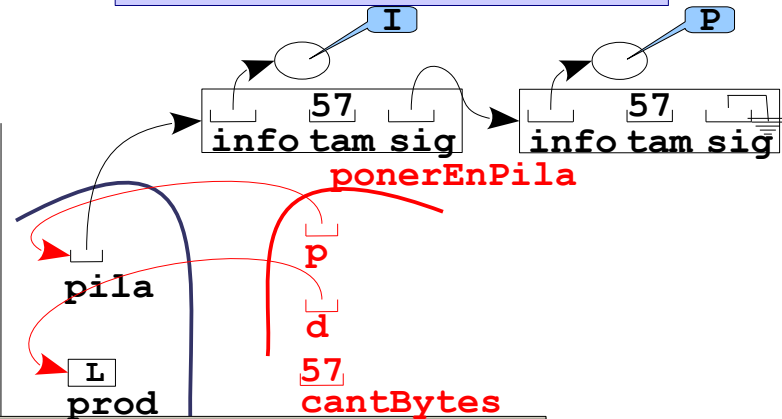
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

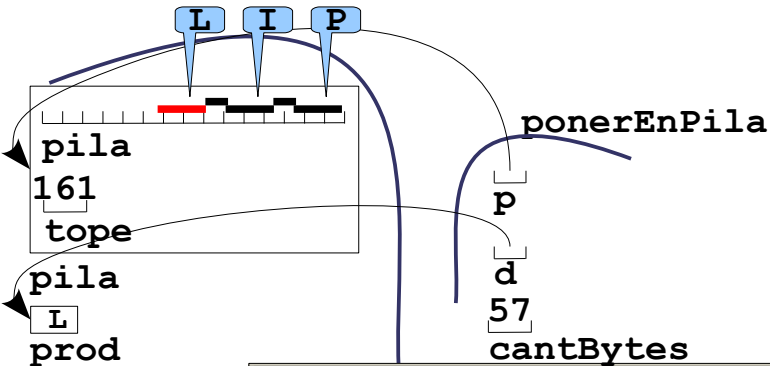
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
int ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);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

```
) &&
sizeof(prod)))
la llena\n");
```

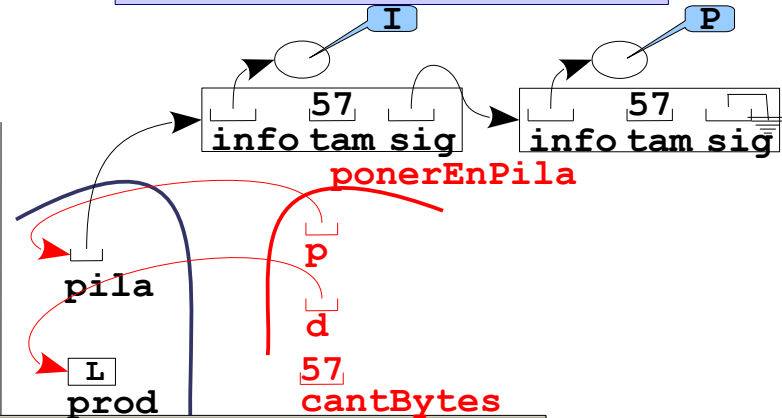
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

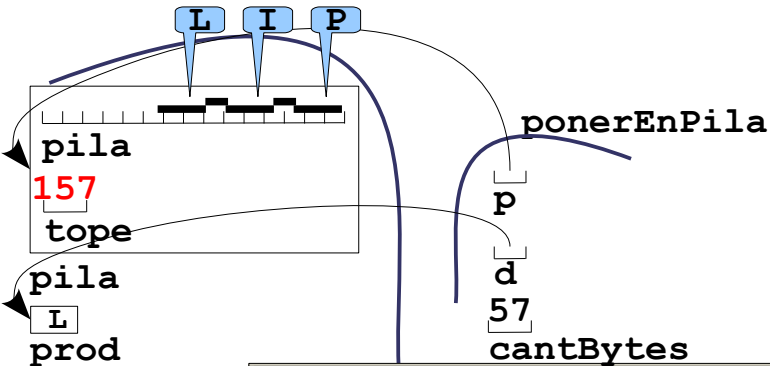
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
int ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);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

```
) &&
sizeof(prod))
la llena\n");
```

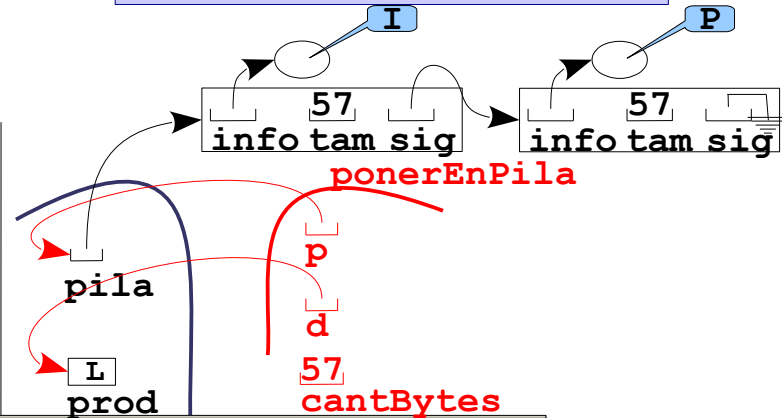
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

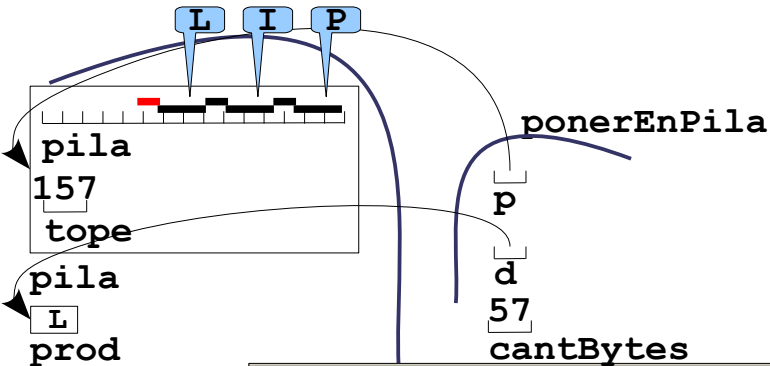
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
int ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);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

```
) &&
sizeof(prod))
la llena\n");
```

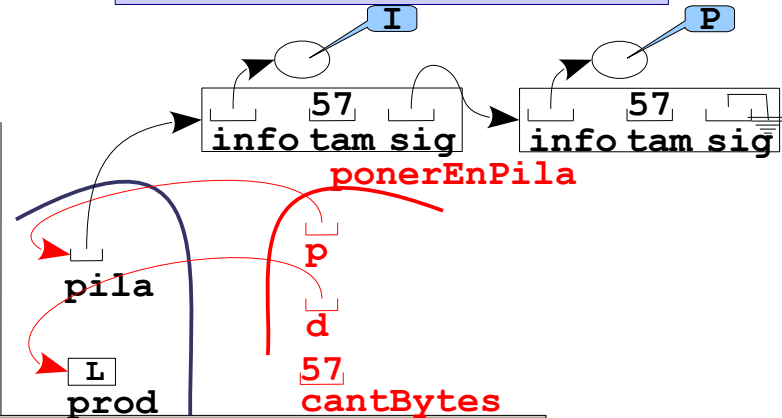
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
```

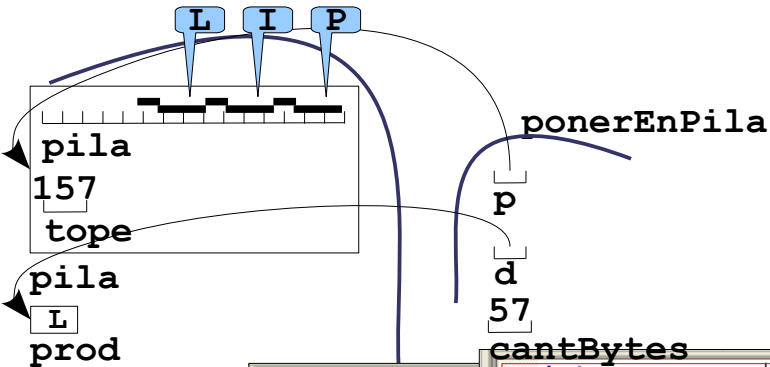
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
int ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    if(p->tope < cantBytes + sizeof(unsigned))
        return 0;
    p->tope -= cantBytes;
    memcpy(p->pila + p->tope, d, cantBytes);
    p->tope -= sizeof(unsigned);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

```
) &&
sizeof(prod))
la llena\n");
```

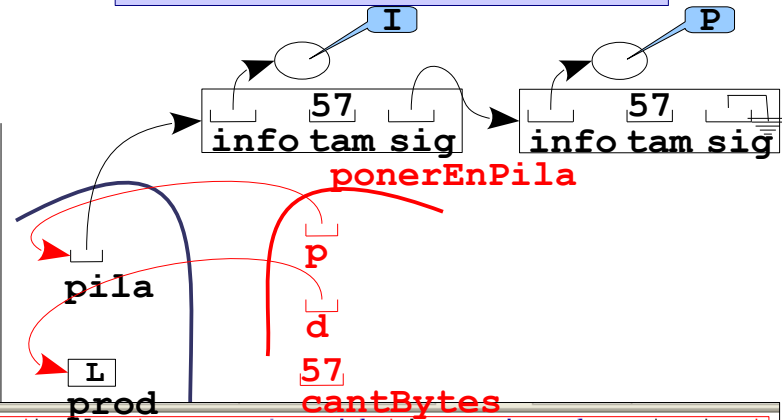
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
int ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
```

```
{
    tNodo *nue;

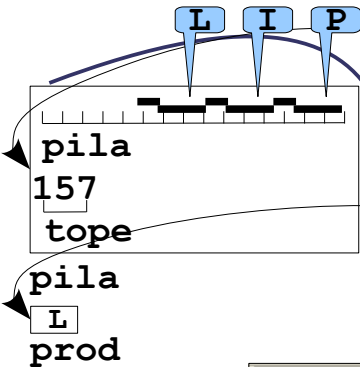
    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||
       (nue->info = malloc(cantBytes)) == NULL)
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info, d, cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}
```

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

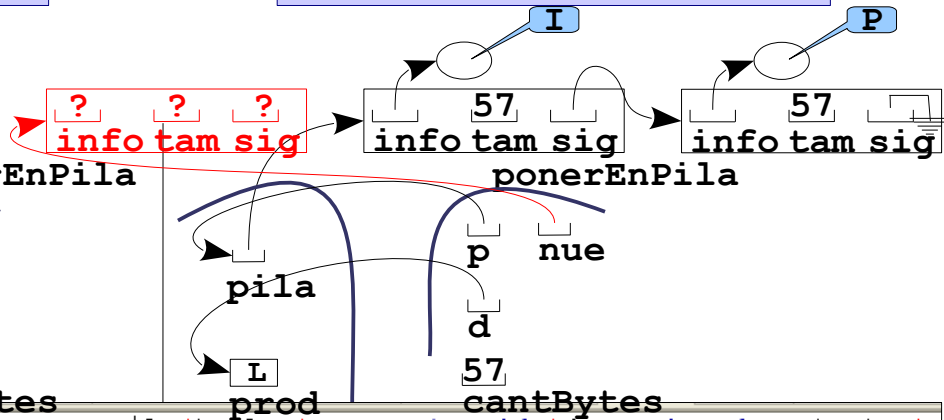
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



main.c x main.h x  
86 voi  
87 f

```
int ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
```

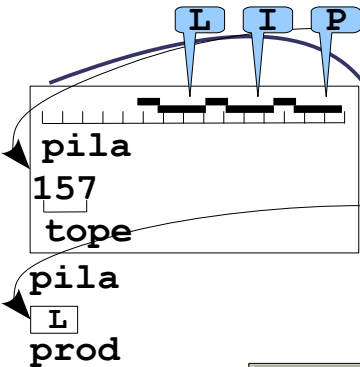
```
{  
    tNodo *nue;  
  
    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL ||  
       (nue->info = malloc(cantBytes)) == NULL)  
    {  
        free(nue);  
        return 0;  
    }  
  
    memcpy(nue->info, d, cantBytes);  
    nue->tamInfo = cantBytes;  
    nue->sig = *p;  
    *p = nue;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

```
main.c x main.h x productos.c x product  
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;  
}
```

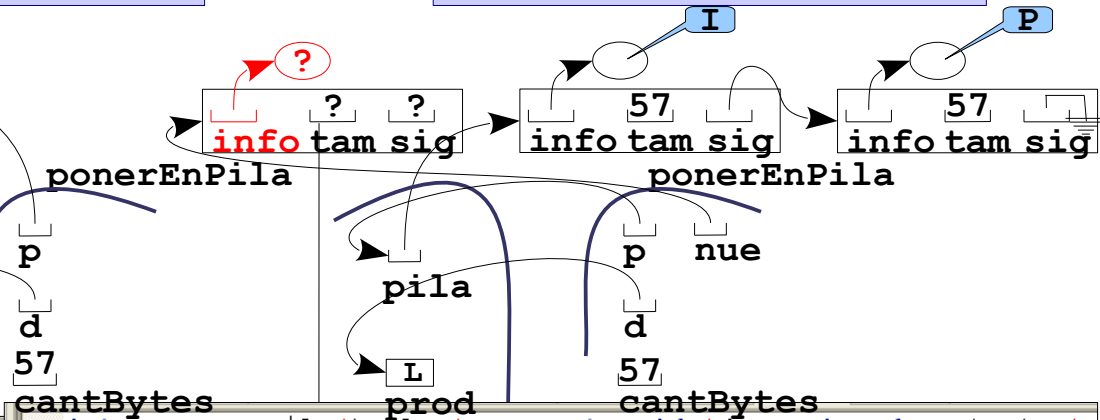
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
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;
}
```



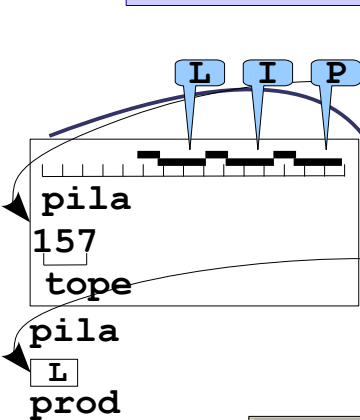




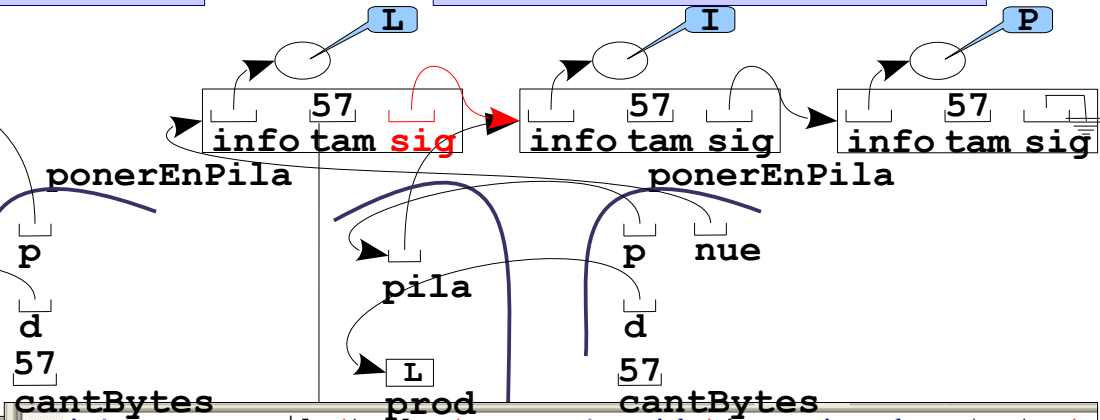
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



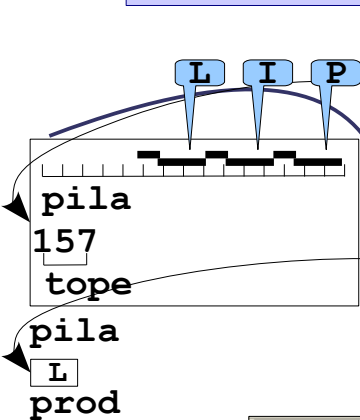
```
int ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    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;
}
```

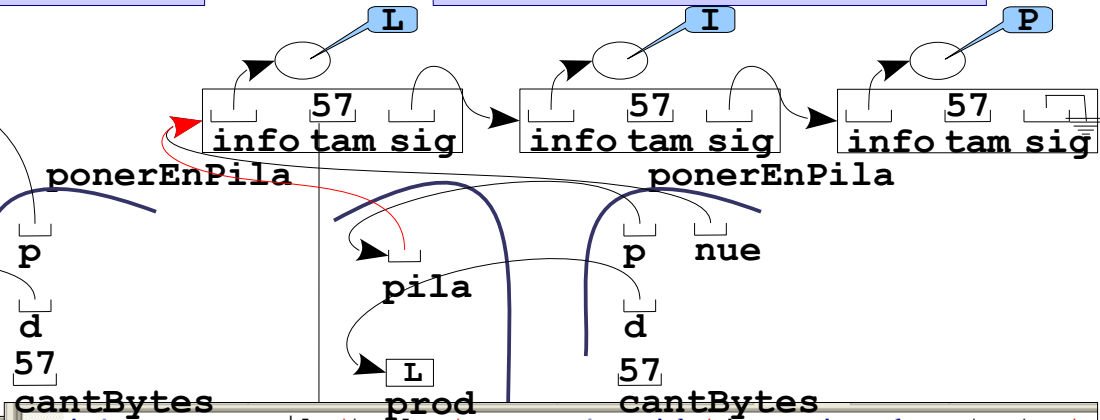
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



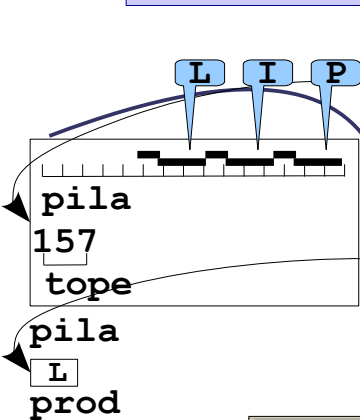
```
int ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    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;
}
```

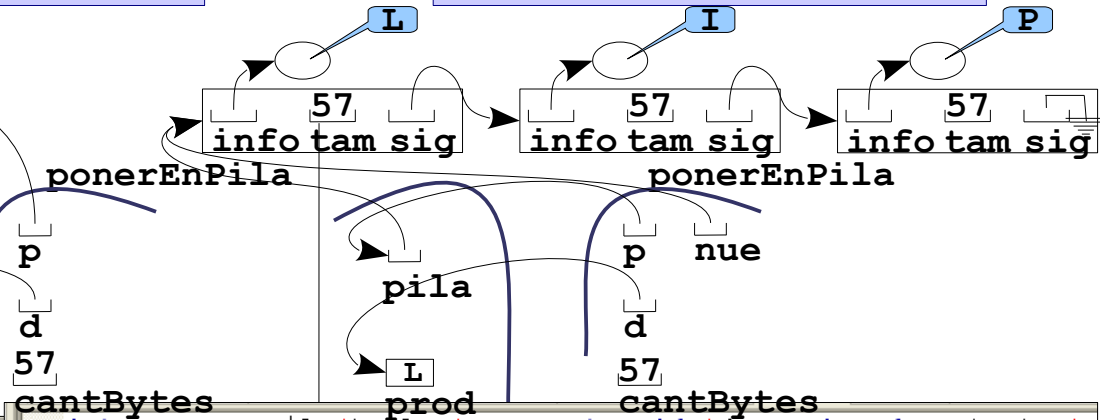
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



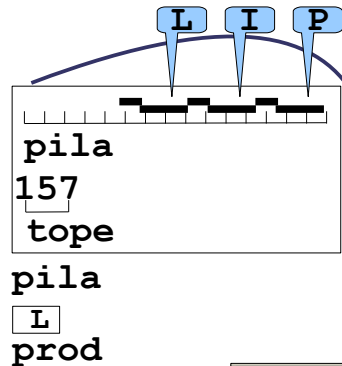
```
int ponerEnPila(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;

    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;
}
```

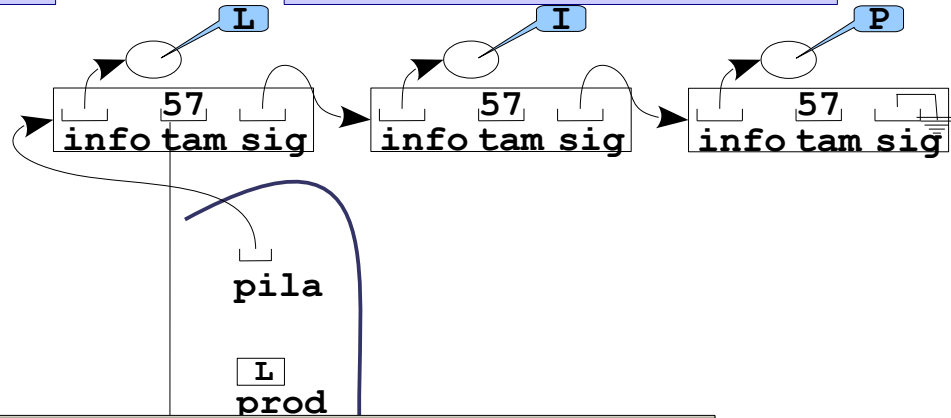
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica

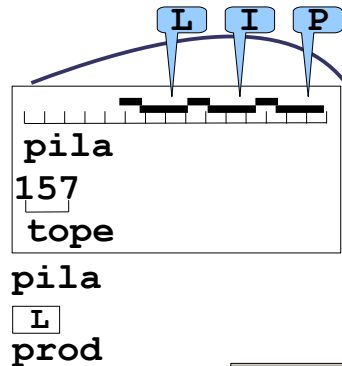


```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd  prod;
89     tPila  pila;
90
91     crearPila(&pila);
92     while(!pilaLlena(&pila, sizeof(prod)) &&
93         ingresarProducto(&prod))
94         if(!ponerEnPila(&pila, &prod, sizeof(prod)))
95             puts("ERROR - inesperado: pila llena\n");
96 }
```

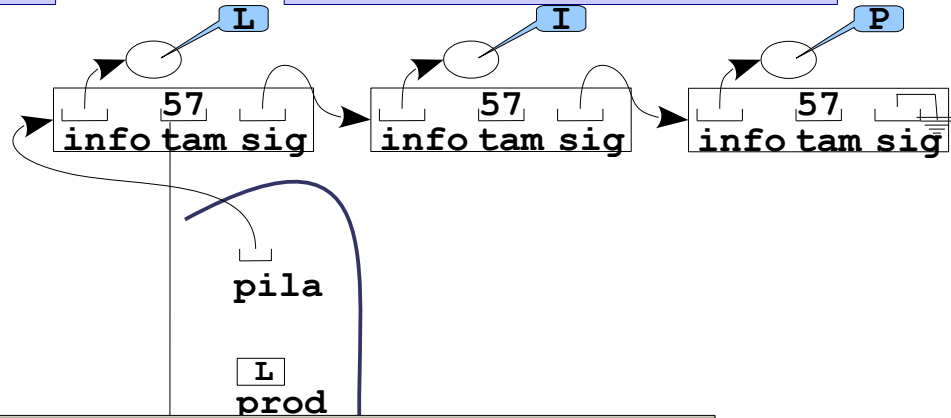
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



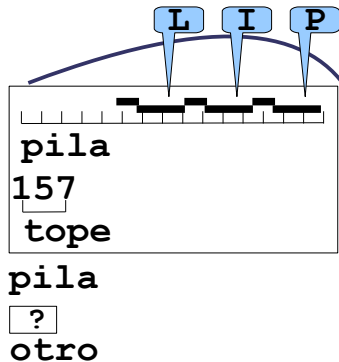
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
86 void probarPonerYSacarDePila(void)
87 {
88     tProd  prod;
89     tPila  pila;
90
91     crearPila(&pila);
92     while (!pilaLlena(&pila, sizeof(prod)) &&
93           ingresarProducto(&prod))
94         if (!ponerEnPila(&pila,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



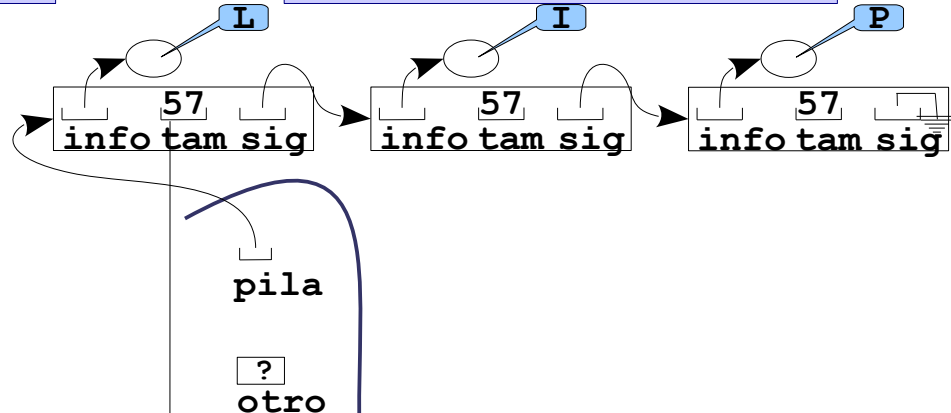
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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.");
```

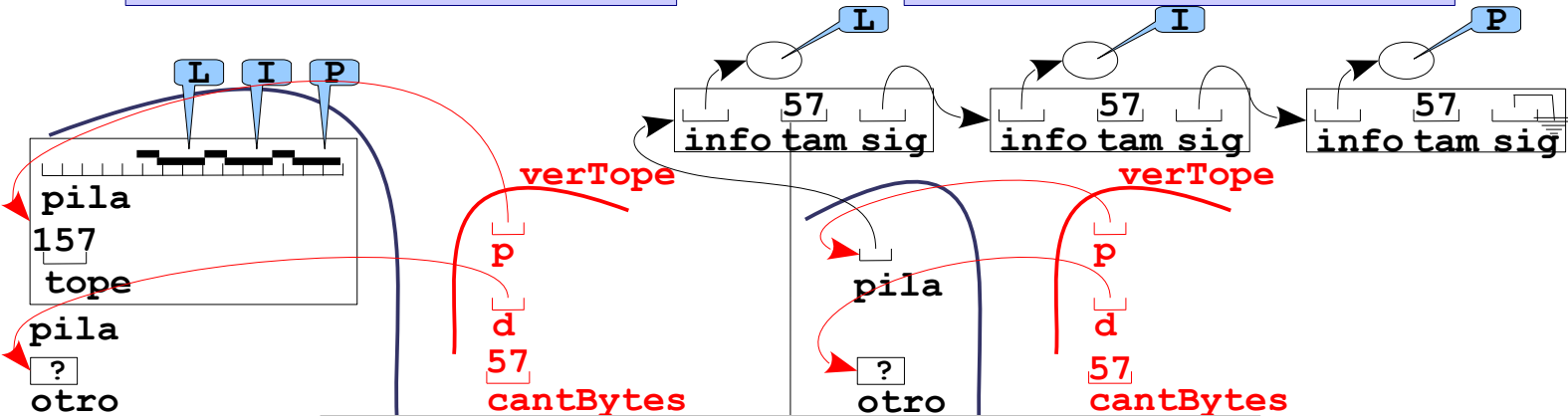


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



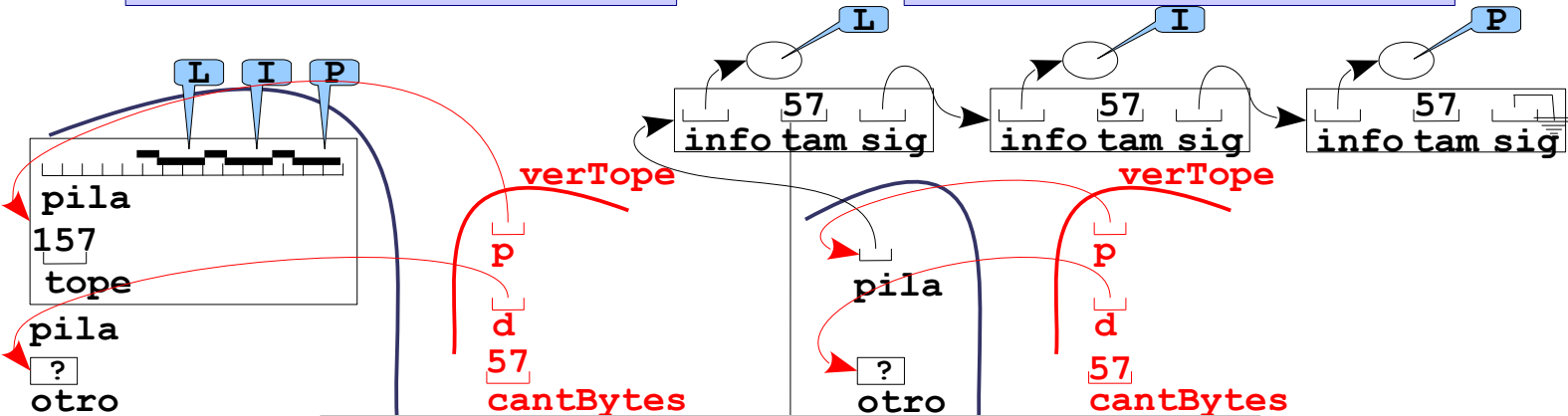
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
107 puts("Ver tope de la pila:");
108 if(verTope(&pila, &otro, sizeof(otro)))
109     mostrarProducto(&otro);
110 else
111     puts("La pila estaba vacia.");
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x

107 nuts("Ver tope de la pila:");

main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x

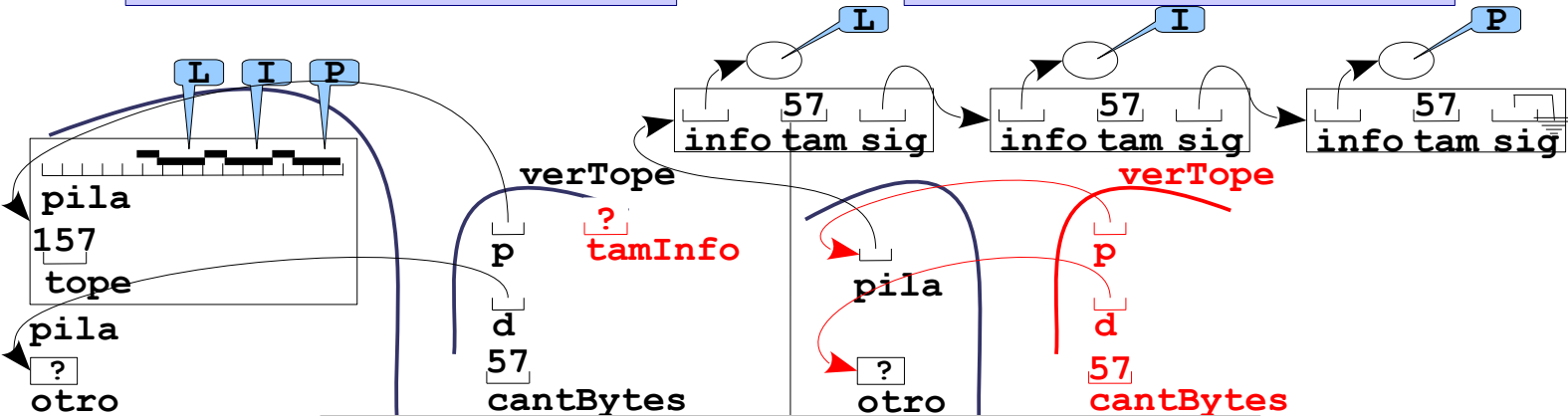
```
30 int verTope(const tPila *p, void *d, unsigned cantBytes)
31 {
32     unsigned tamInfo;
33
34     if(p->tope == TAM_PILA)
35         return 0;
36     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
37     memcpy(d, p->pila + p->tope + sizeof(unsigned),
38           minimo(cantBytes, tamInfo));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x

107 nuts("Ver tope de la pila:");

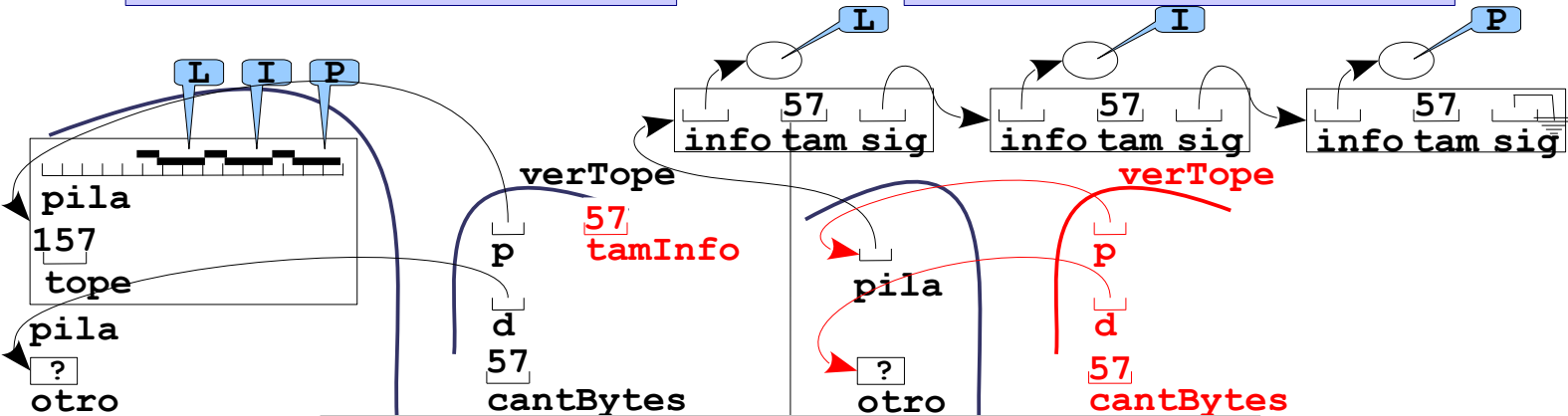
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
30 int verTope(const tPila *p, void *d, unsigned cantBytes)
31 {
32     unsigned tamInfo;
33
34     if(p->tope == TAM_PILA)
35         return 0;
36     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
37     memcpy(d, 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 x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x

107 nuts("Ver tope de la pila:");

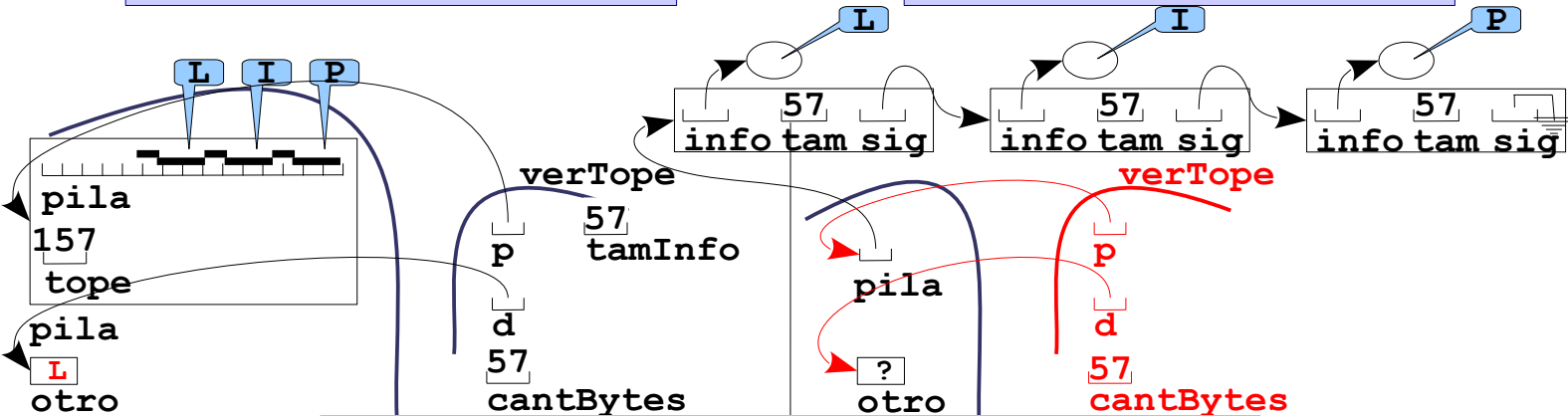
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
30 int verTope(const tPila *p, void *d, unsigned cantBytes)
31 {
32     unsigned tamInfo;
33
34     if(p->tope == TAM_PILA)
35         return 0;
36     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
37     memcpy(d, 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



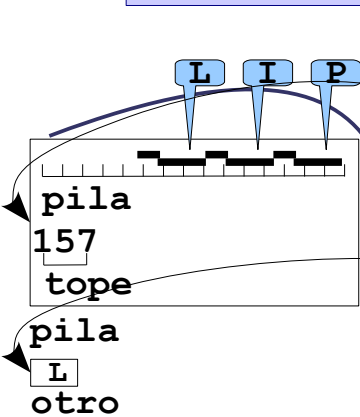
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x  
107 nuts("Ver tope de la pila:");

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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 }
```

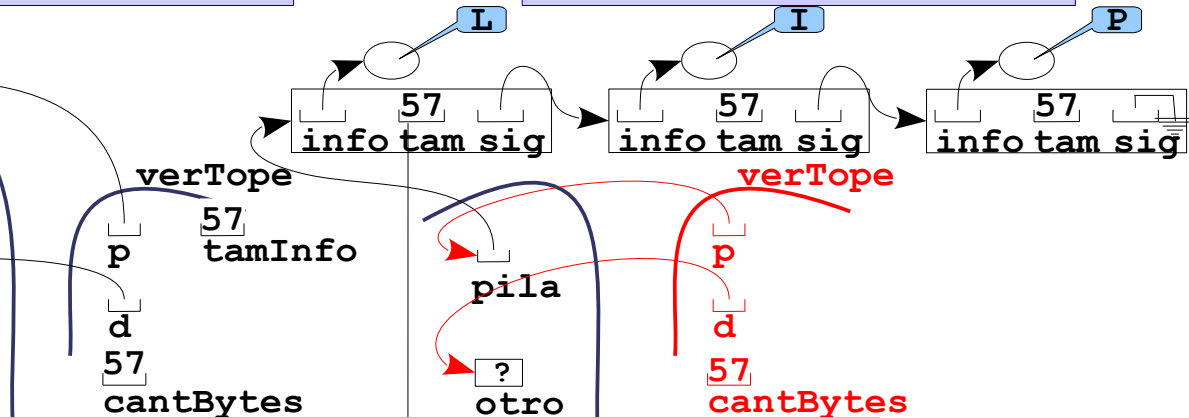
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



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

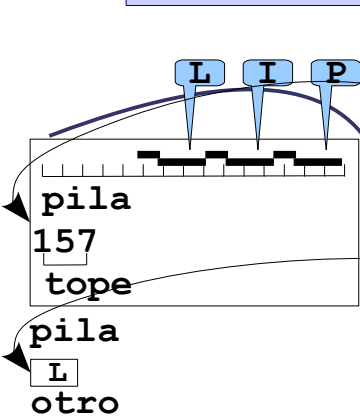
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
30 int verTope(const tPila *p, void *d, unsigned cantBytes)
31 {
32     unsigned tamInfo;
33
34     if(p->tope == 0)
35         return 0;
36     memcpy(&tamInfo, p->tope, sizeof(tamInfo));
37     memcpy(d, p->info,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

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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 }
```

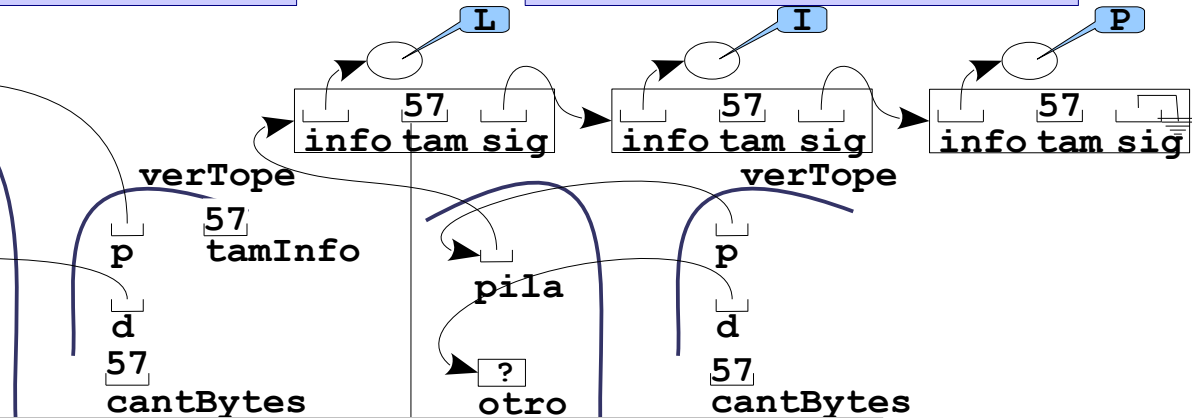
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



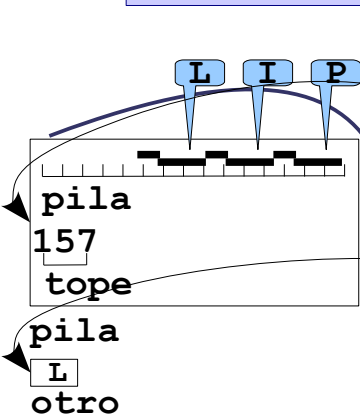
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x  
107 nuts("Ver tope de la pila:");

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
30 int verTope(const tPila *p, void *d, unsigned cantBytes)
31 {
32     unsigned tamInfo;
33
34     if(p->tope == NULL)
35         return 0;
36     memcpy(&tamInfo, p->tope->sig, sizeof(tamInfo));
37     memcpy(d, p->tope->info, minimo(cantBytes, tamInfo));
38     return 1;
39 }
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 }
```

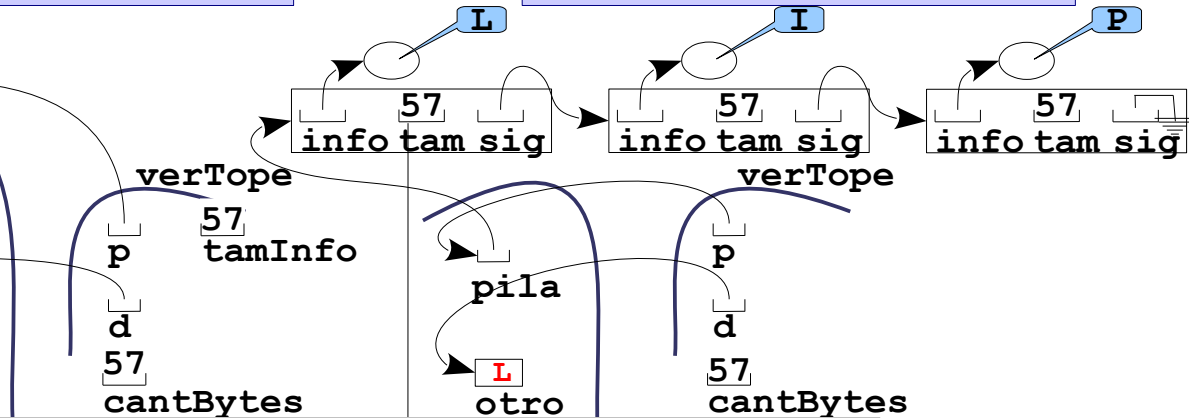
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x  
107 nuts("Ver tope de la pila:");

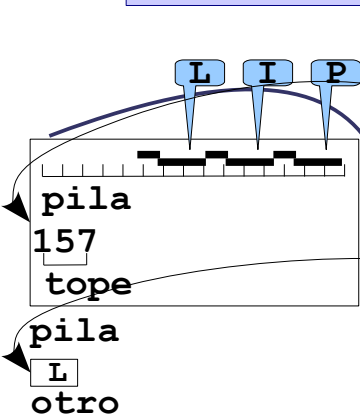
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
30 int verTope(const tPila *p, void *d, unsigned cantBytes)
31 {
32     unsigned tamInfo;
33
34     if(p->tope == NULL)
35         return 0;
36     memcpy(&tamInfo, p->tope->tam, sizeof(tamInfo));
37     memcpy(d, p->tope->info, minimo(cantBytes, tamInfo));
38     return 1;
39 }
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 }
```



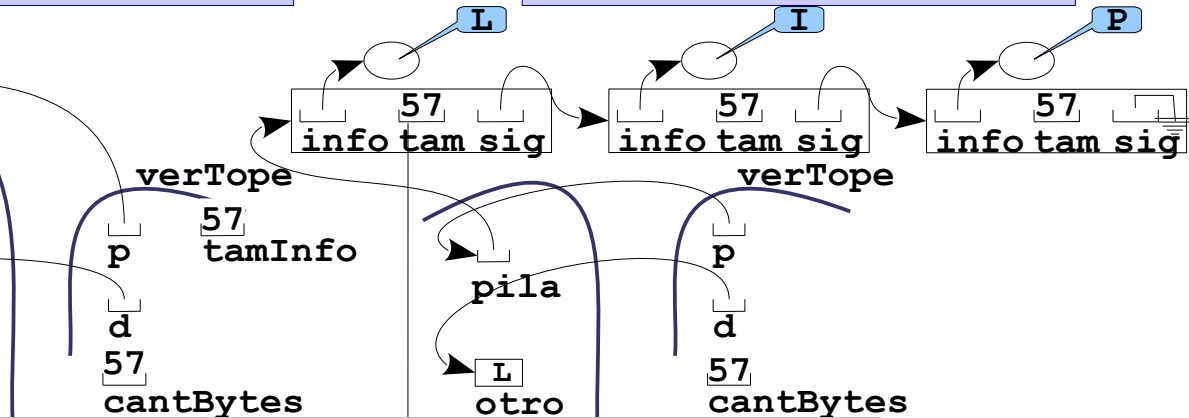
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



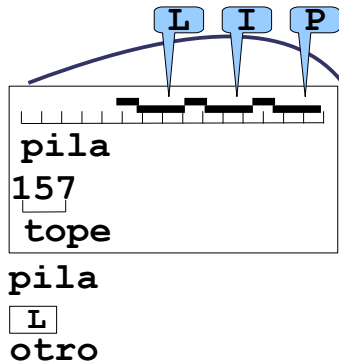
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x  
107 nuts("Ver tope de la pila:");

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

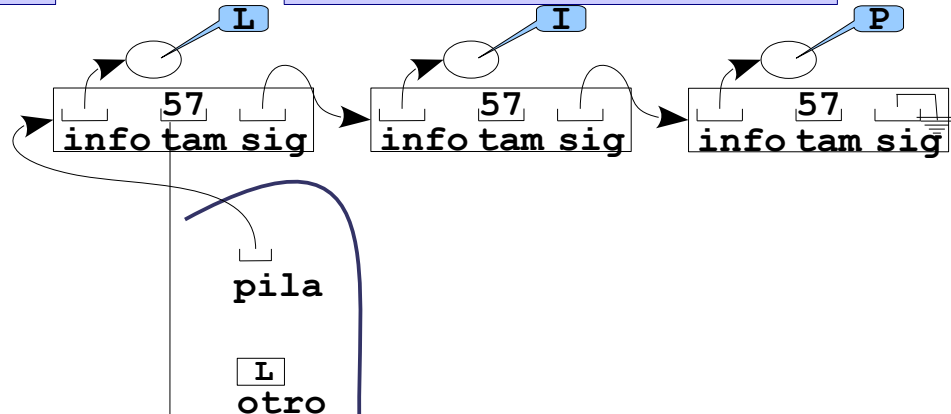
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



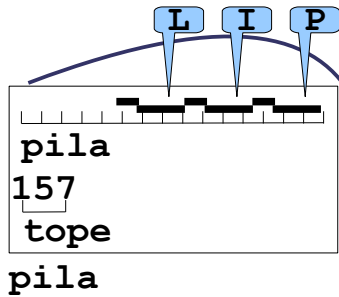
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
107 puts("Ver tope de la pila:");
108 if(verTope(&pila, &otro, sizeof(otro)))
109     mostrarProducto(&otro);
110 else
111     puts("La pila estaba vacia.");
```



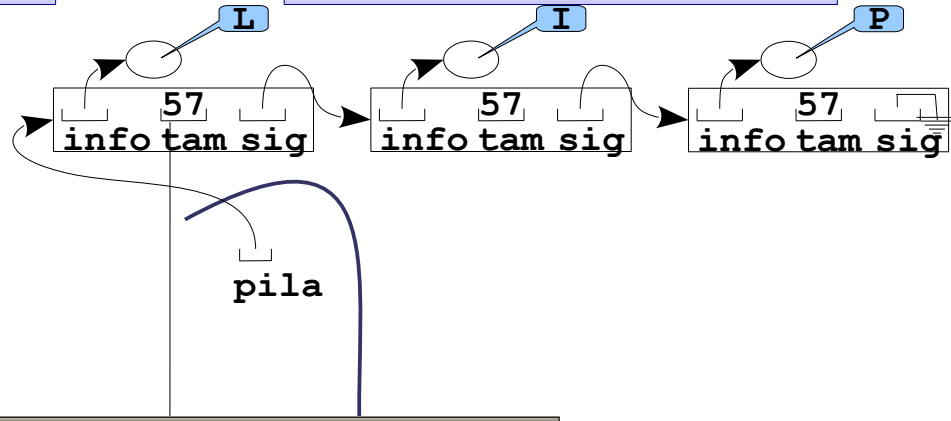
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



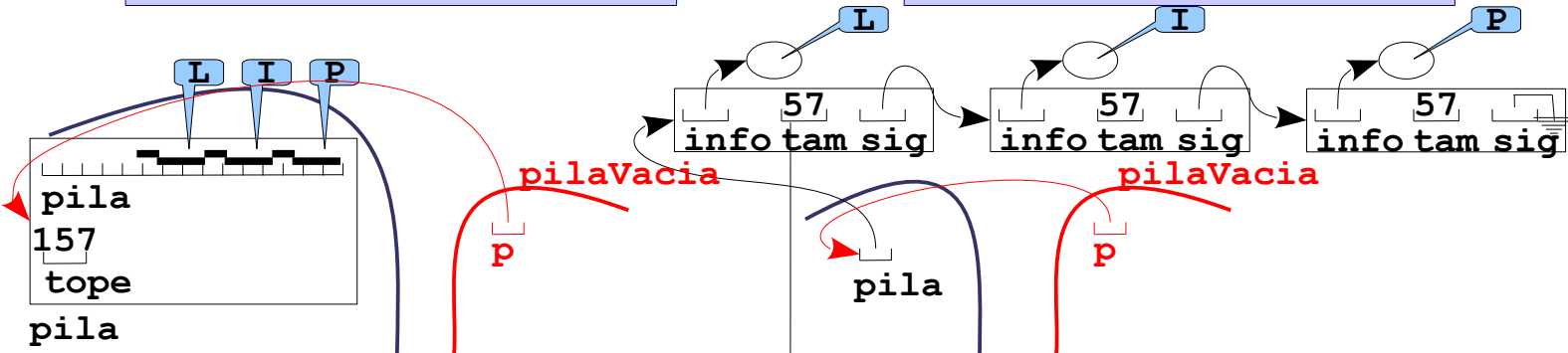
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacia(&pila))
120     puts("La pila esta vacía");
121 else
122     {
123         puts("Vaciando la pila");
124         vaciarPila(&pila);
125     }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



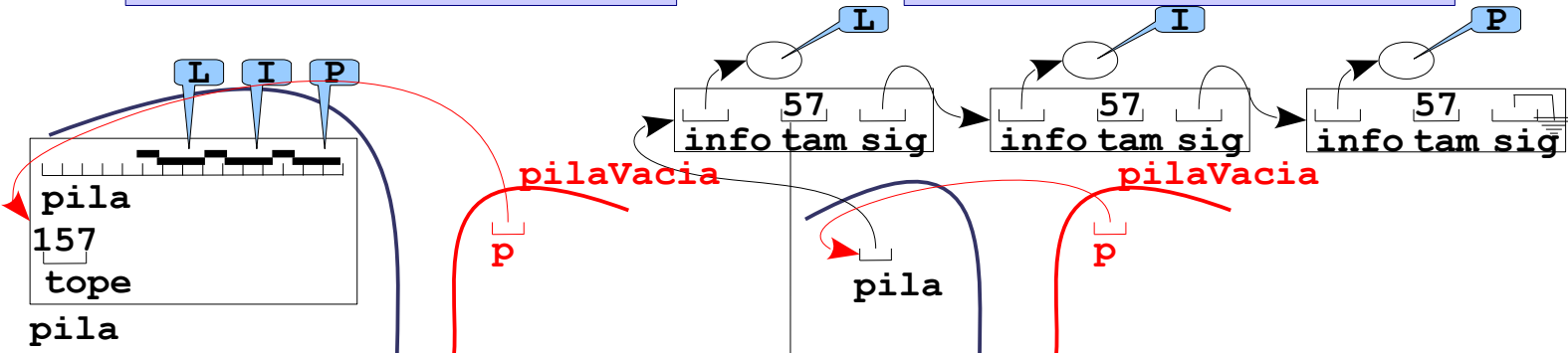
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
120     puts("La pila está vacía");
121 else
122     {
123         puts("Vaciando la pila");
124         vaciarPila(&pila);
125     }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacia(&pila))
120     puts("La pila esta vacía");
121 else
122     {
123         puts("Vaciando la pila");
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h
42 int pilaVacia(const tPila *p)
43 {
44     return p->tope == TAM_PILA;
45 }
```

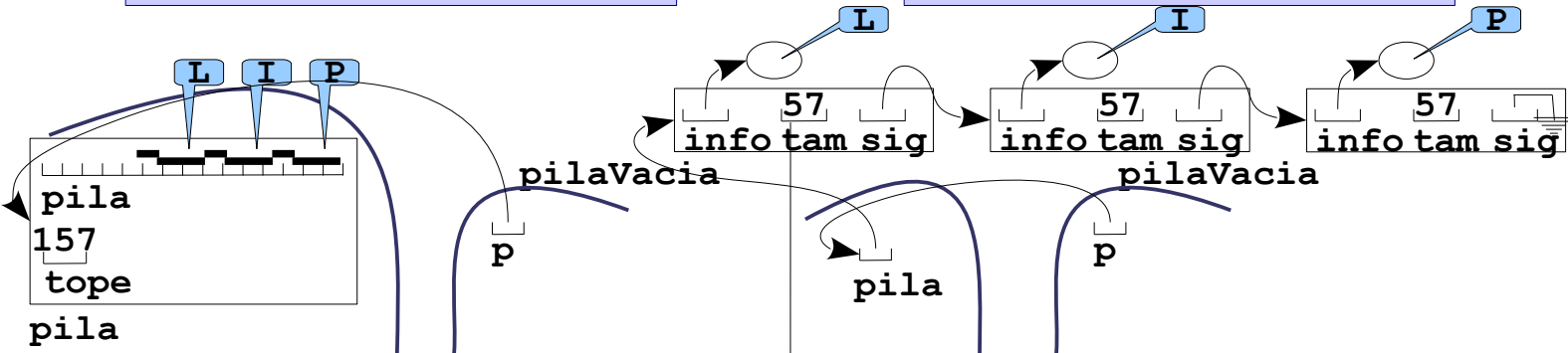
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h
49 int pilaVacia(const tPila *p)
50 {
51     return *p == NULL;
52 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacia(&pila))
120     puts("La pila esta vacía");
121 else
122     {
123         puts("Vaciando la pila");
```

aciarPil

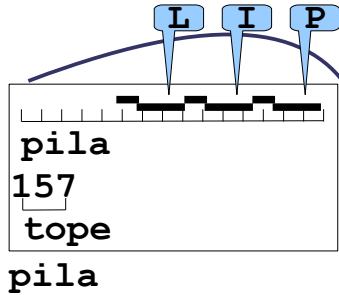
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h
42 int pilaVacia(const tPila *p)
43 {
44     return p->tope == TAM_PILA;
45 }
```

```
main.c x main.h x productos.c x productos.h x pila.c x pila.h
49 int pilaVacia(const tPila *p)
50 {
51     return *p == NULL;
52 }
```

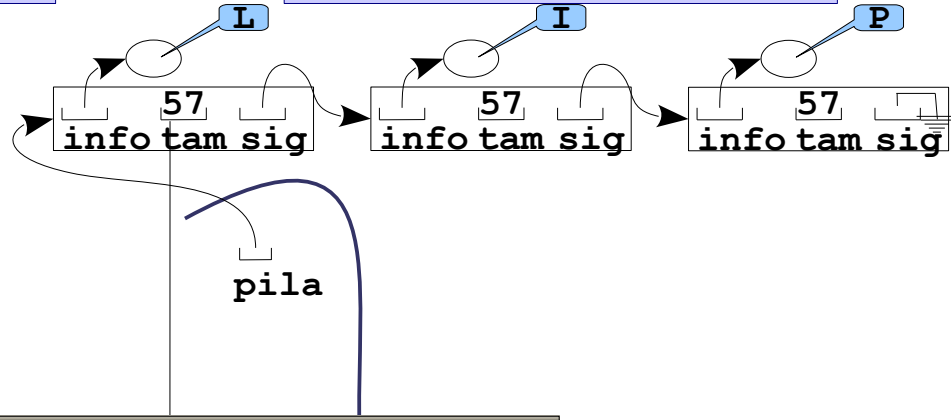
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



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

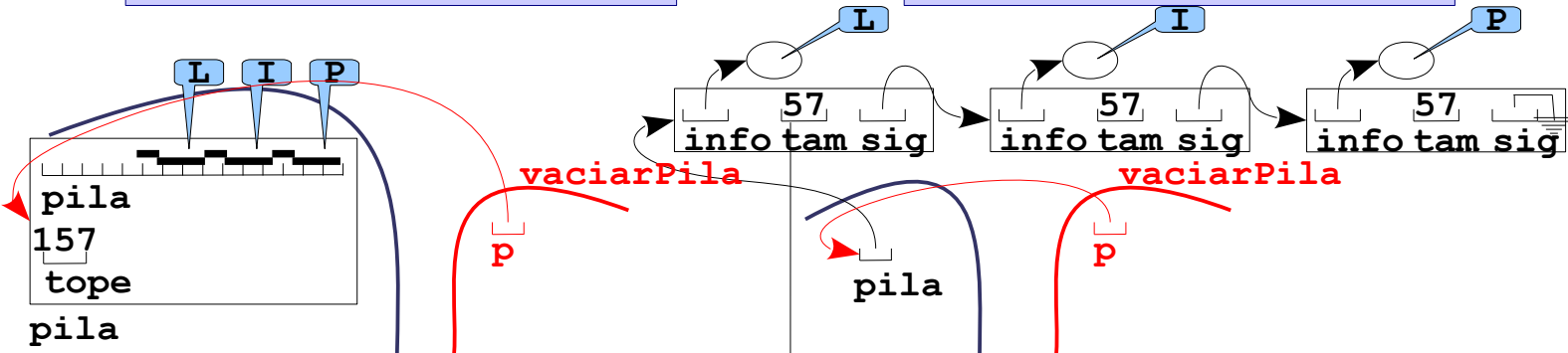


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



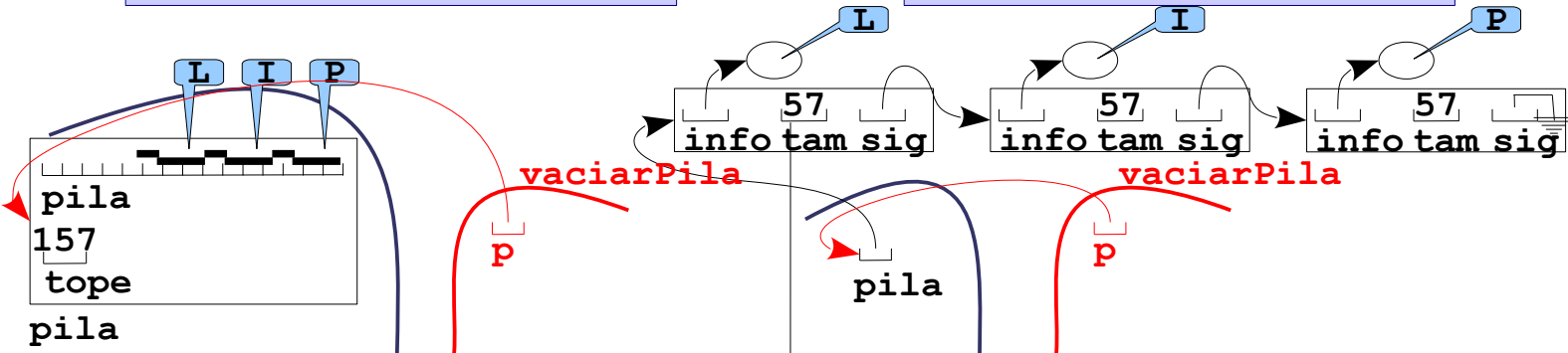
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119     if(pilaVacia(&pila))
120     ... puts("La pila esta vacía");
121     else
122     {
123         puts("Vaciando la pila");
124         vaciarPila(&pila);
125     }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía");
else
    puts("Vacianando la pila");
    vaciarPila(&pila);
```

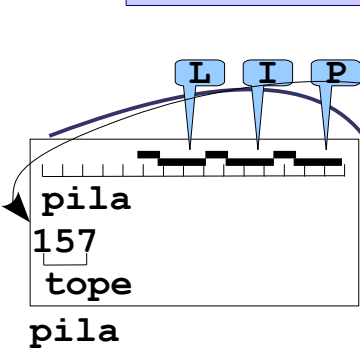
```
main.c x main.h x productos.c x productos.h x pila.c x
60 void vaciarPila(tPila *p)
61 {
62     p->tope = TAM_PILA;
63 }
```

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

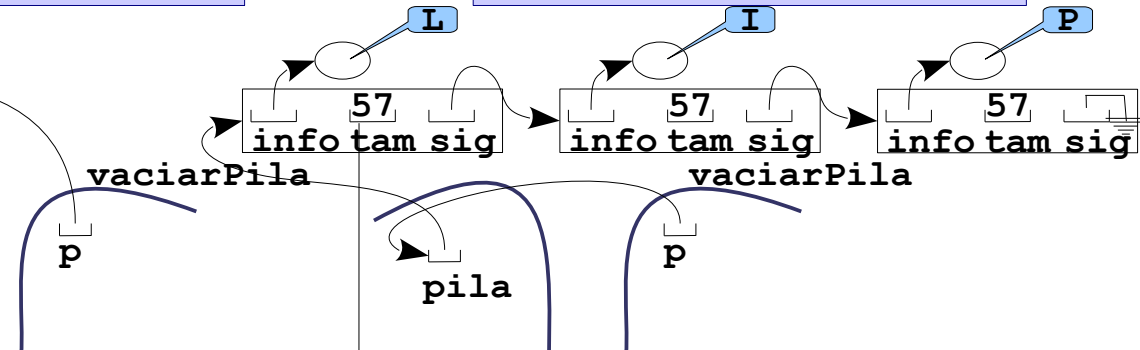
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía.")
else
    puts("Vacianando la pila.")
    vaciarPila(&pila)
```

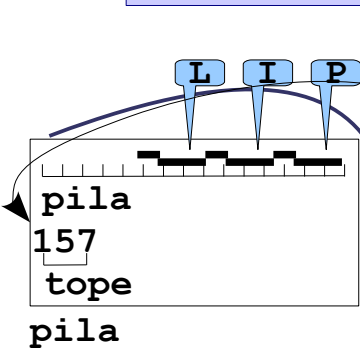
```
main.c x main.h x productos.c x productos.h x pila.c x
60 void vaciarPila(tPila *p)
61 {
62     p->tope = TAM_PILA;
63 }
```

```
main.c x main.h x productos.c x productos.h x pila.c x
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 }
```

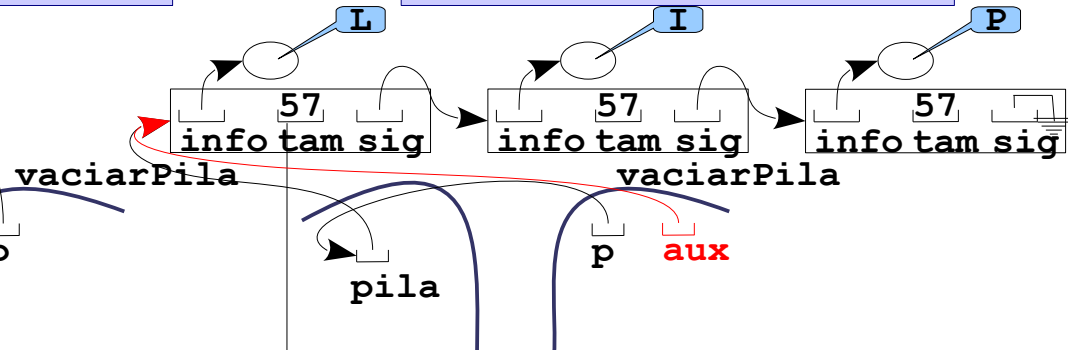
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacia(&pila))
    puts("La pila está vacía.");
else
    puts("Vacianando la pila.");
    vaciarPila(&pila);
```

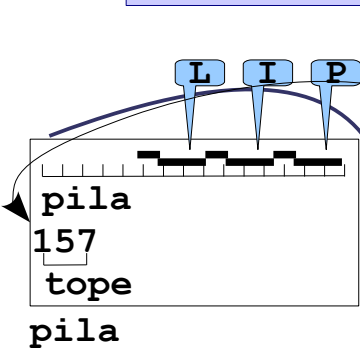
```
main.c x main.h x productos.c x productos.h x pila.c x
60 void vaciarPila(tPila *p)
61 {
62     p->tope = TAM_PILA;
63 }
```

```
main.c x main.h x productos.c x productos.h x pila.c x
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 }
```

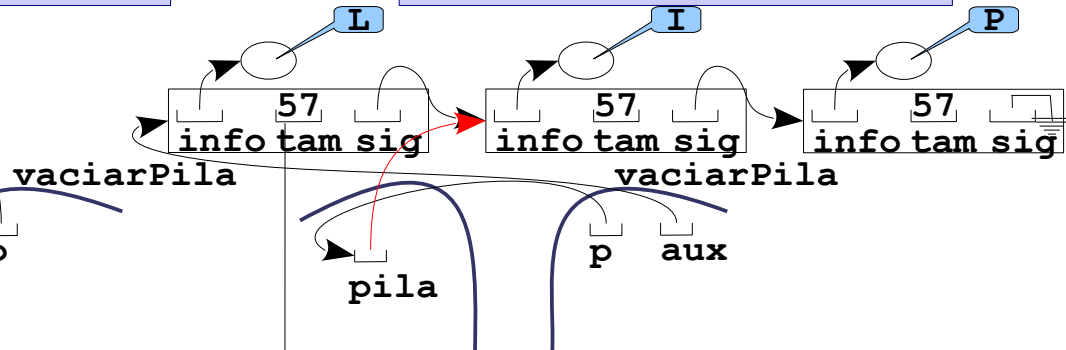
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía.");
else
    puts("Vacianando la pila.");
    vaciarPila(&pila);
```

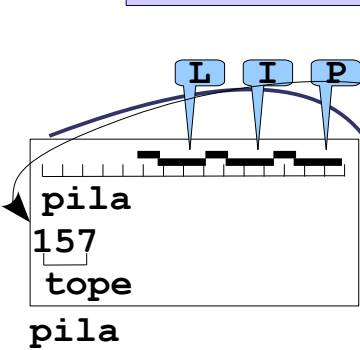
```
main.c x main.h x productos.c x productos.h x pila.c x
60 void vaciarPila(tPila *p)
61 {
62     p->tope = TAM_PILA;
63 }
```

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

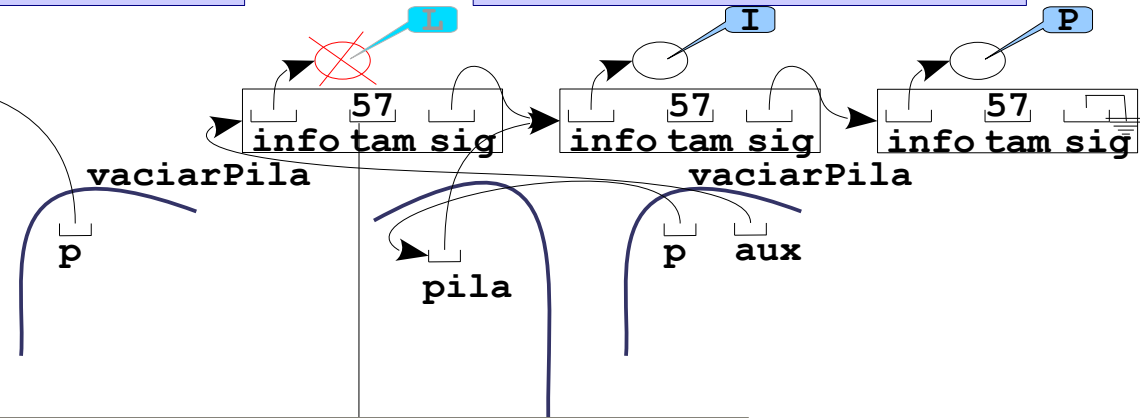
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía.");
else
    puts("Vacianando la pila.");
    vaciarPila(&pila);
```

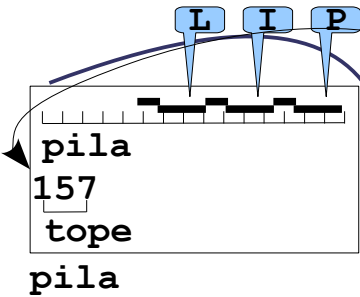
```
main.c x main.h x productos.c x productos.h x pila.c x
60 void vaciarPila(tPila *p)
61 {
62     p->tope = TAM_PILA;
63 }
```

```
main.c x main.h x productos.c x productos.h x pila.c x
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 }
```

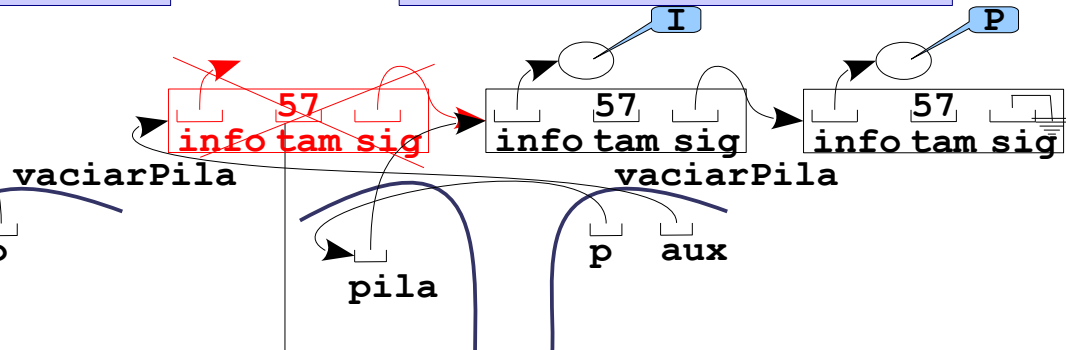
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacia(&pila))
    puts("La pila está vacía.");
else
    puts("Vacianando la pila.");
    vaciarPila(&pila);
```

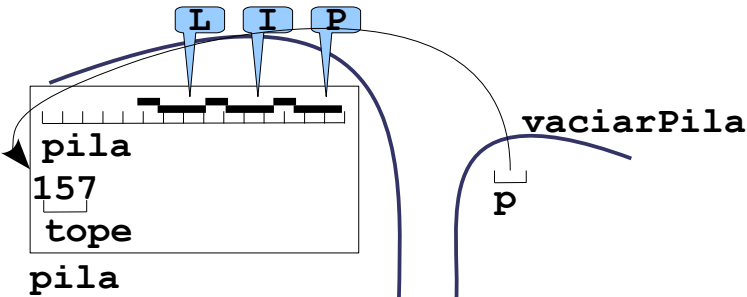
```
main.c x main.h x productos.c x productos.h x pila.c x
60 void vaciarPila(tPila *p)
61 {
62     p->tope = TAM_PILA;
63 }
```

```
main.c x main.h x productos.c x productos.h x pila.c x
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 }
```

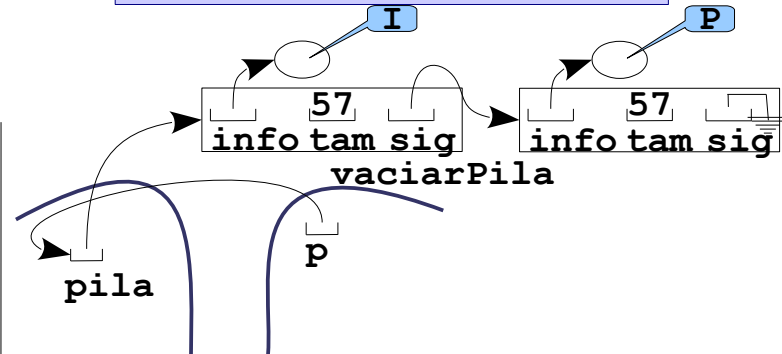
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía.");
else
    puts("Vacianando la pila.");
    vaciarPila(&pila);
```

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

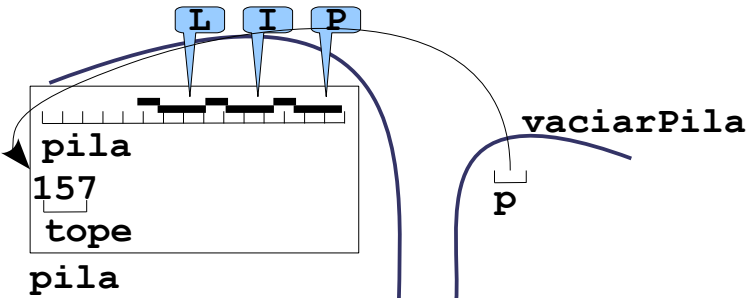
```
main.c x main.h x productos.c x productos.h x pila.c x
67 void vaciarPila(tPila *p)
68 {
69     while(*p)
70     {
71         tNodo *aux = *p;
72
73         *p = aux->sig;
74         free(aux->info);
75         free(aux);
76     }
77 }
```



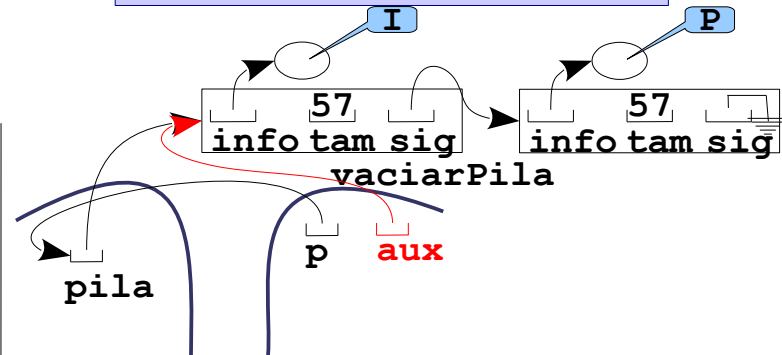
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía.");
else
    puts("Vacianando la pila.");
    vaciarPila(&pila);
```

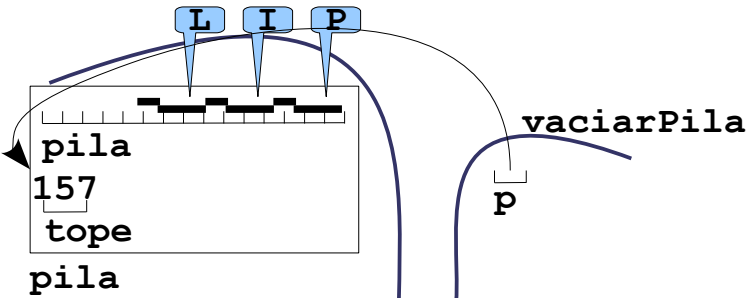
```
main.c x main.h x productos.c x productos.h x pila.c x
60 void vaciarPila(tPila *p)
61 {
62     p->tope = TAM_PILA;
63 }
```

```
main.c x main.h x productos.c x productos.h x pila.c x
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 }
```

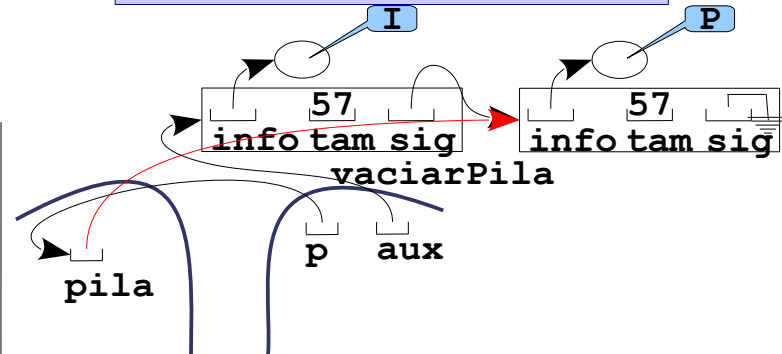
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía.");
else
    puts("Vacianando la pila.");
    vaciarPila(&pila);
```

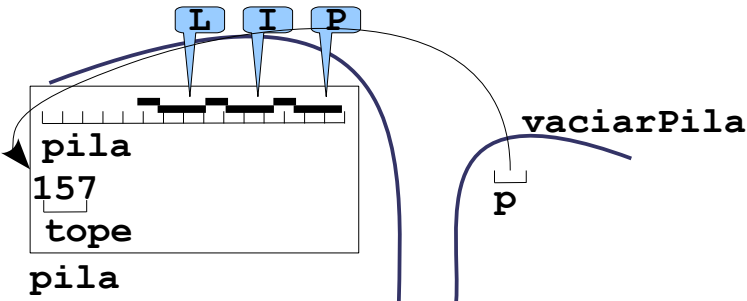
```
main.c x main.h x productos.c x productos.h x pila.c x
60 void vaciarPila(tPila *p)
61 {
62     p->tope = TAM_PILA;
63 }
```

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

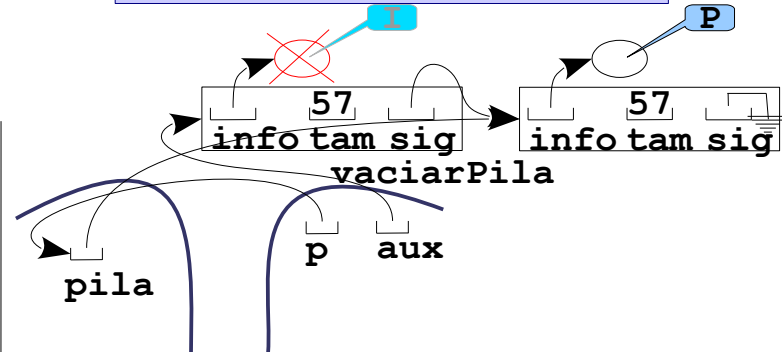
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía.");
else
    puts("Vacianando la pila.");
    vaciarPila(&pila);
```

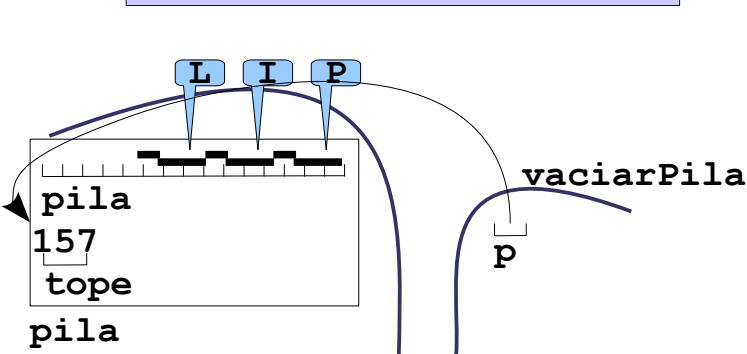
```
main.c x main.h x productos.c x productos.h x pila.c x
60 void vaciarPila(tPila *p)
61 {
62     p->tope = TAM_PILA;
63 }
```

```
main.c x main.h x productos.c x productos.h x pila.c x
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 }
```

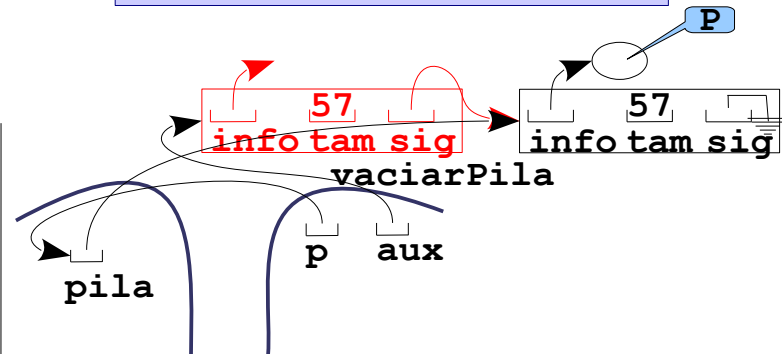
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía.");
else
    puts("Vacianando la pila.");
    vaciarPila(&pila);
```

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

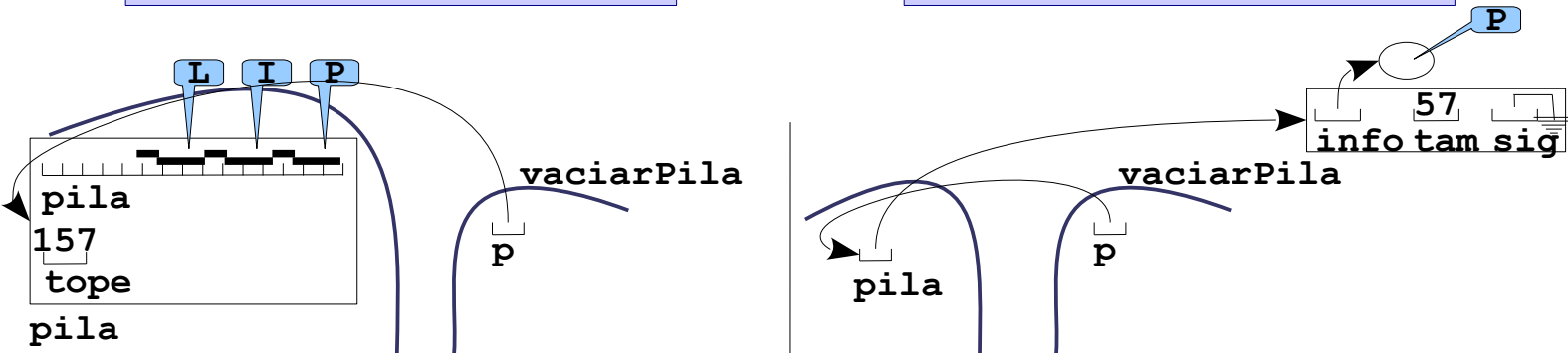
```
main.c x main.h x productos.c x productos.h x pila.c x
67 void vaciarPila(tPila *p)
68 {
69     while(*p)
70     {
71         tNodo *aux = *p;
72
73         *p = aux->sig;
74         free(aux->info);
75         free(aux);
76     }
77 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía");
else
    puts("Vacianando la pila");
    vaciarPila(&pila);
```

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

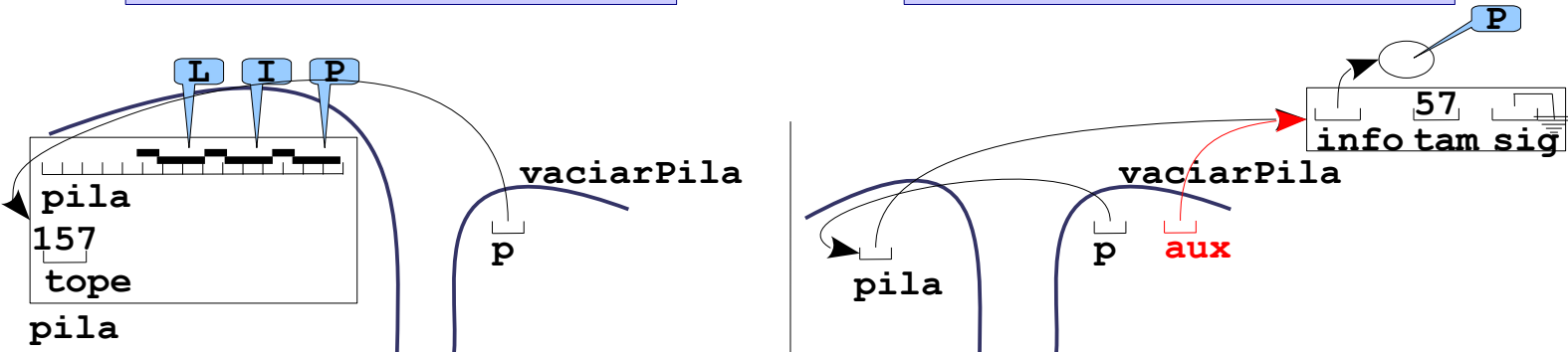
```
main.c x main.h x productos.c x productos.h x pila.c x
67 void vaciarPila(tPila *p)
68 {
69     while(*p)
70     {
71         tNodo *aux = *p;
72
73         *p = aux->sig;
74         free(aux->info);
75         free(aux);
76     }
77 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía.");
else
    puts("Vacianando la pila.");
    vaciarPila(&pila);
```

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

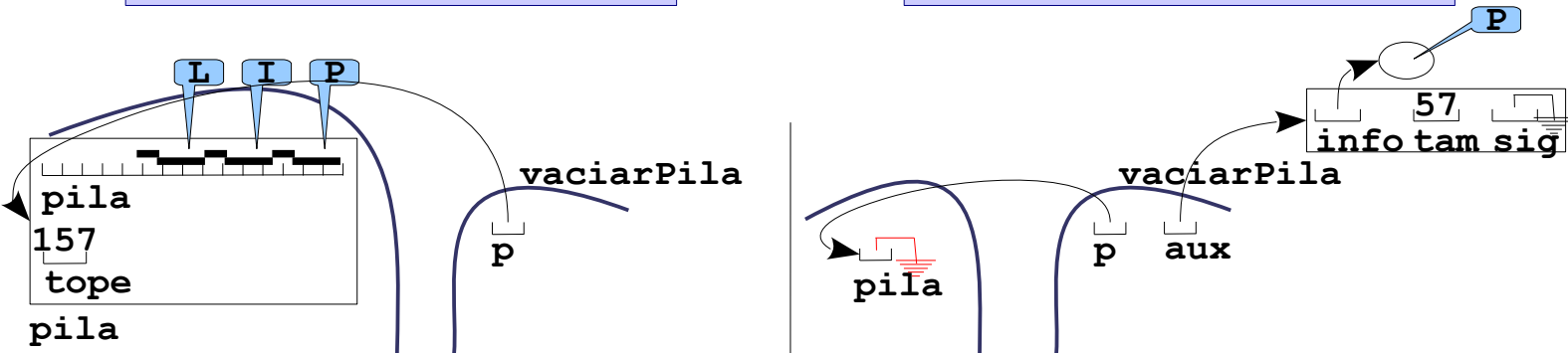
```
main.c x main.h x productos.c x productos.h x pila.c x
67 void vaciarPila(tPila *p)
68 {
69     while(*p)
70     {
71         tNodo *aux = *p;
72
73         *p = aux->sig;
74         free(aux->info);
75         free(aux);
76     }
77 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía.");
else
    puts("Vacianando la pila.");
    vaciarPila(&pila);
```

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

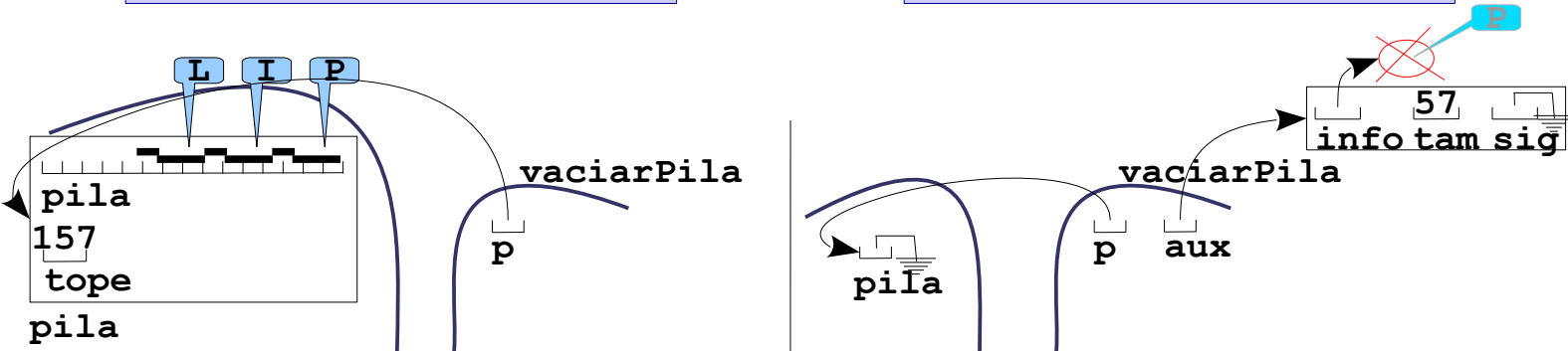
```
main.c x main.h x productos.c x productos.h x pila.c x
67 void vaciarPila(tPila *p)
68 {
69     while(*p)
70     {
71         tNodo *aux = *p;
72         *p = aux->sig;
73         free(aux->info);
74         free(aux);
75     }
76 }
77 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
puts("La pila está vacía")
else
puts("Vacianando la pila")
vaciarPila(&pila)
```

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

```
main.c x main.h x productos.c x productos.h x pila.c x
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 }
```

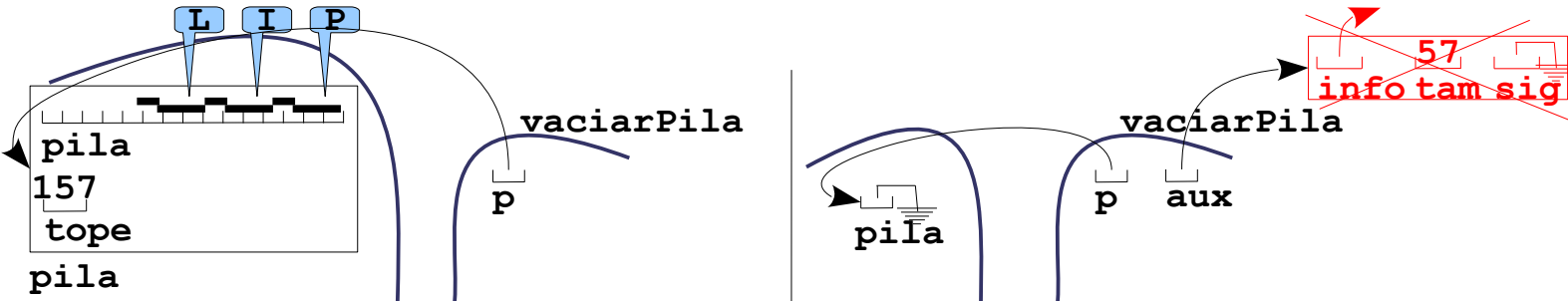


# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía");
else
    puts("Vacianando la pila");
    vaciarPila(&pila);
```

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

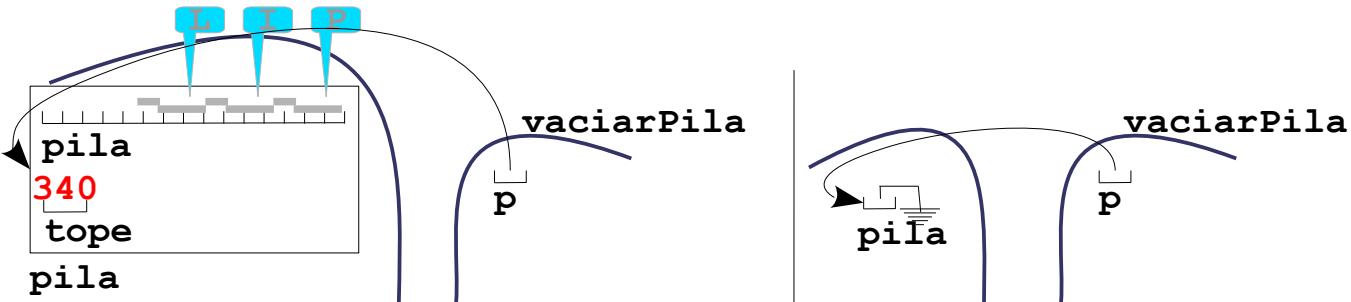
```
main.c x main.h x productos.c x productos.h x pila.c x
67 void vaciarPila(tPila *p)
68 {
69     while(*p)
70     {
71         tNodo *aux = *p;
72
73         *p = aux->sig;
74         free(aux->info);
75         free(aux);
76     }
77 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacía(&pila))
    puts("La pila está vacía");
else
    puts("Vacianando la pila");
    vaciarPila(&pila);
```

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

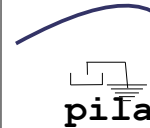
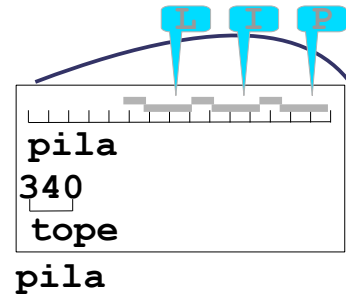
```
main.c x main.h x productos.c x productos.h x pila.c x
67 void vaciarPila(tPila *p)
68 {
69     while(*p)
70     {
71         tNode *aux = *p;
72         *p = aux->sig;
73         free(aux->info);
74         free(aux);
75     }
76 }
77 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119 if(pilaVacia(&pila))
    puts("La pila está vacía");
else
    puts("Vacianando la pila");
    vaciarPila(&pila);
```

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

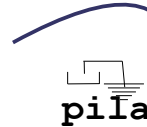
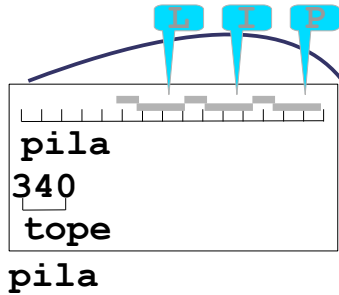
```
main.c x main.h x productos.c x productos.h x pila.c x
67 void vaciarPila(tPila *p)
68 {
69     while(*p)
70     {
71         tNodo *aux = *p;
72
73         *p = aux->sig;
74         free(aux->info);
75         free(aux);
76     }
77 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



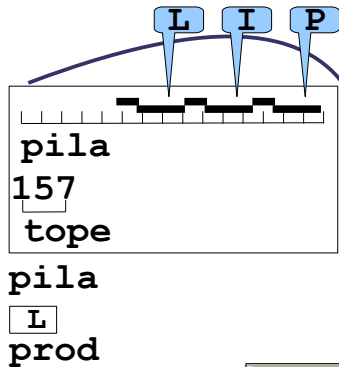
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x def
119      if(pilaVacía(&pila))
120          puts("La pila está vacía");
121      else
122      {
123          puts("Vaciando la pila");
124          vaciarPila(&pila);
125      }
```



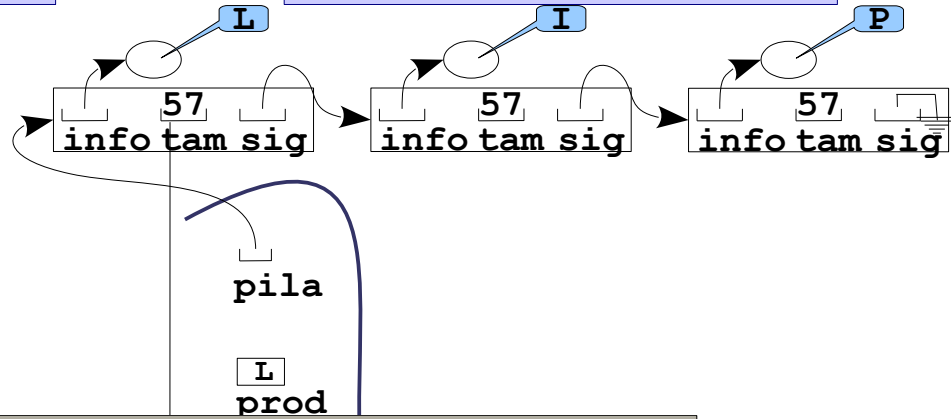
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



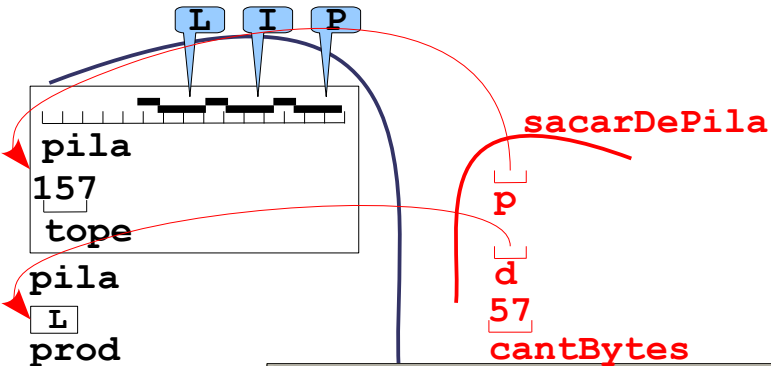
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd)))
116     mostrarProducto(&prod);
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



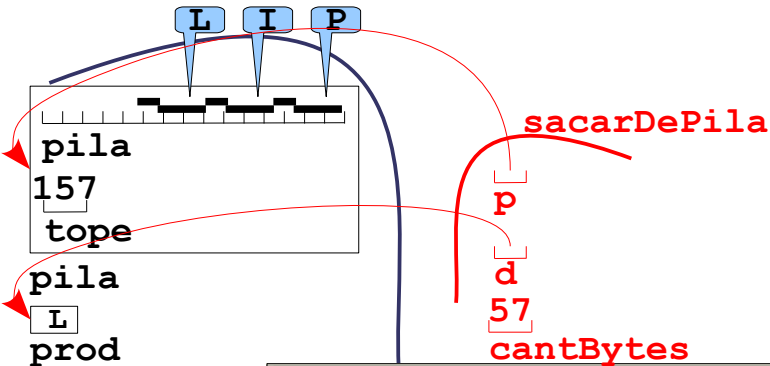
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd)))
116     mostrarProducto(&prod);
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
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 }
```

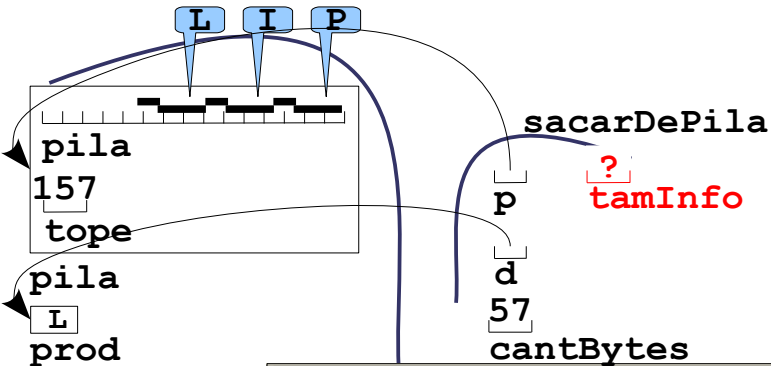


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



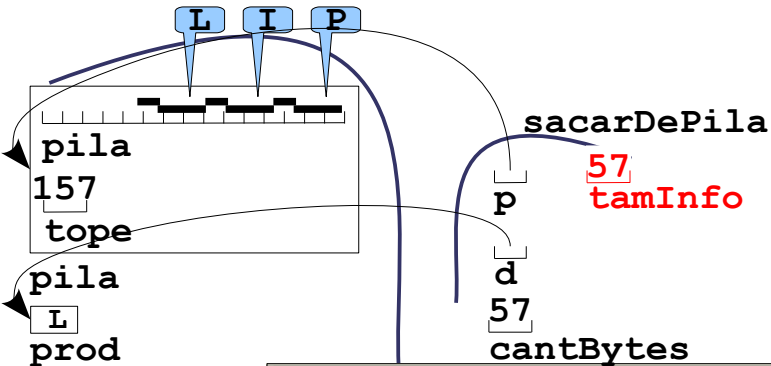
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x (tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



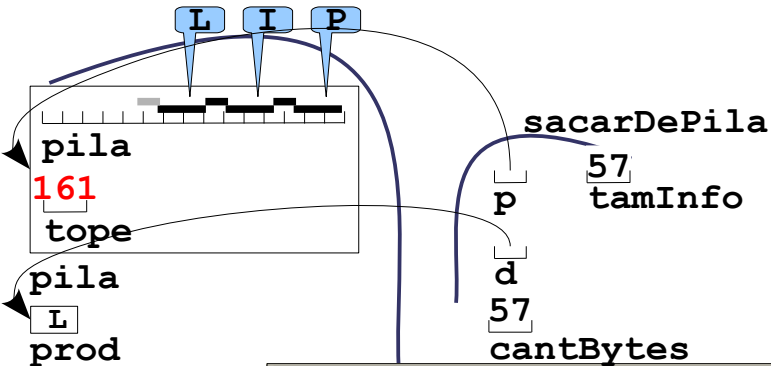
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



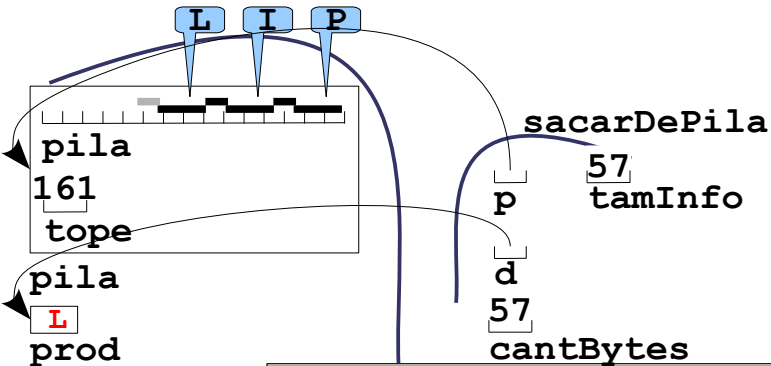
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



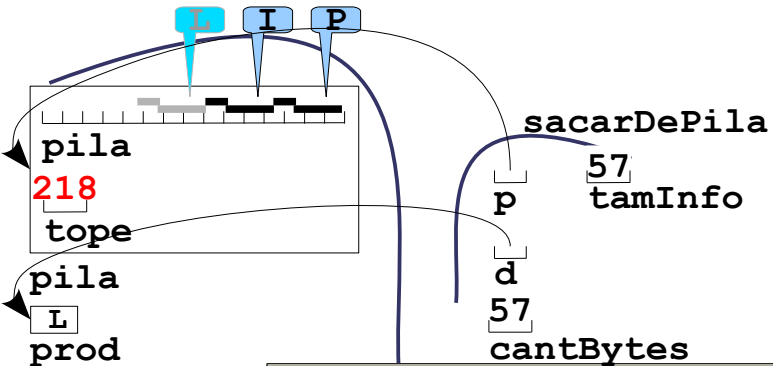
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



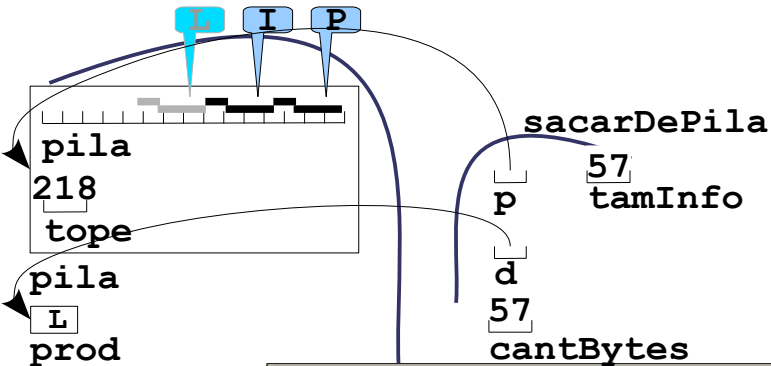
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
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 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

I I P



pila  
L  
prod

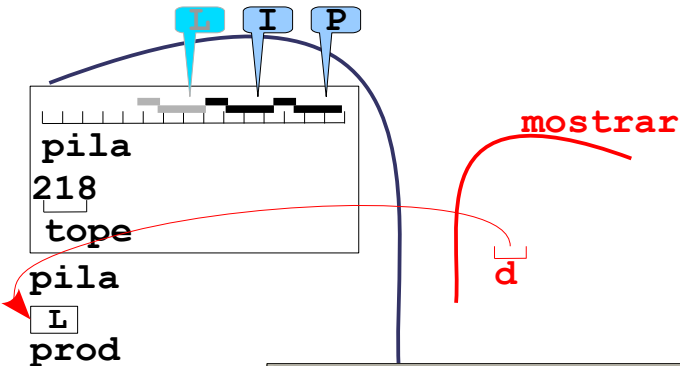
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd))) {
116     mostrarProducto(&prod);
117 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



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

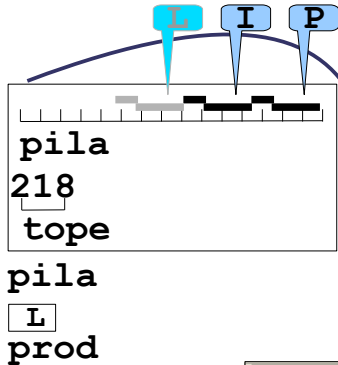


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



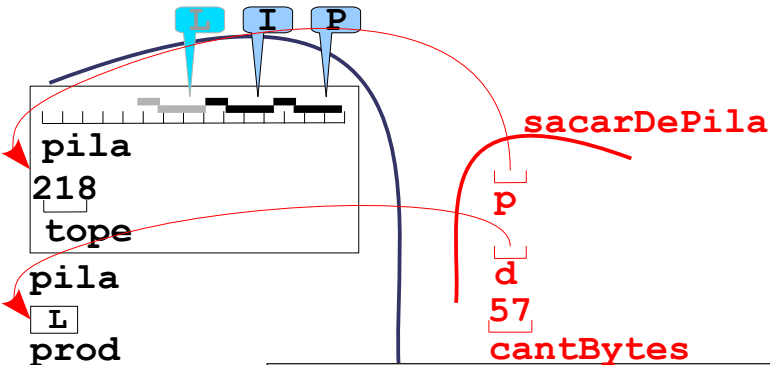
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd)))
116     mostrarProducto(&prod);
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



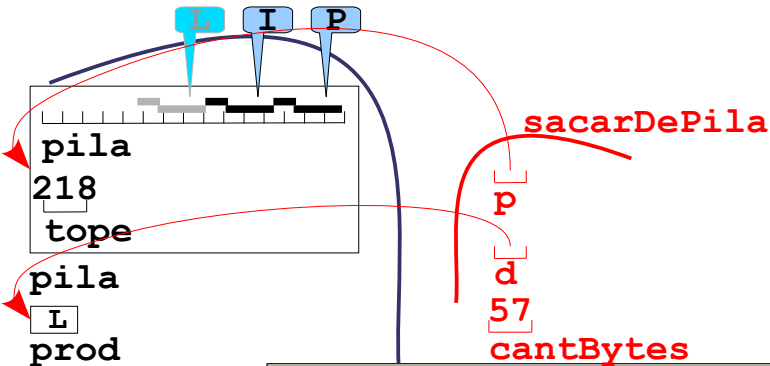
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd)))
116     mostrarProducto(&prod);
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



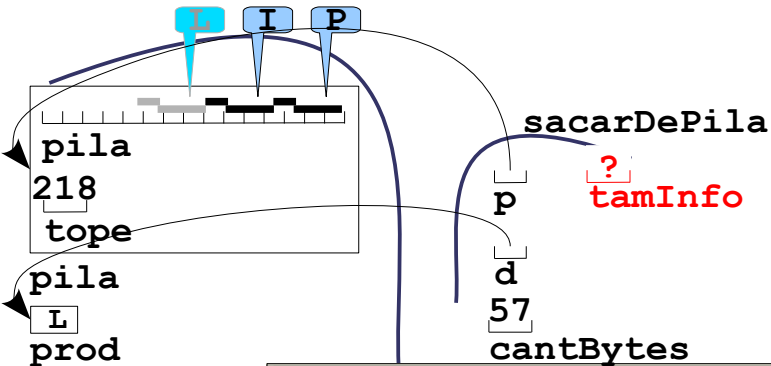
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



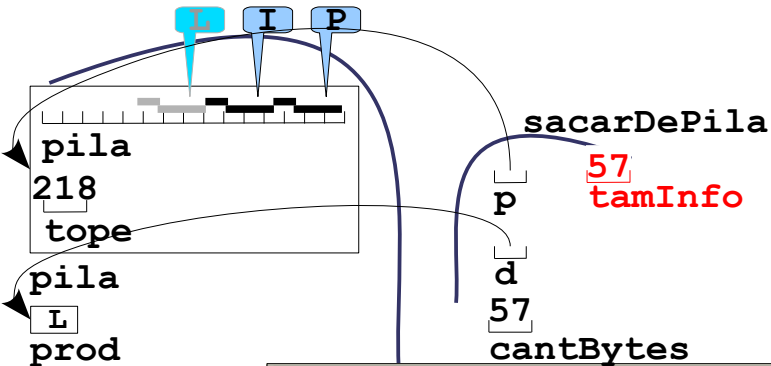
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



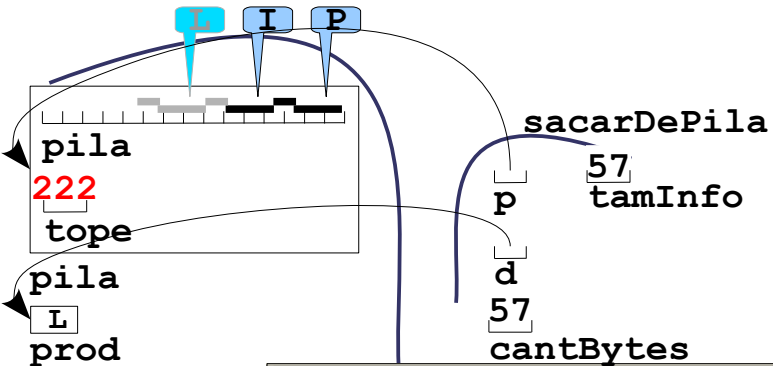
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



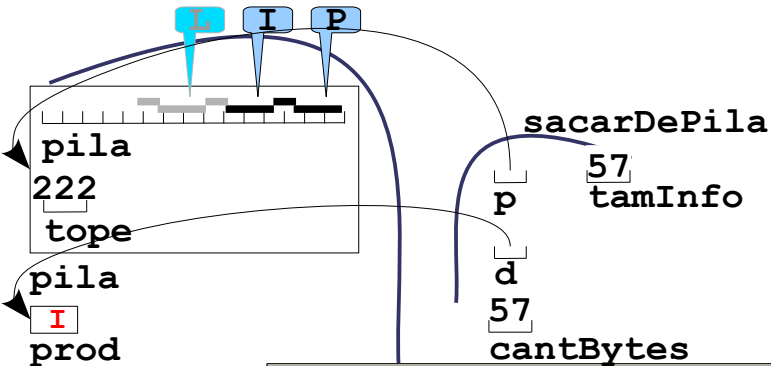
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



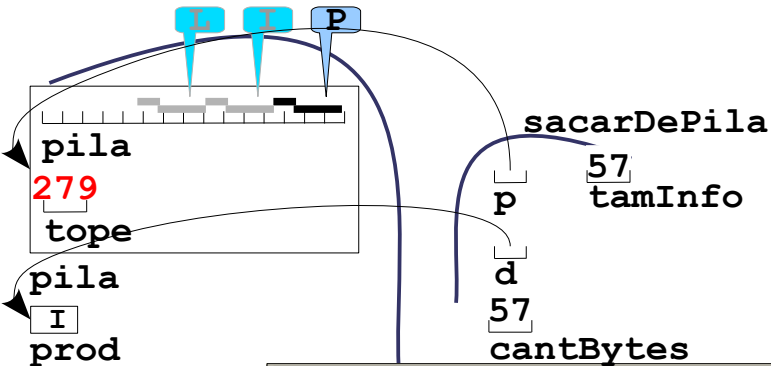
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
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 }
```

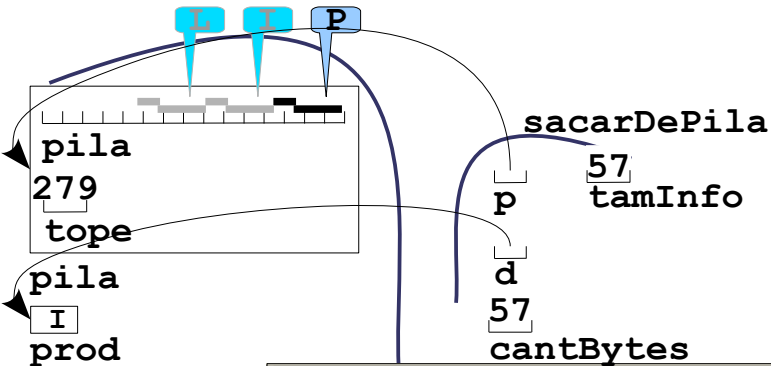


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



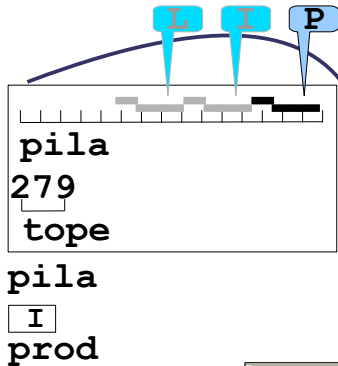
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54     p->tope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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



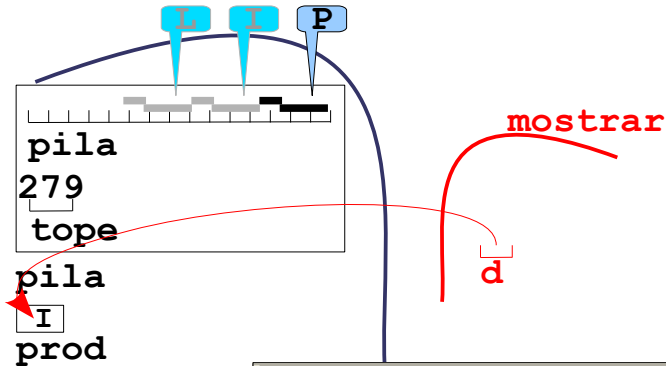
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd))) {
116     mostrarProducto(&prod);
117 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



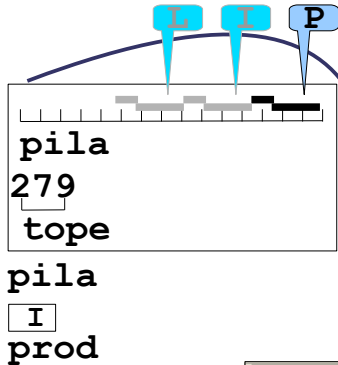
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd)))
116     mostrarProducto(&prod);
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



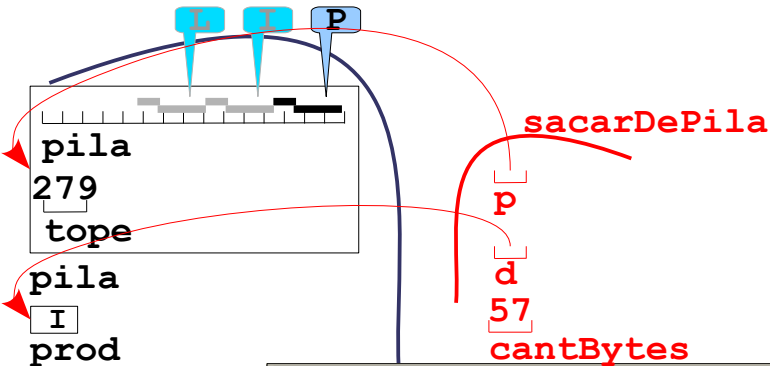
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd)))
116     mostrarProducto(&prod);
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



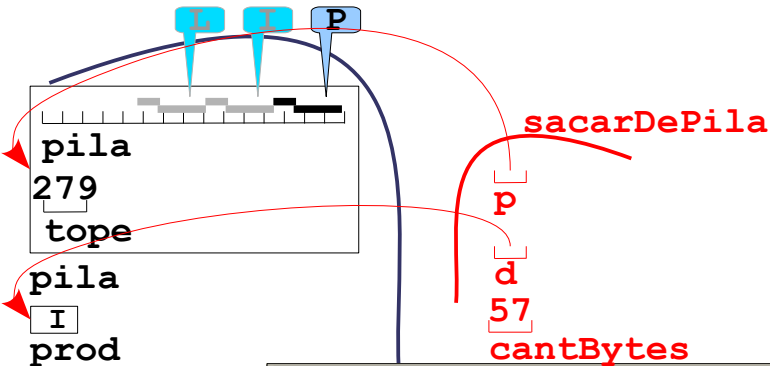
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd)))
116     mostrarProducto(&prod);
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



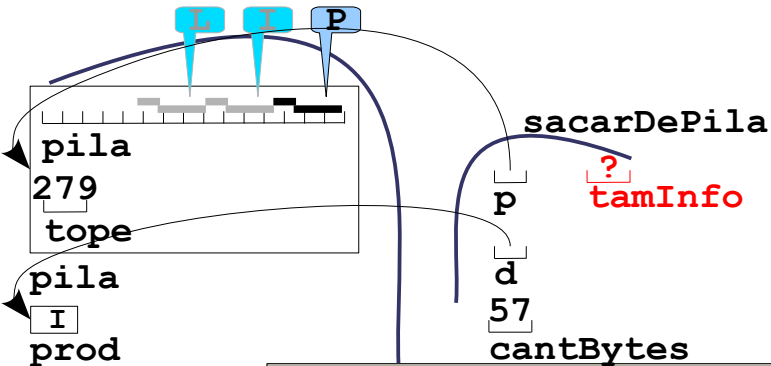
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



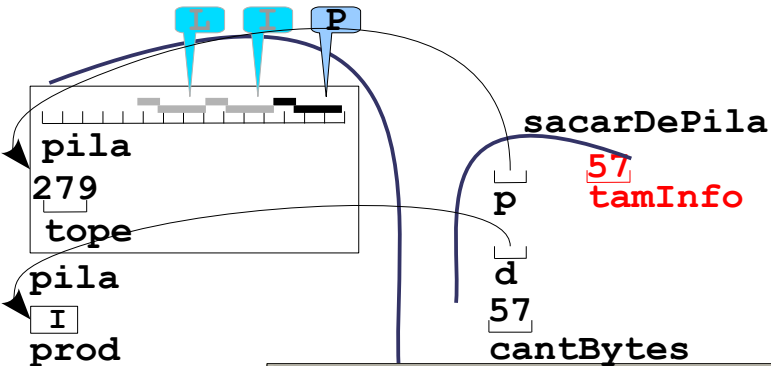
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x (tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x (tProd)))
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 }
```

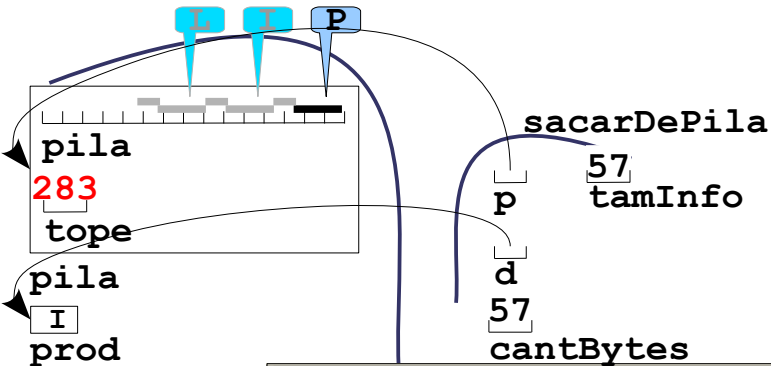


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



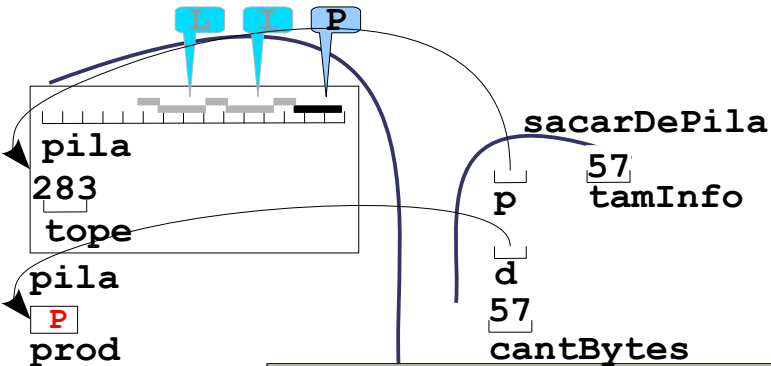
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x (tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



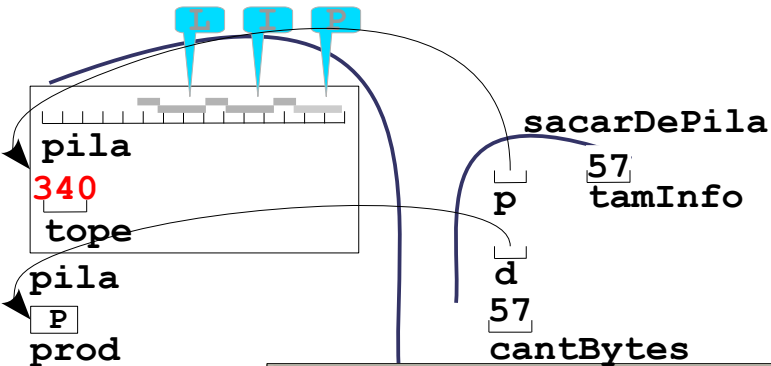
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x (tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



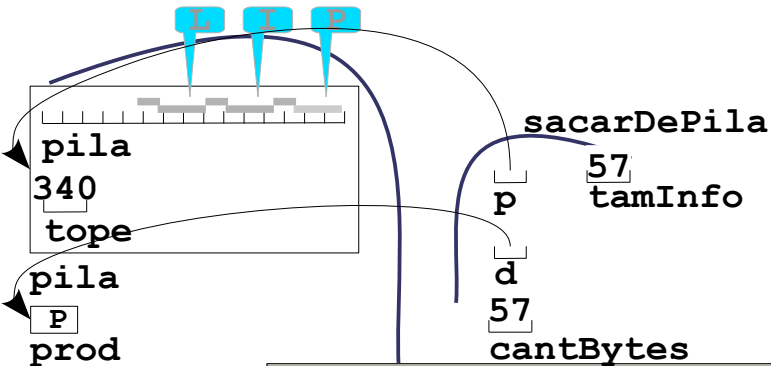
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x (tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x (tProd)))
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 }
```

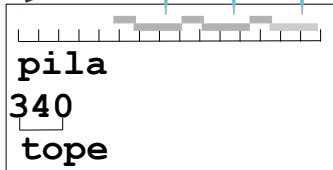
# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

I I P



pila  
P  
prod

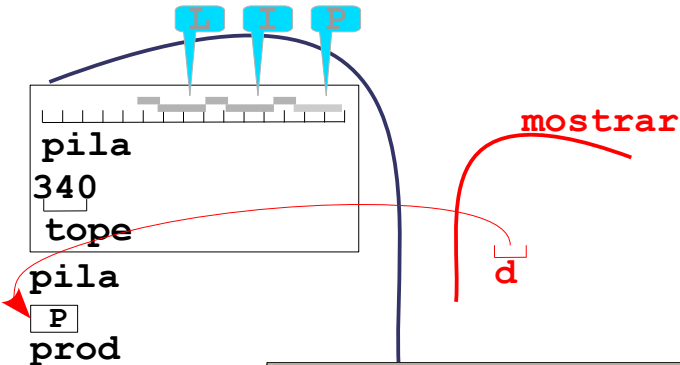
```
115 while(sacarDePila(&pila, &prod, sizeof(tProd))) {  
116     mostrarProducto(&prod);  
117 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



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

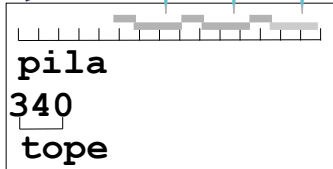
# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

I I P



pila  
P  
prod

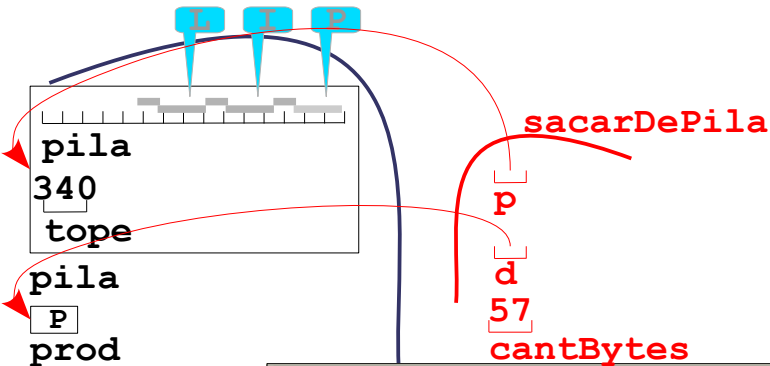
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd)))
116     mostrarProducto(&prod);
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



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

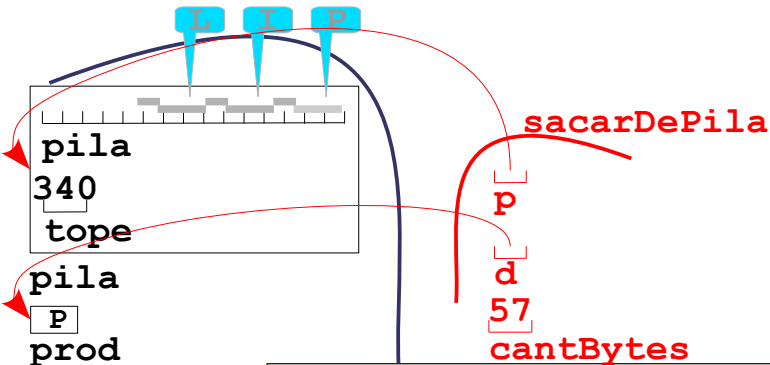


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



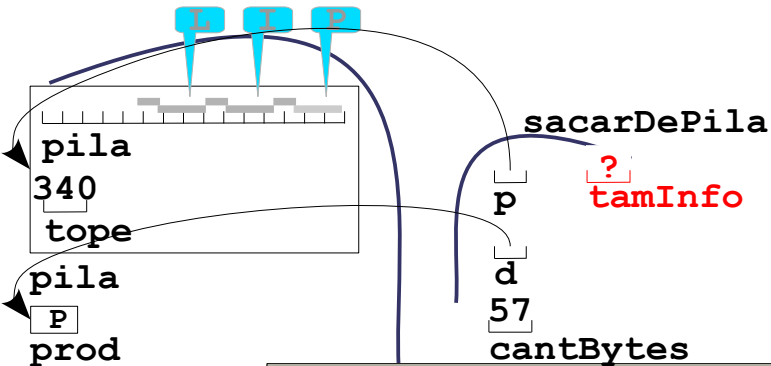
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



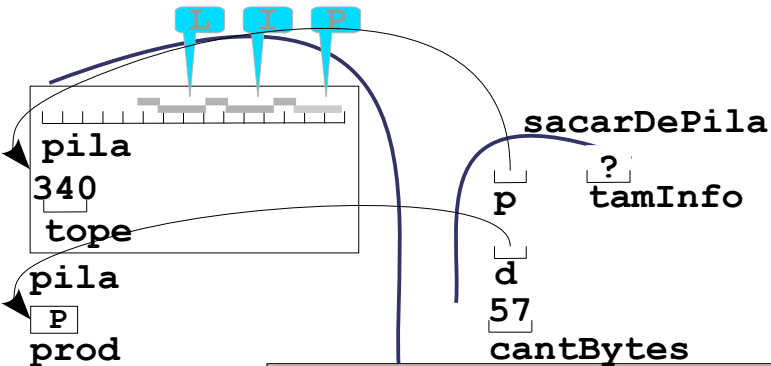
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x tProd)))
47 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
48 {
49     unsigned tamInfo;
50
51     if(p->tope == TAM_PILA)
52         return 0;
53     memcpy(&tamInfo, p->pila + p->tope, sizeof(unsigned));
54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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
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 }
```

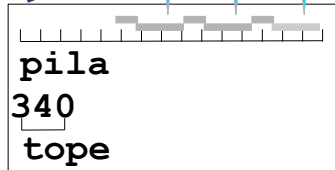
# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

I I P



pila  
P  
prod

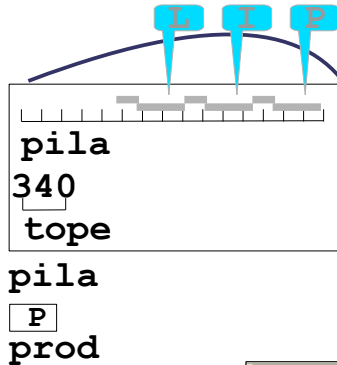
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd))) {
116     mostrarProducto(&prod);
}
```



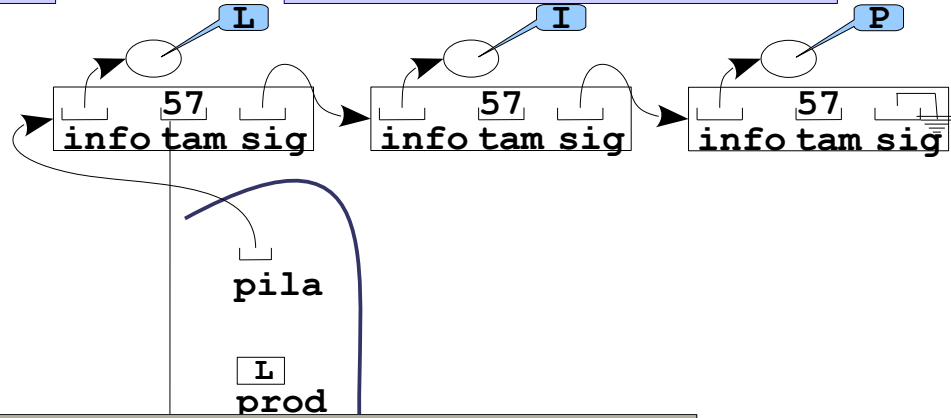
# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática



### Implementación dinámica



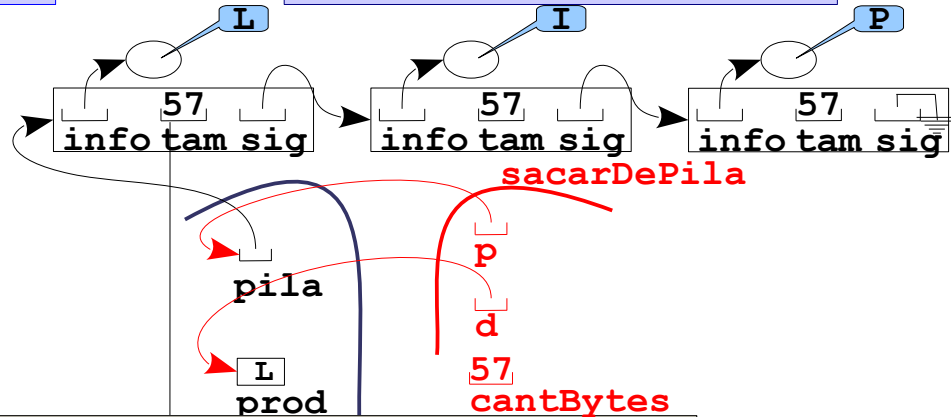
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd)))
116     mostrarProducto(&prod);
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



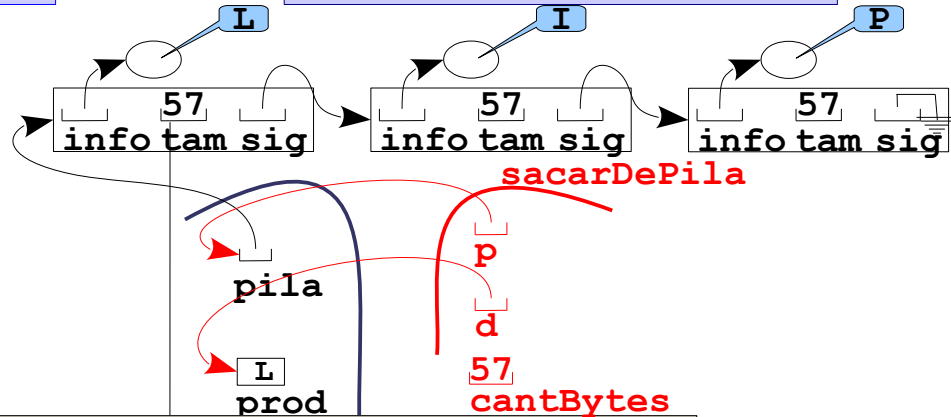
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115 while(sacarDePila(&pila, &prod, sizeof(tProd)))
116     mostrarProducto(&prod);
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<
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 }
```

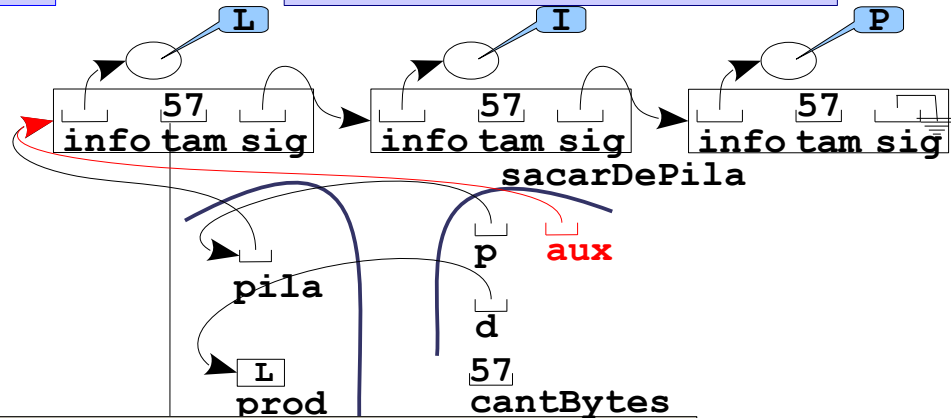


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



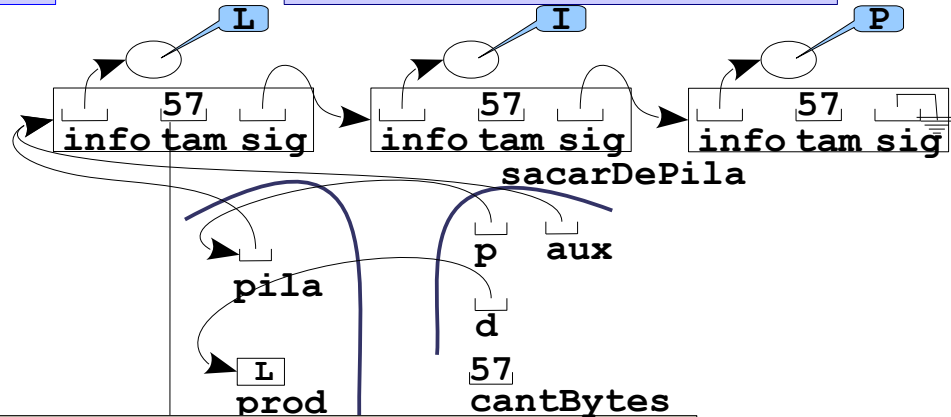
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<
main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
54 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
55 {
56     tNodo *aux = *p;
57
58     if(aux == NULL)
59         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 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<

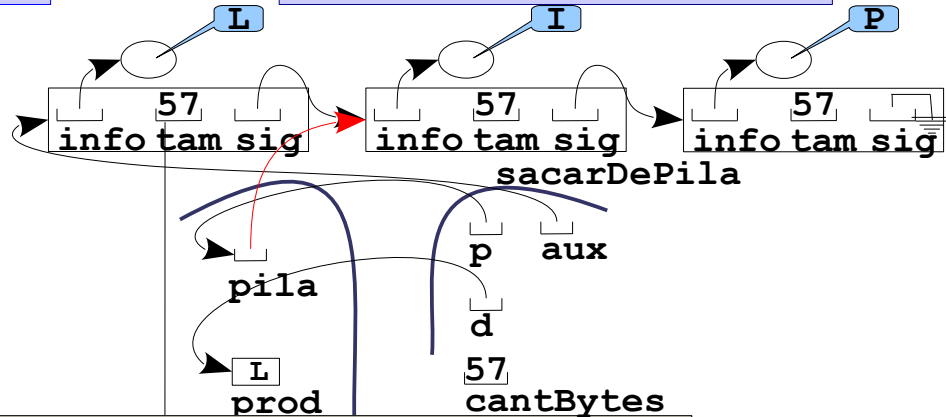
main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
54 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
55 {
56     tNodo *aux = *p;
57
58     if(aux == NULL)
59         return 0;
60     *p = aux->sig;
61     memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<

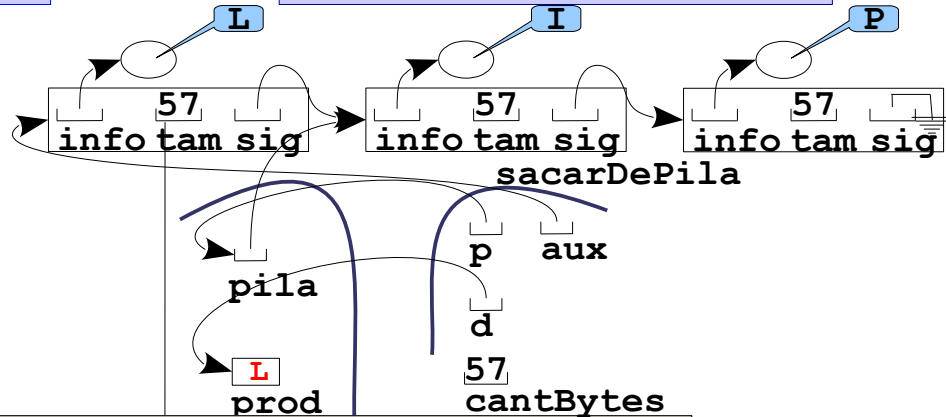
main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
54 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
55 {
56     tNodo *aux = *p;
57
58     if(aux == NULL)
59         return 0;
60     *p = aux->sig;
61     memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



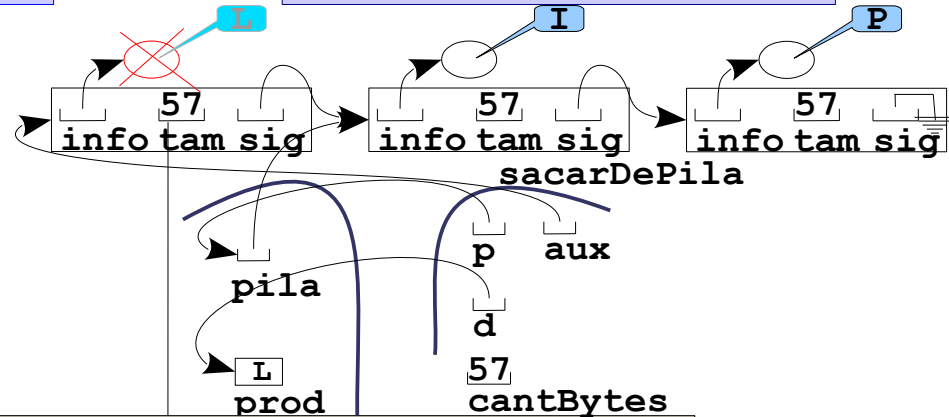
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<
main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
54 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
55 {
56     tNodo *aux = *p;
57
58     if(aux == NULL)
59         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 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

### Implementación estática

### Implementación dinámica



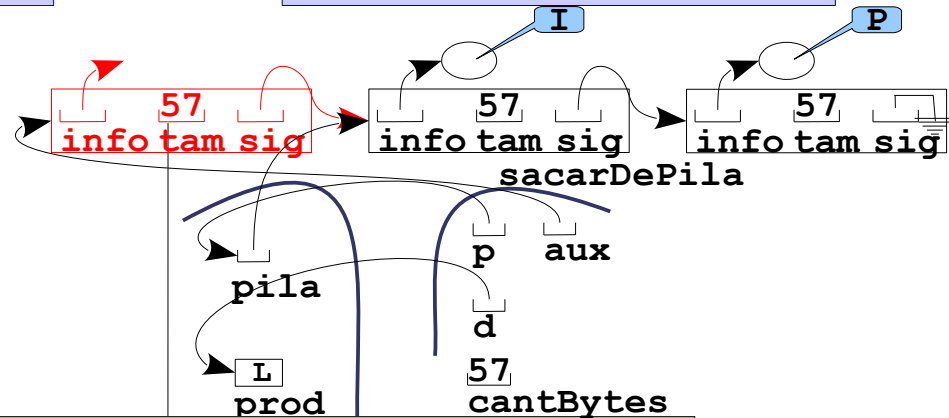
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<
main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
54 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
55 {
56     tNode *aux = *p;
57
58     if(aux == NULL)
59         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 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<

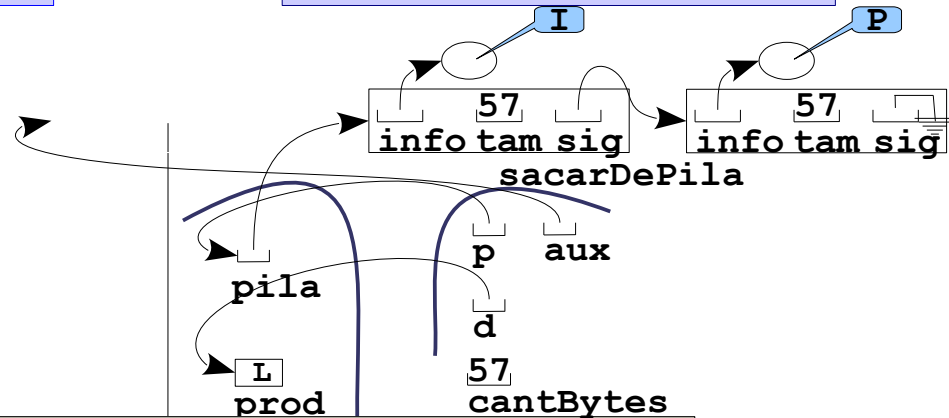
main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
54 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
55 {
56     tNodo *aux = *p;
57
58     if(aux == NULL)
59         return 0;
60     *p = aux->sig;
61     memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<

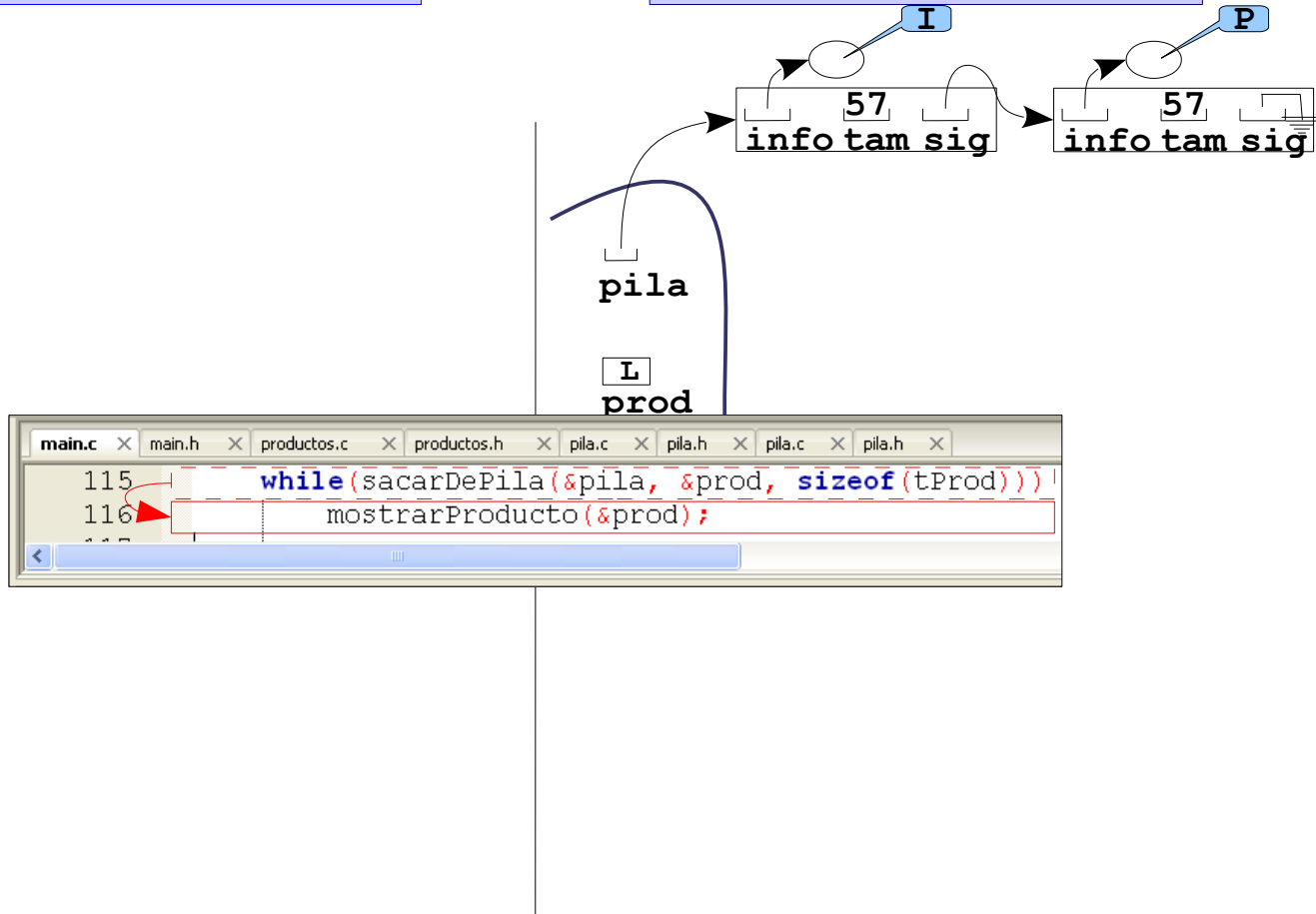
main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
54 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
55 {
56     tNodo *aux = *p;
57
58     if(aux == NULL)
59         return 0;
60     *p = aux->sig;
61     memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



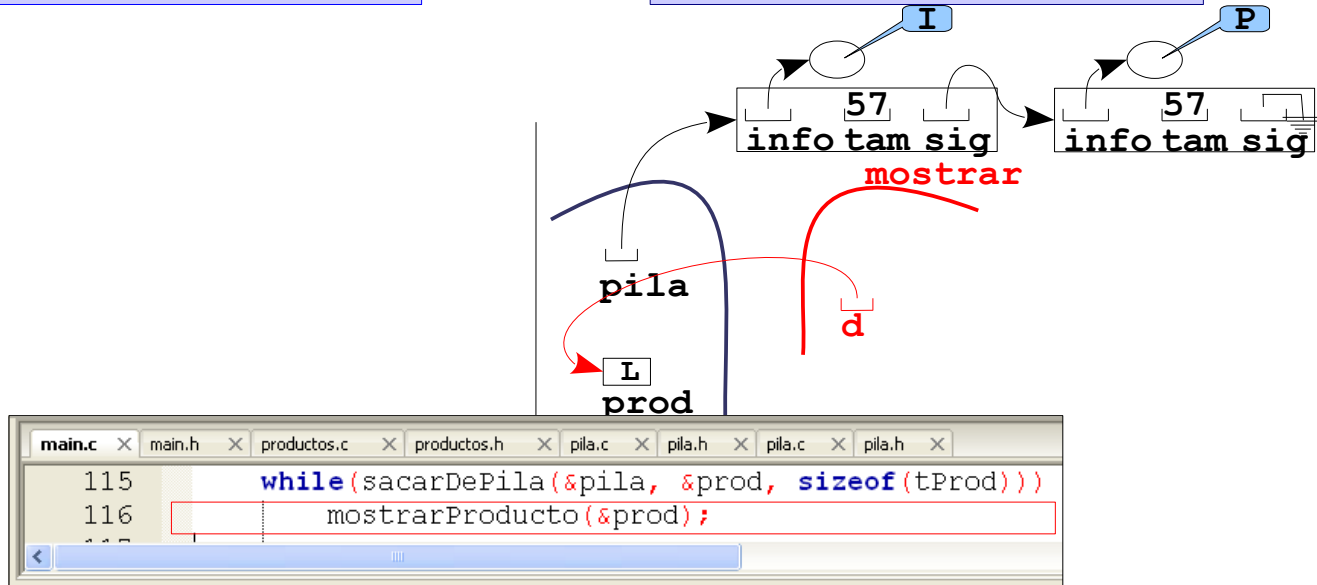


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

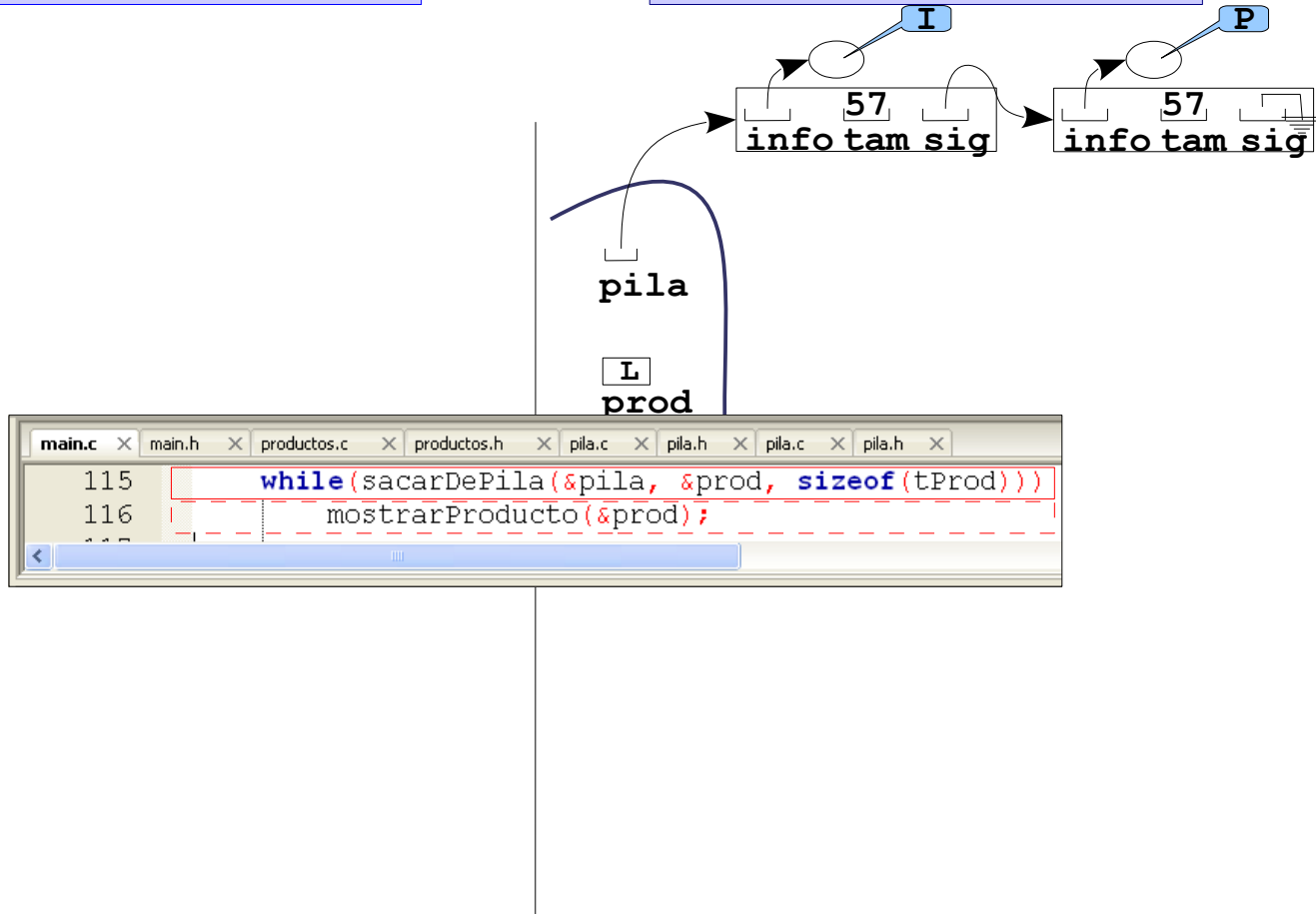


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

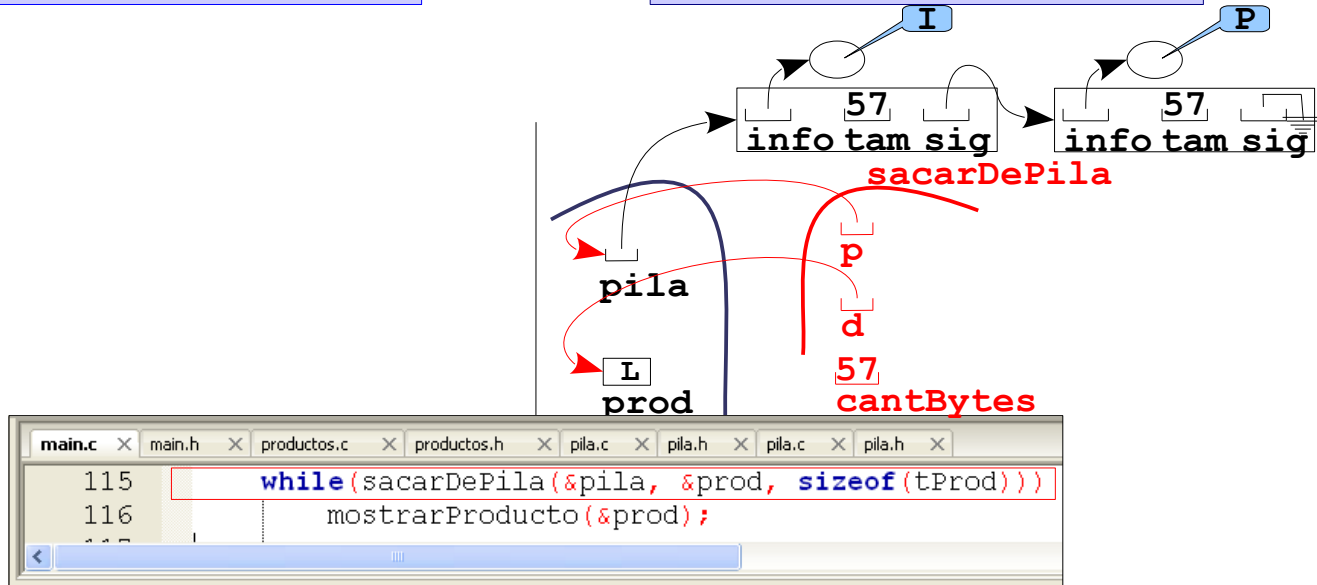


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

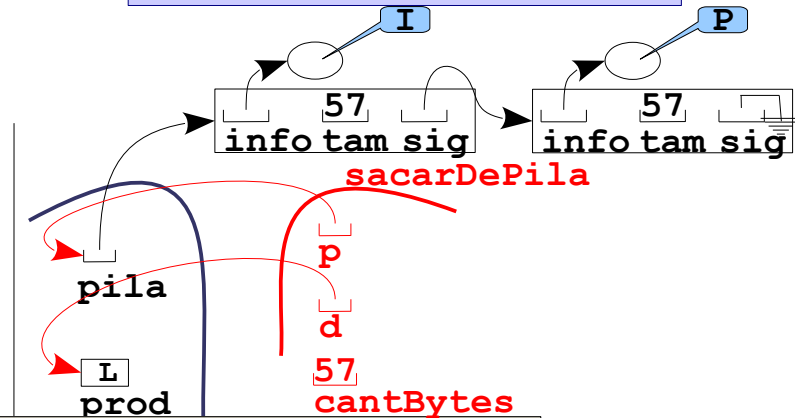


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



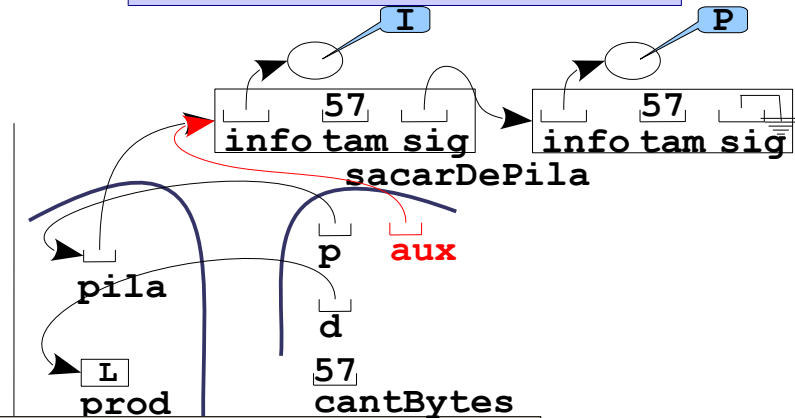
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<
main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
54 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
55 {
56     tNodo *aux = *p;
57
58     if(aux == NULL)
59         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 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<

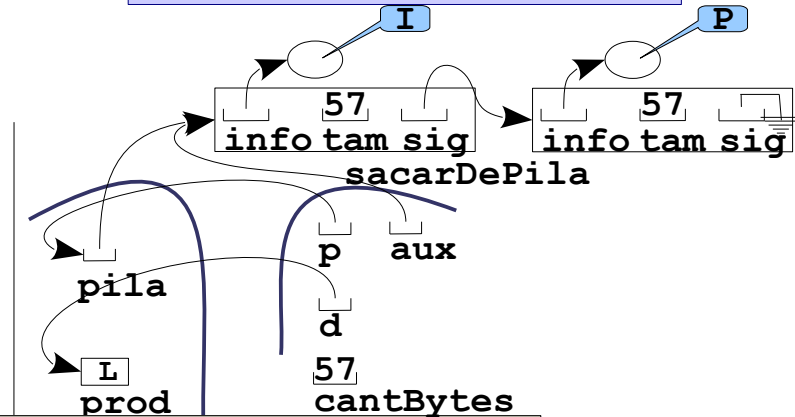
main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
54 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
55 {
56     tNodo *aux = *p;
57
58     if(aux == NULL)
59         return 0;
60     *p = aux->sig;
61     memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



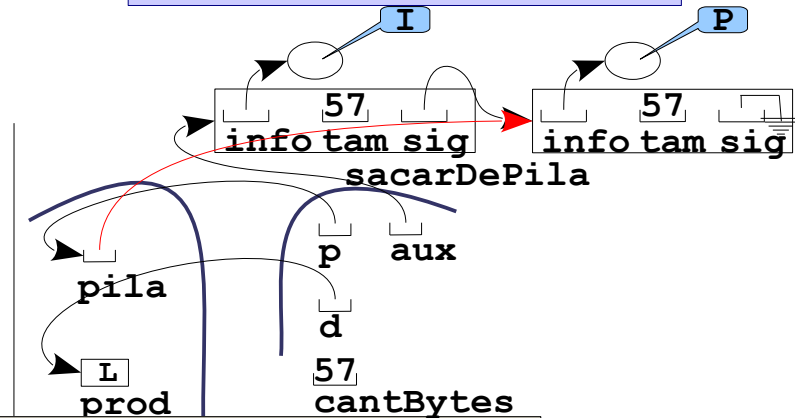
```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<
54 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
55 {
56     tNodo *aux = *p;
57
58     if(aux == NULL)
59         return 0;
60     *p = aux->sig;
61     memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
62     free(aux->info);
63     free(aux);
64     return 1;
65 }
```

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<

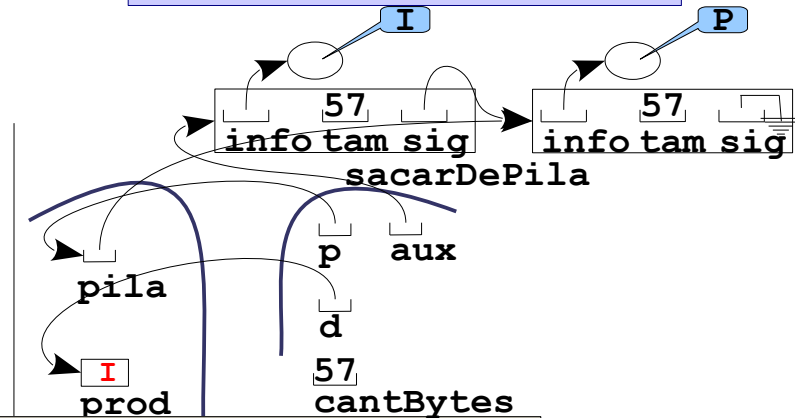
main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
54 int sacarDePila(tPila *p, void *d, unsigned cantBytes)
55 {
56     tNodo *aux = *p;
57
58     if(aux == NULL)
59         return 0;
60     *p = aux->sig;
61     memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));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

# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<
main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
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 }
```

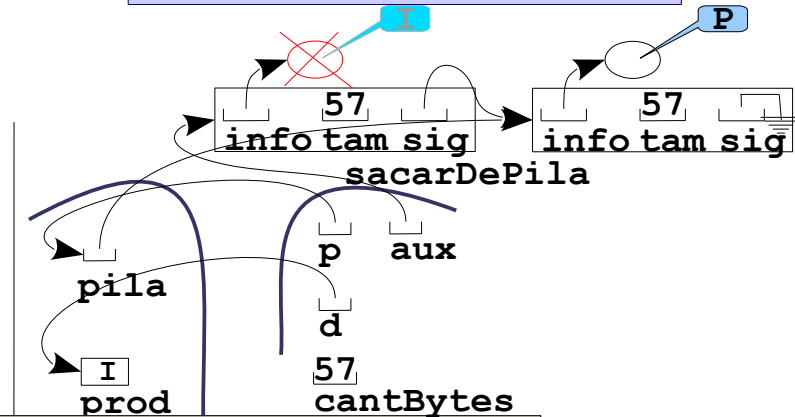


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



```
main.c x main.h x productos.c x productos.h x pila.c x pila.h x pila.c x pila.h x
115
116
<

main.c x main.h x productos.c x productos.h x pila.c x pila.h x *pila.c x pila.h x
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 }
```

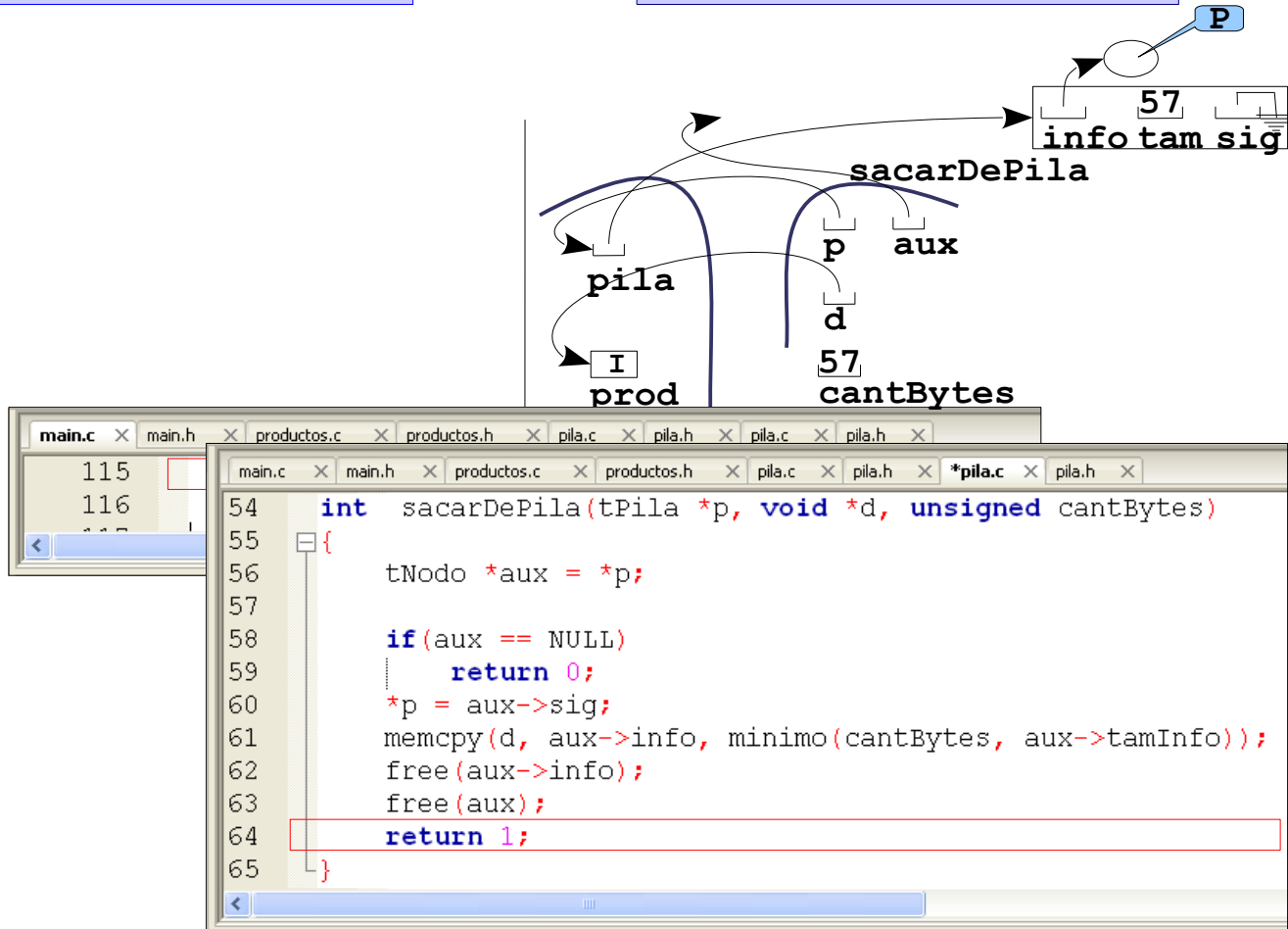


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

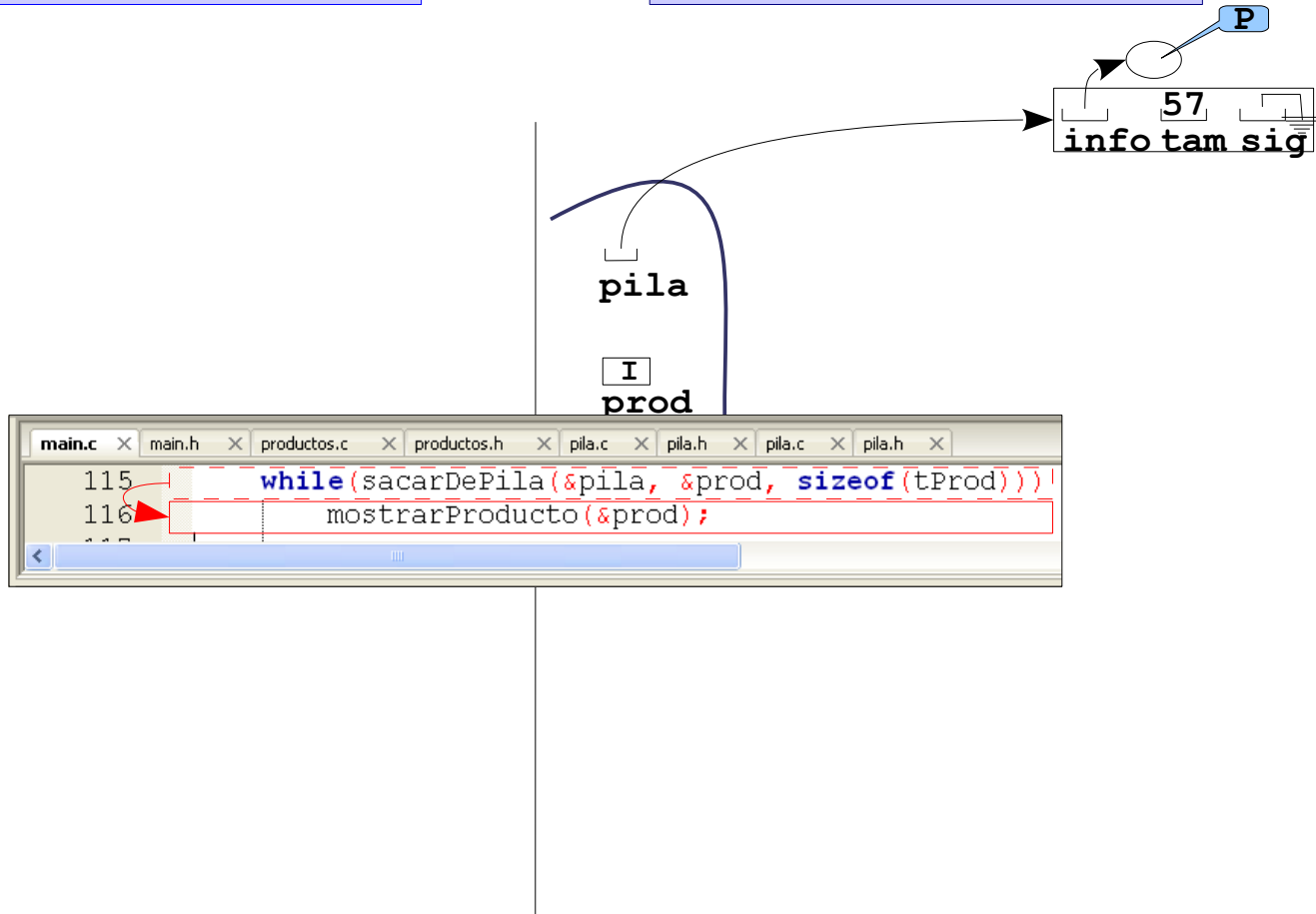


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

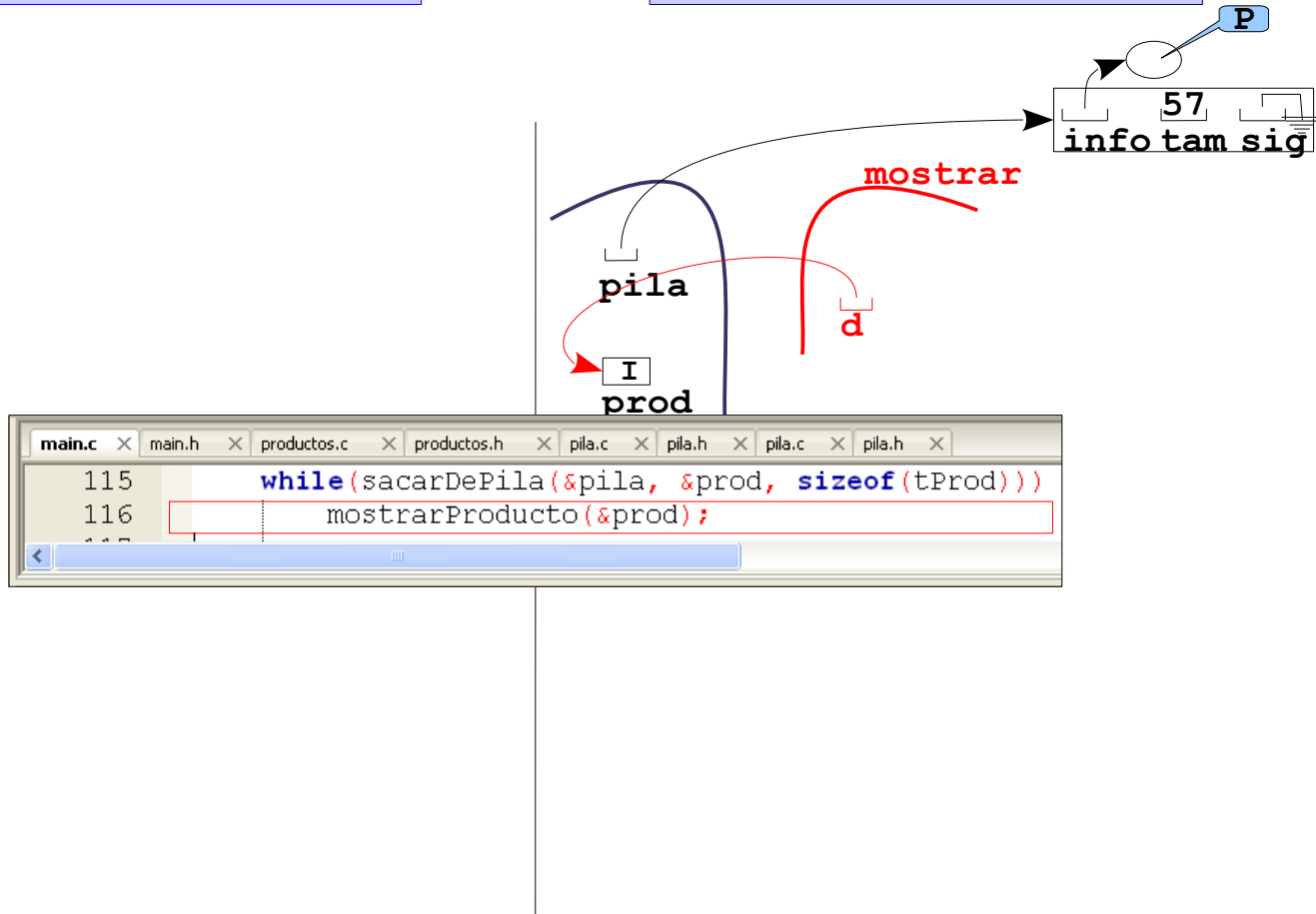


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

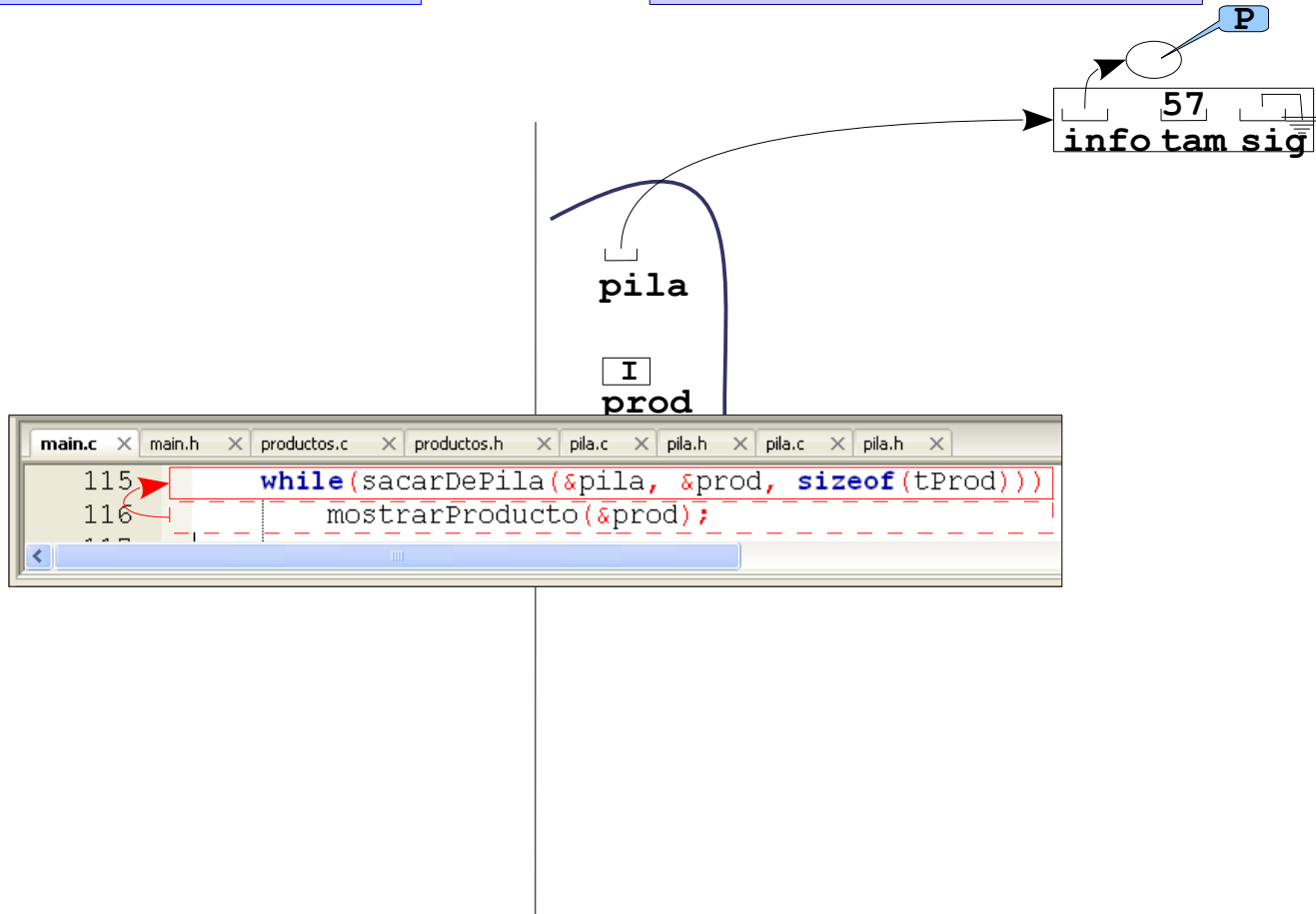


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



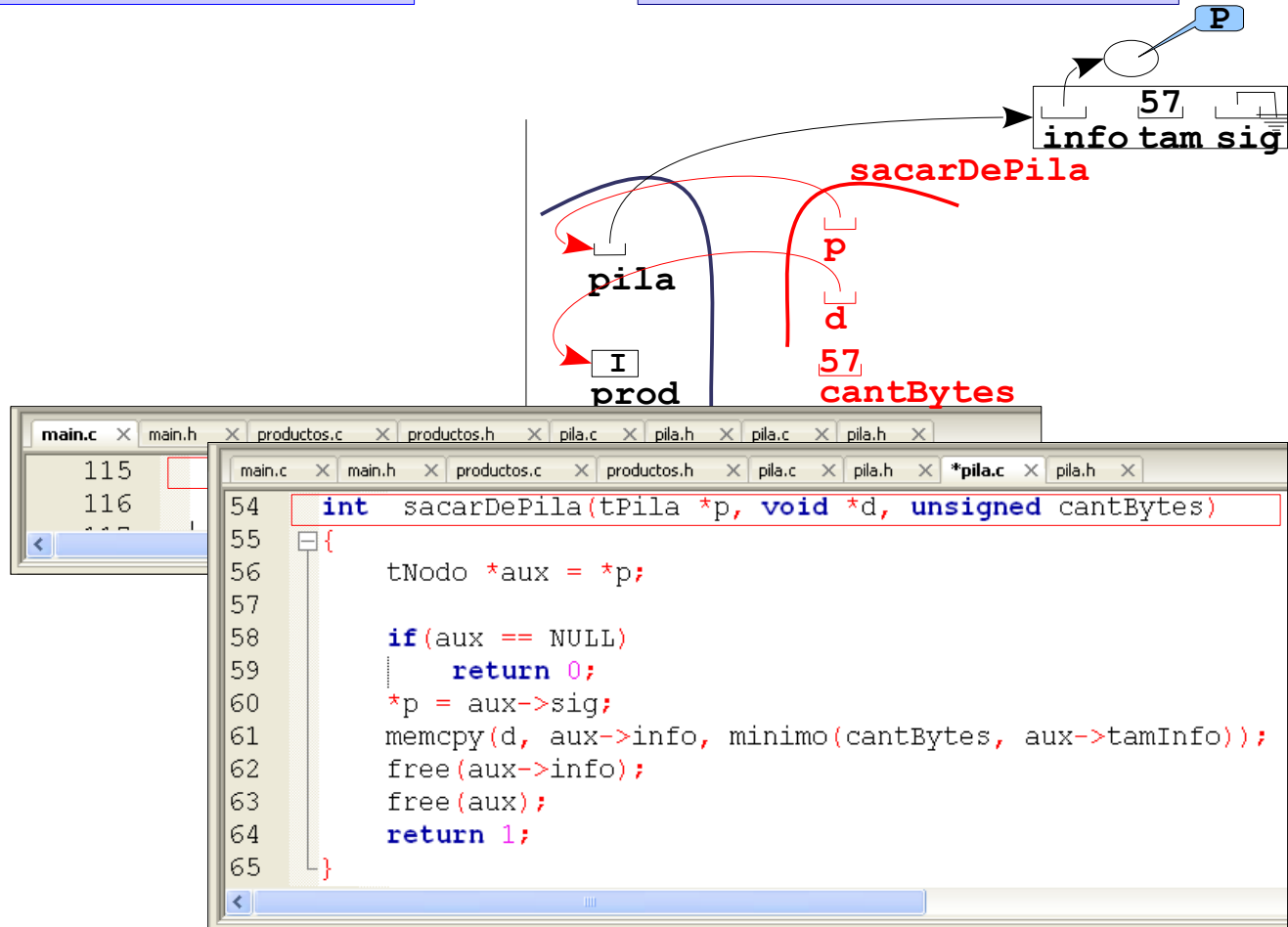


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



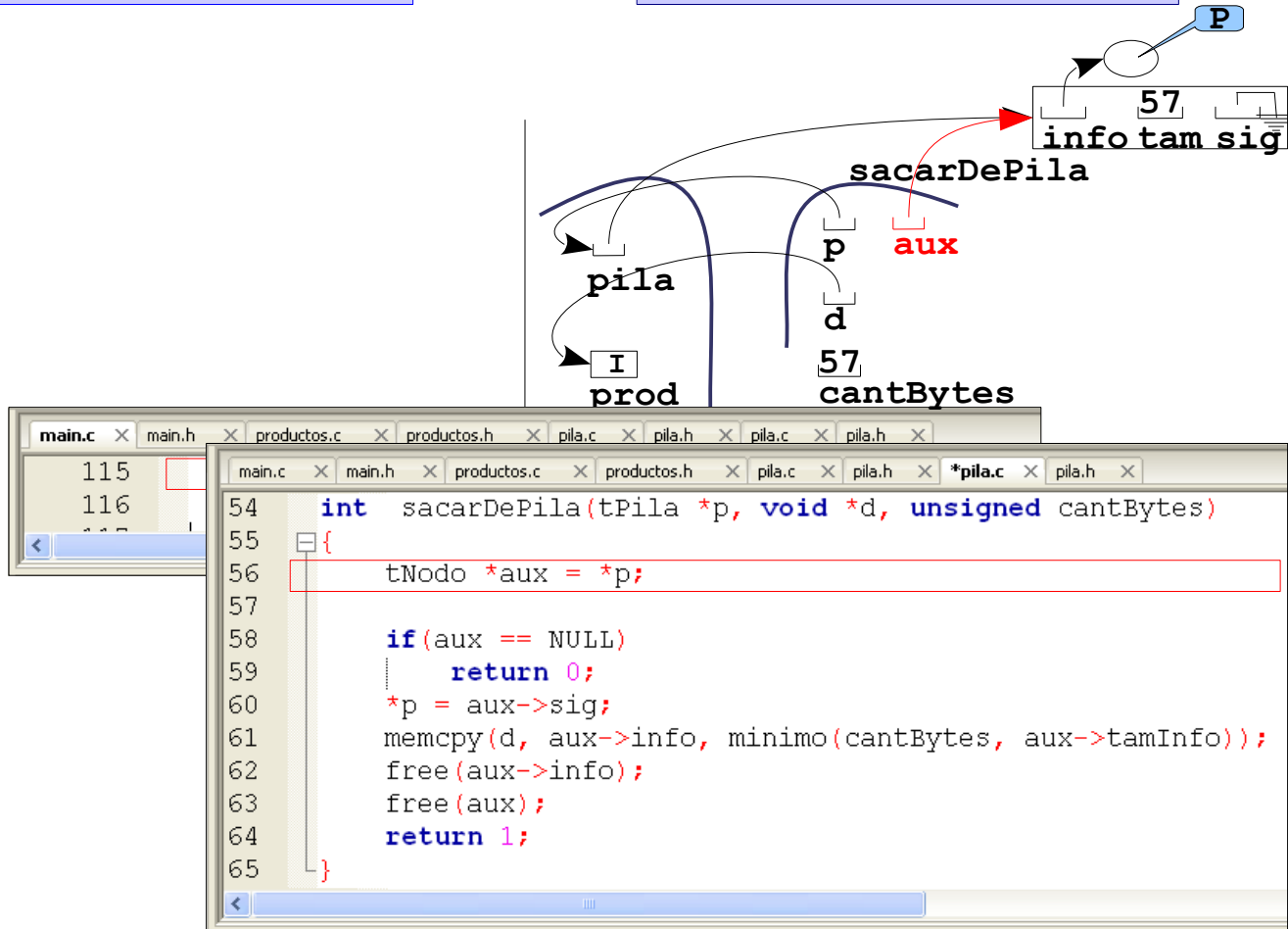


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

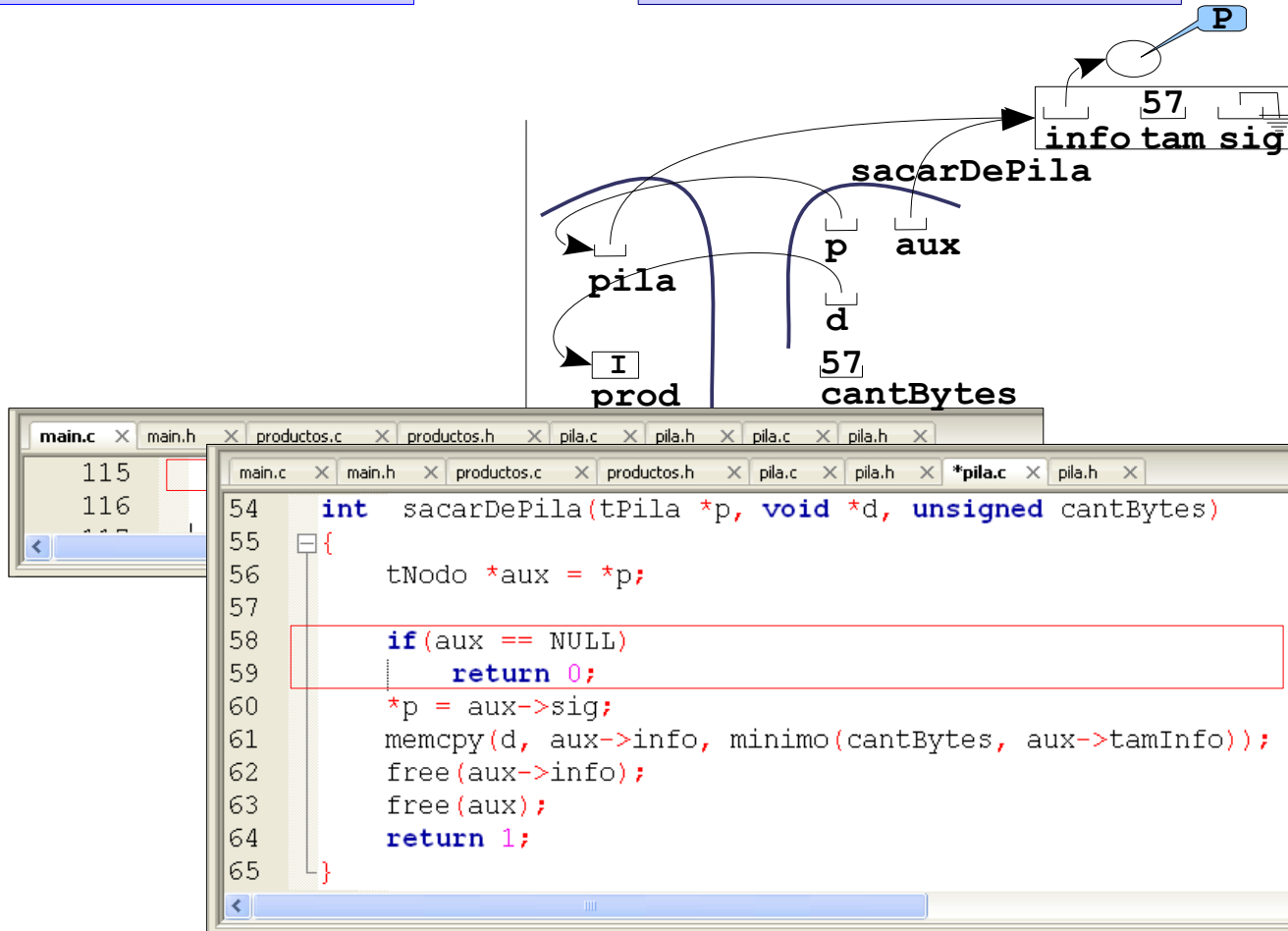


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

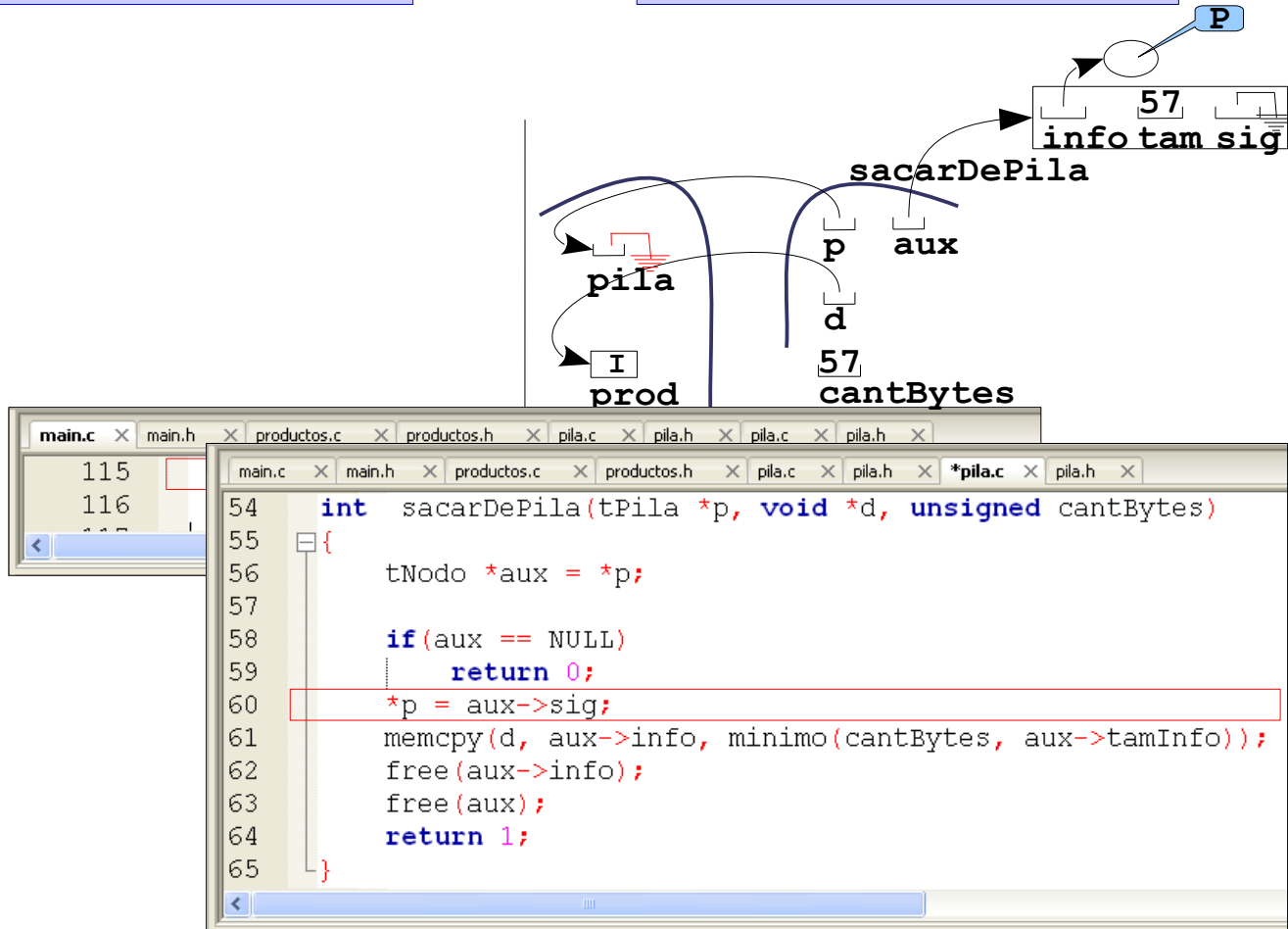


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

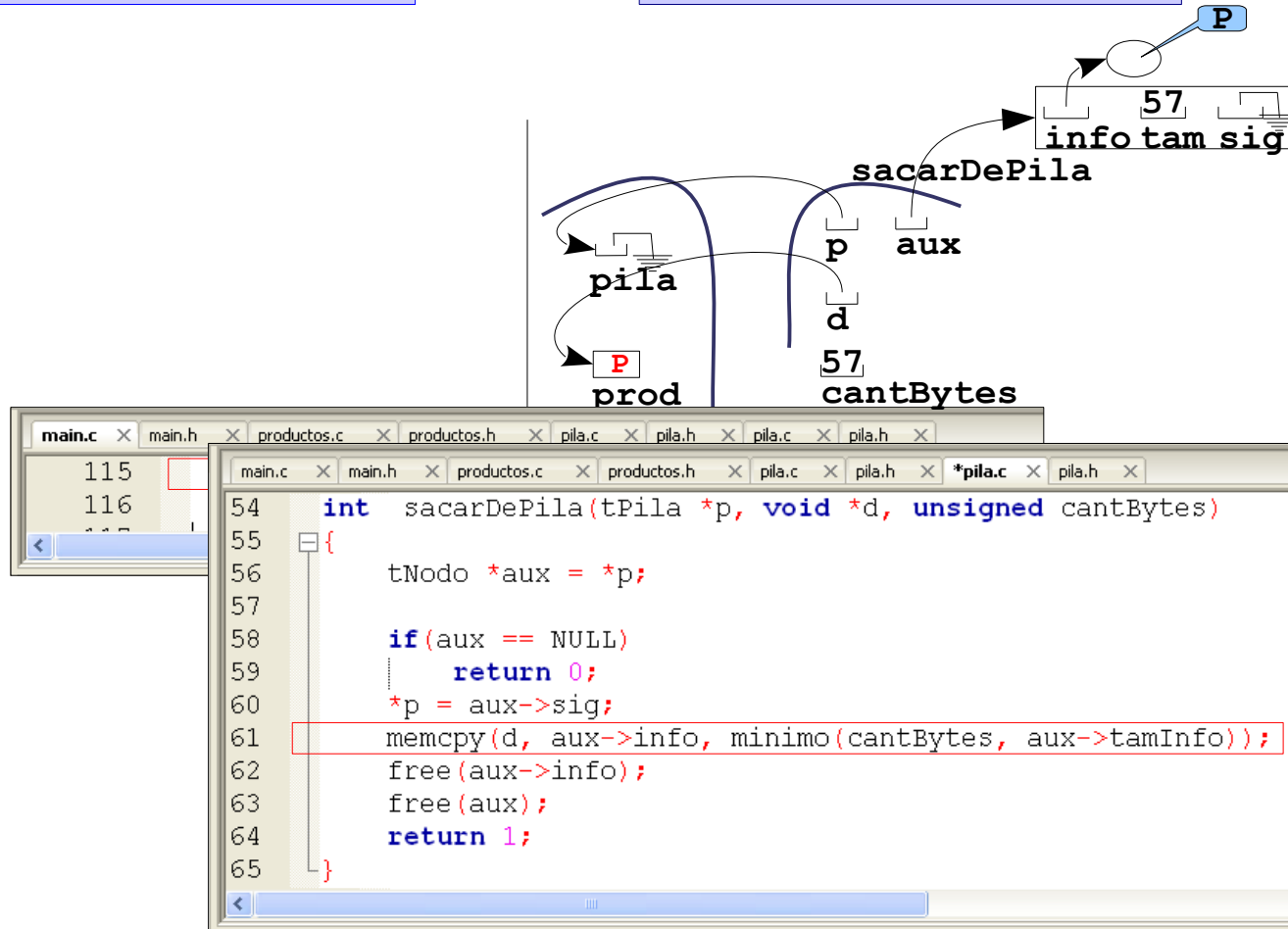


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



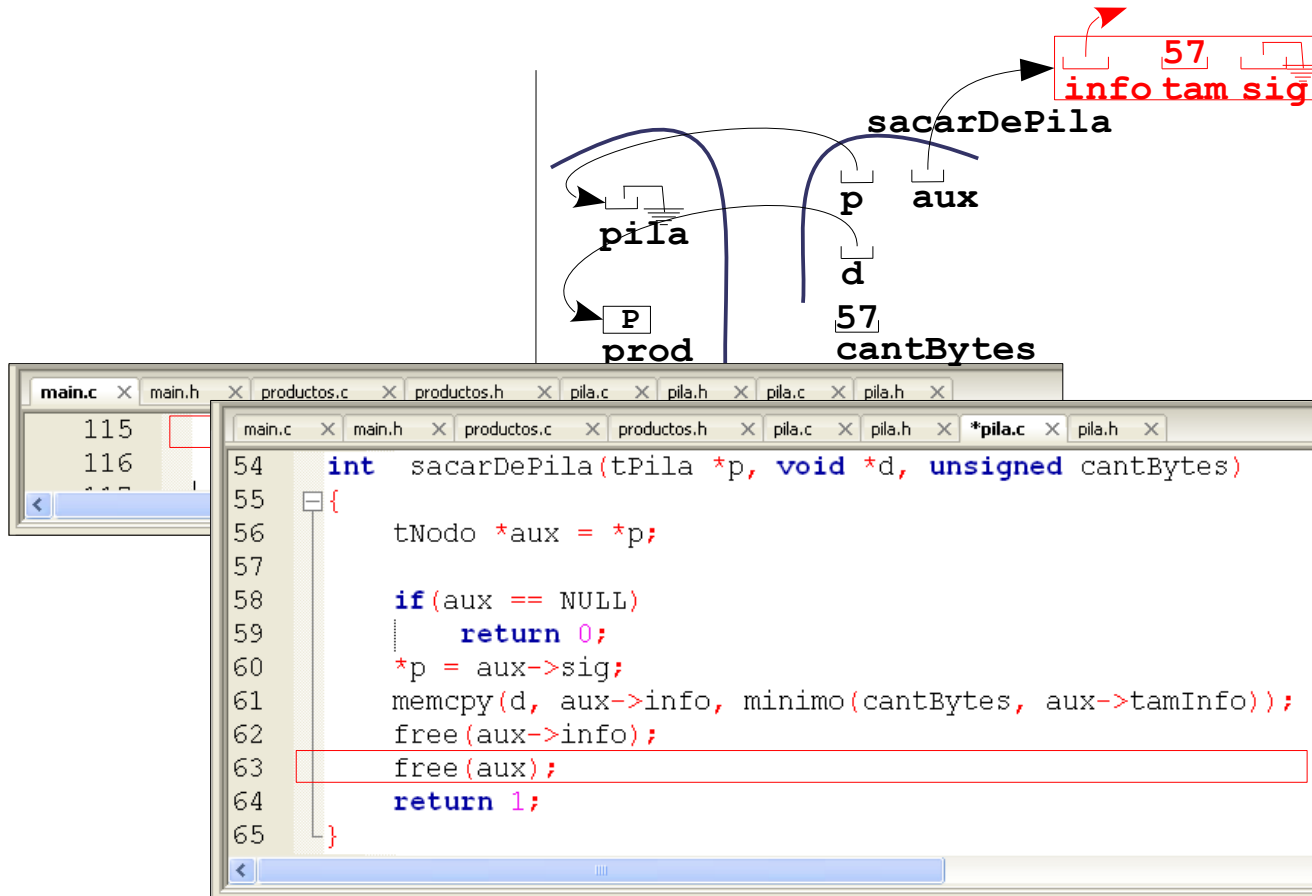


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

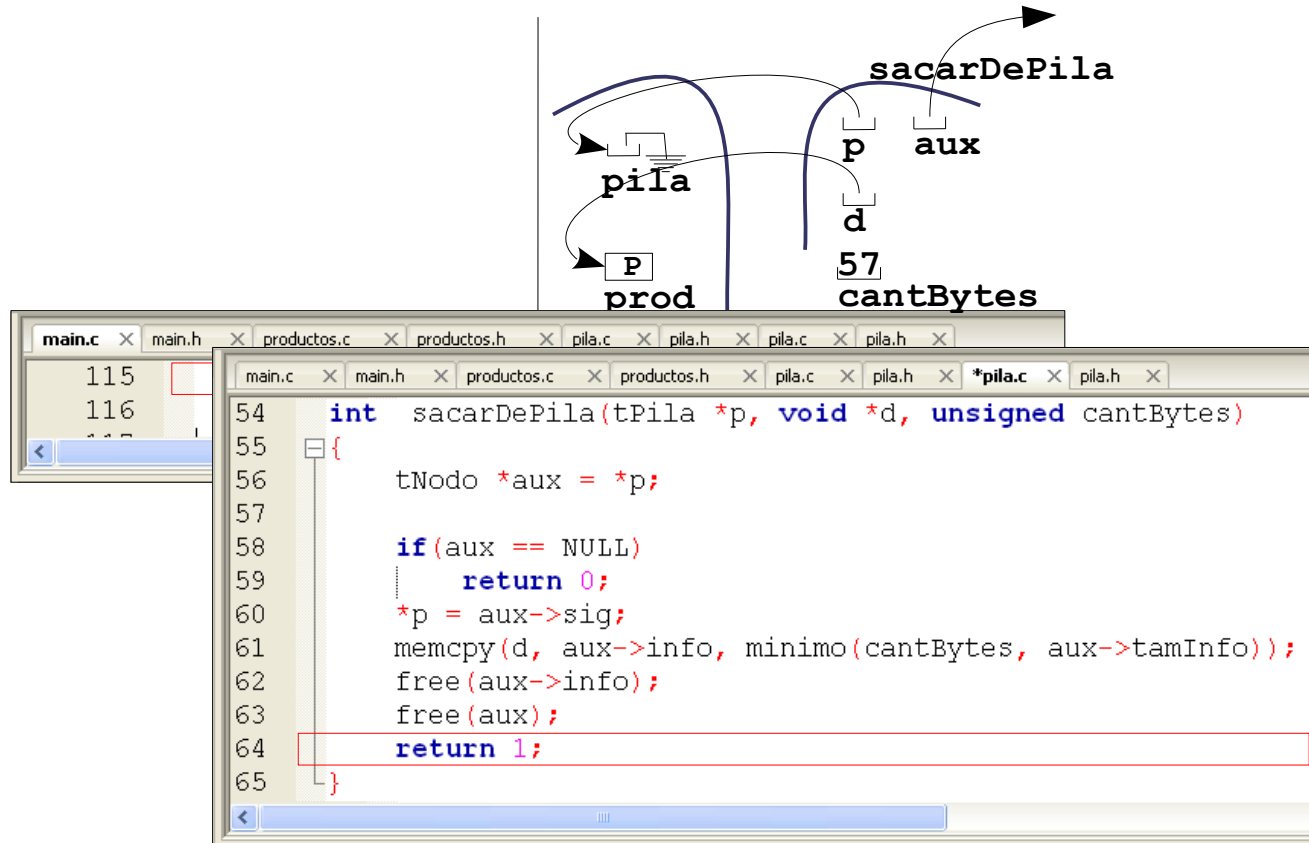


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

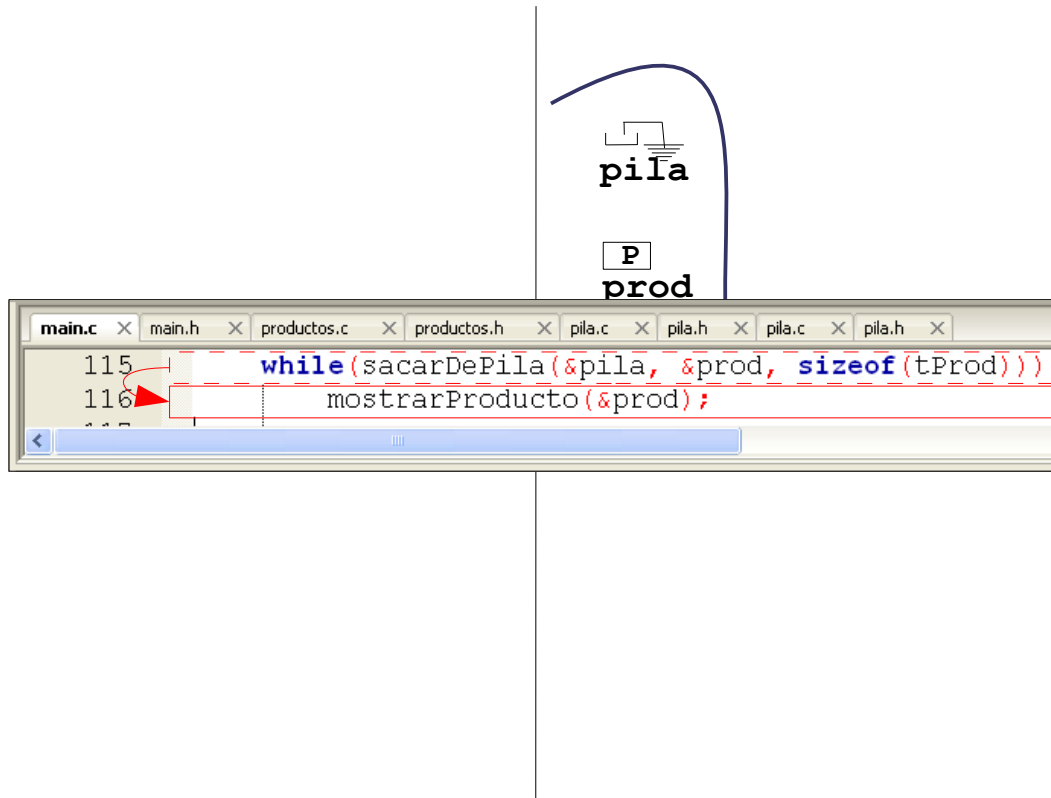


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica



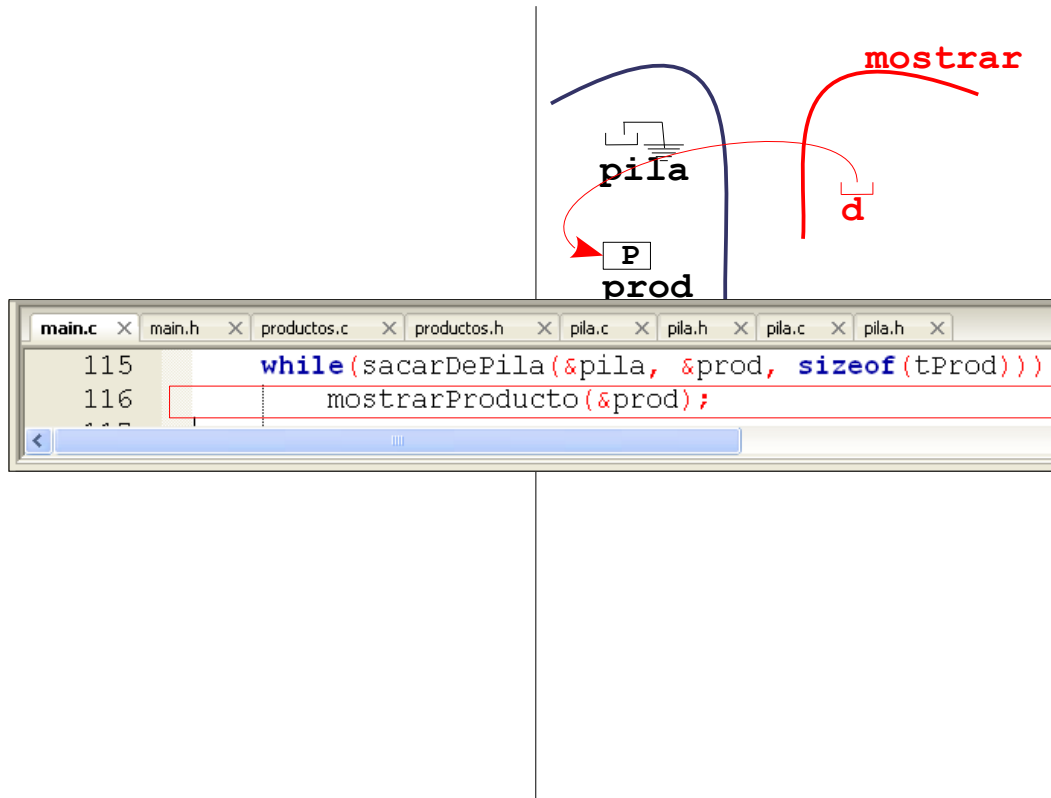


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

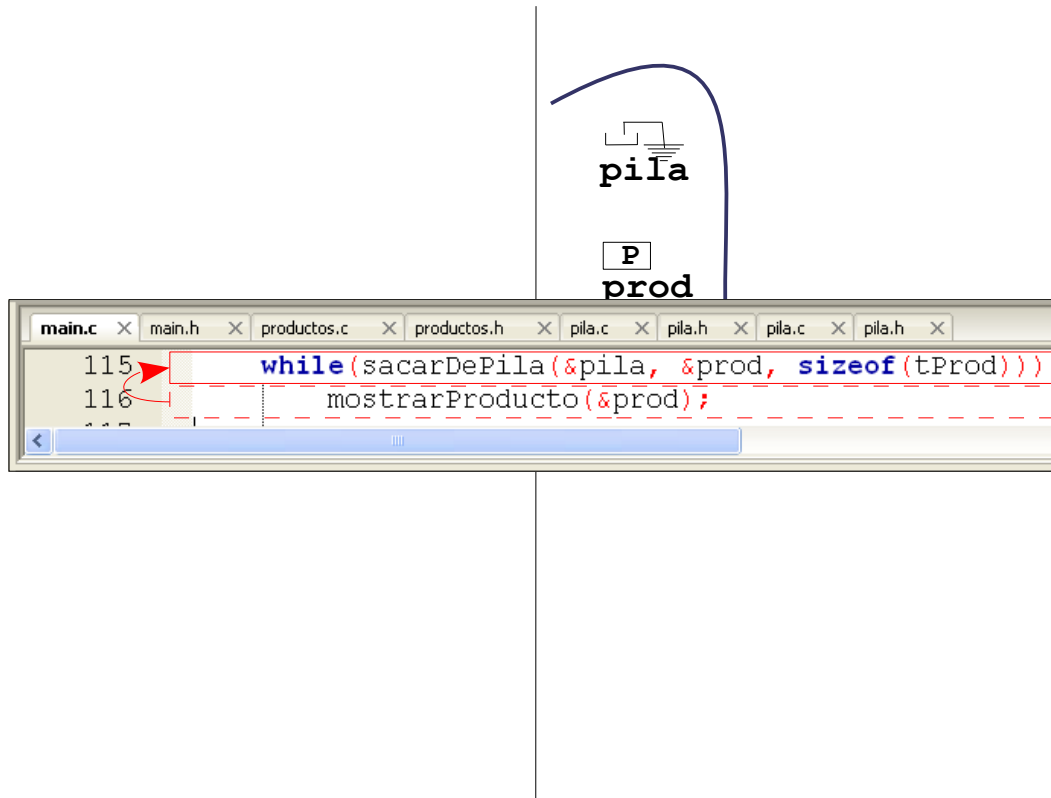


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

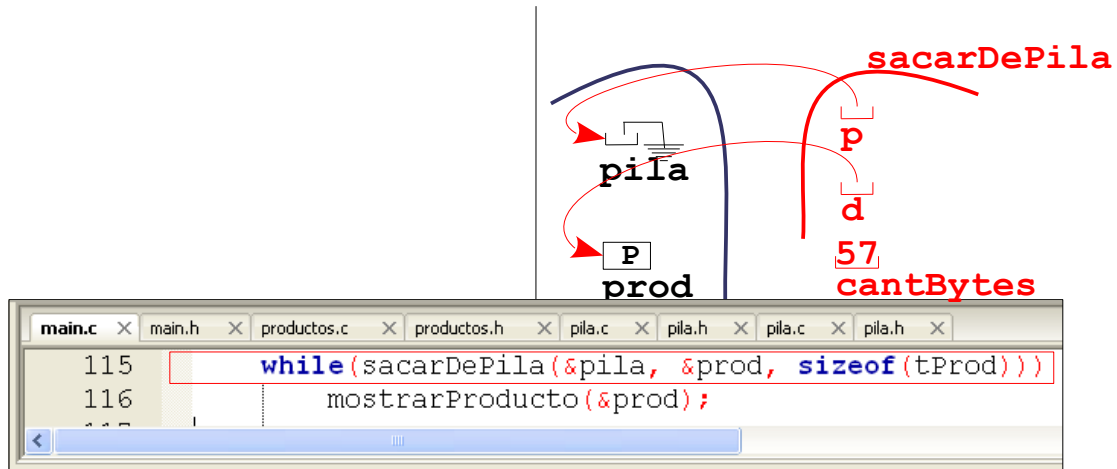


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

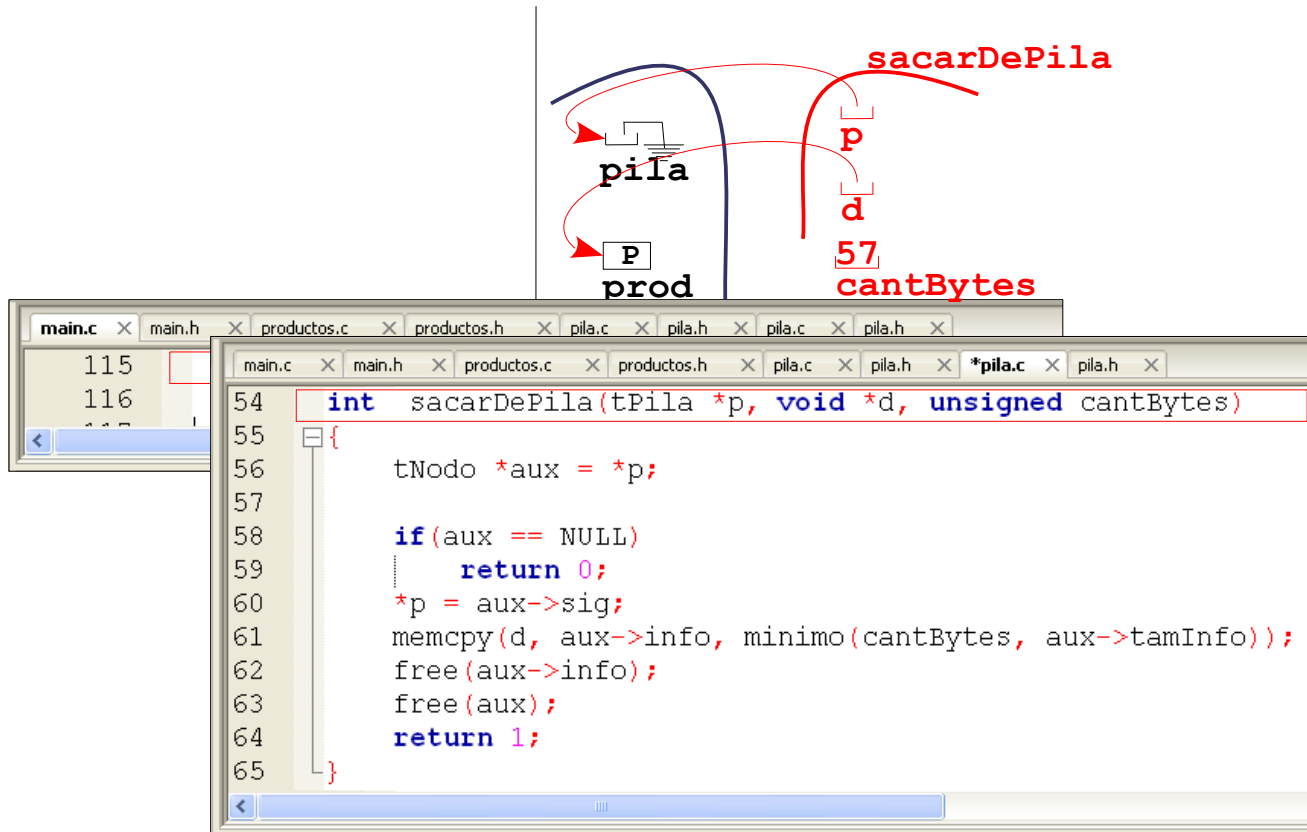


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

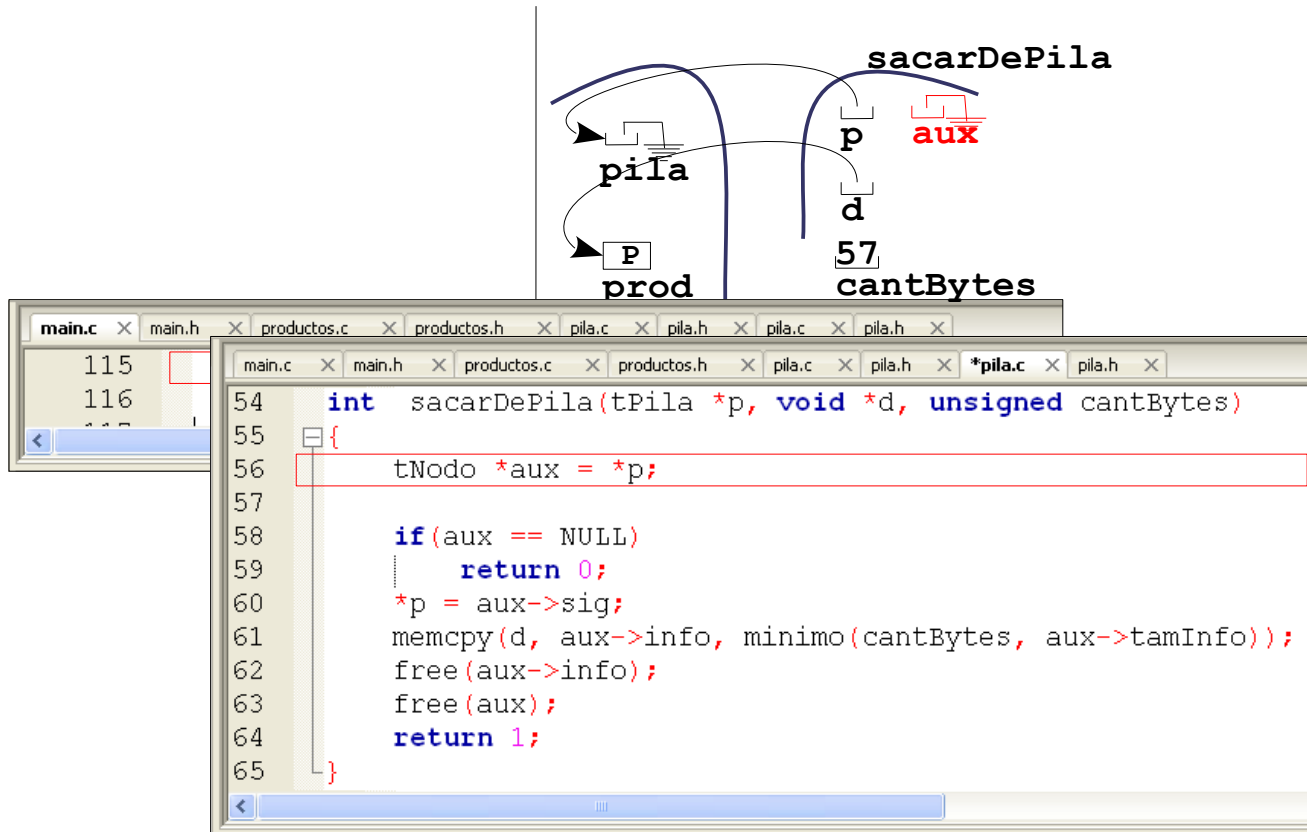


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

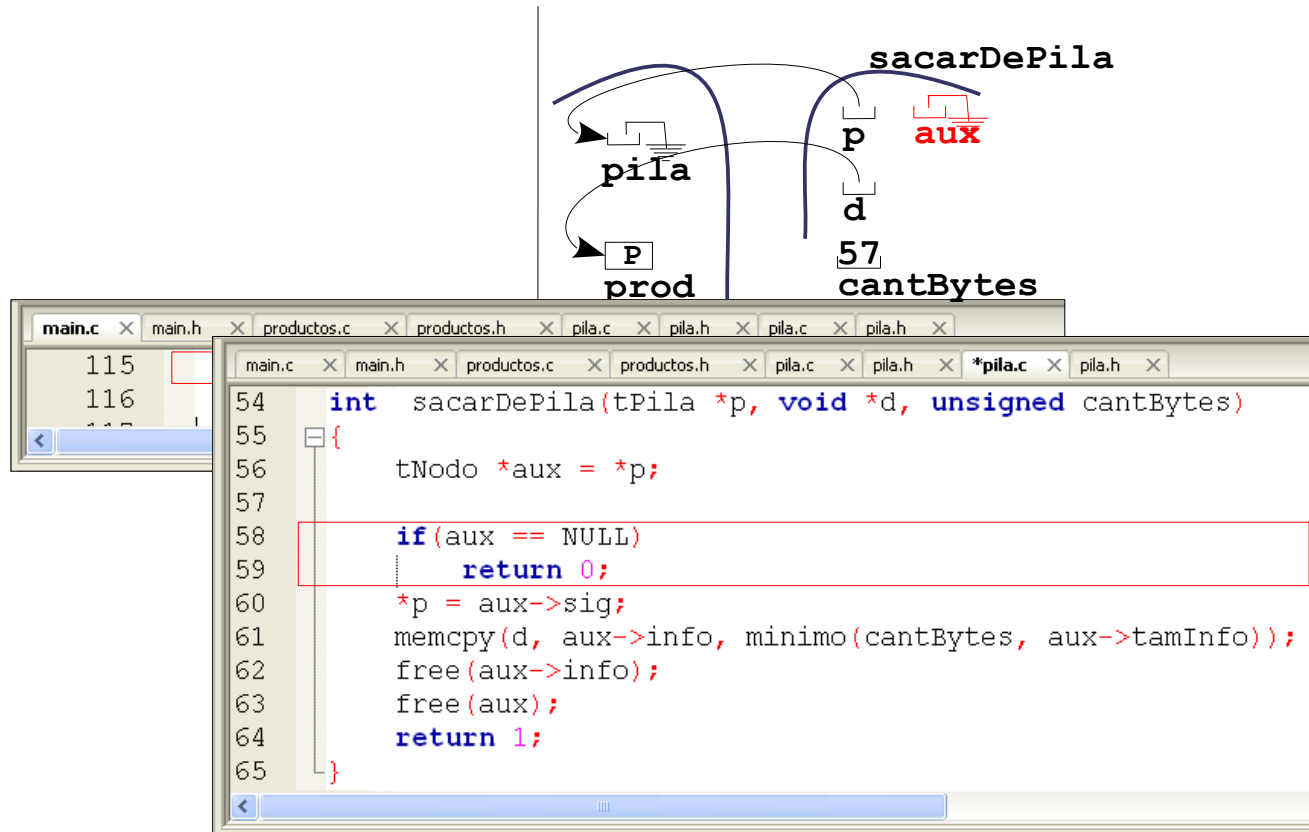


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

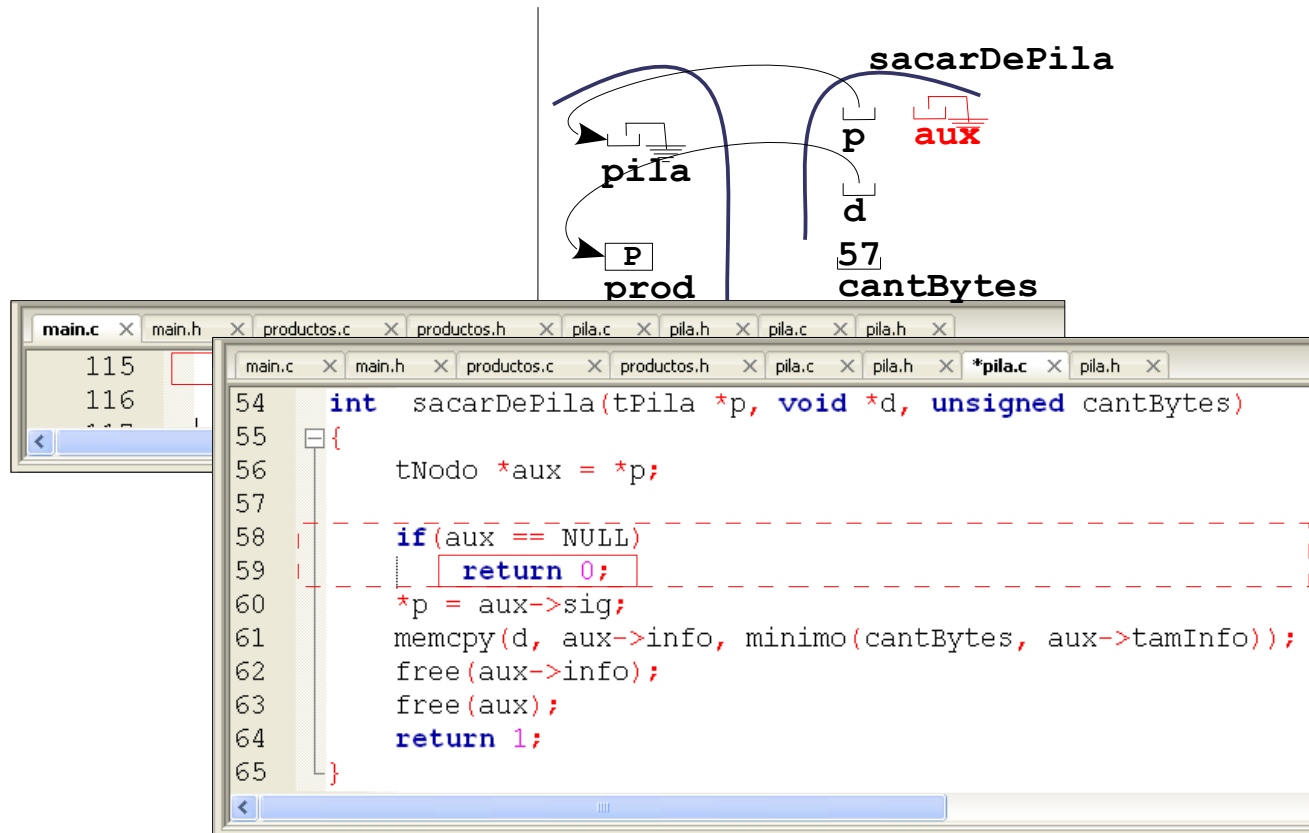


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

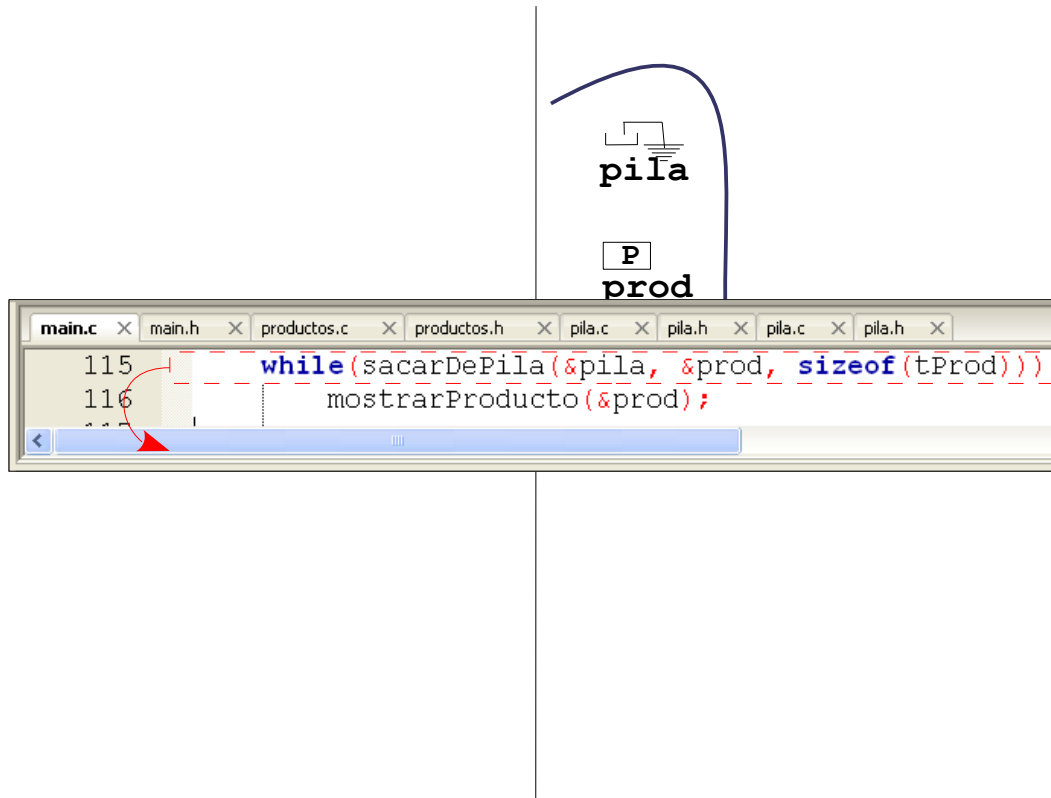


# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica





# Tipos de Datos Abstractos

## Tipo de dato Pila.

Implementación estática

Implementación dinámica

