


La memoria dinámica

RAII: Resource Acquisition Is Initialization



RAII es un modelo de gestión de recursos (en general)

Adquirir recurso

Utilizar recurso

Liberar recurso

Fichero

Memoria dinámica

Conexión de red

Impresora

...

Este esquema permite asegurar un correcto uso de recursos.
Es importante no olvidar la liberación de recursos tras su uso.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 1



La memoria dinámica

RAII: Resource Acquisition Is Initialization



Gestión manual de memoria dinámica: el programador decide cuando pide memoria y cuando la libera.

Riesgos:

- Olvidar su liberación (*memory leak*)
- Liberar varias veces
- Uso de la memoria tras su liberación

El esquema RAII intenta evitar los riesgos.

Por ahora haremos RAII manual, más adelante (usando clases) lo automatizaremos.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 2

La memoria dinámica

RAII: Resource Acquisition Is Initialization

Ejemplo de mala organización de código:

<http://www.hackcraft.net/raii/>

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)

La memoria dinámica

RAII: Resource Acquisition Is Initialization

La forma de aplicar RAII es:

- Un objeto adquiere sus recursos en la inicialización.
- Un objeto libera sus recursos cuando se destruye.


En nuestro contexto:

- Un objeto (**variable**) pertenece a un **ámbito** determinado:
 - Inicialización:** Cuando se declara.
 - Destrucción:** Cuando se acaba su ámbito.

```
void procesar()
{
    int *ptr;
    ptr = new int [10];
    ... hacer lo que sea con ptr ...
    delete [] ptr;
}
```


Ámbito de ptr

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)



La memoria dinámica

RAII: Resource Acquisition Is Initialization



```
void procesar()
{
    int *ptr;
    ptr = new int [10];
    ... hacer lo que sea con ptr ...
    // Olvidamos hacer delete
}
```

"Memory leak"
(Perdemos memoria)

```
void procesar()
{
    int *ptr;
    ptr = new int [10];
    ... hacer lo que sea con ptr ...
    delete [] ptr;
    // ... usamos de nuevo ptr ...
}
```

Usamos ptr fuera de su ámbito.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)
5



La memoria dinámica

RAII: Resource Acquisition Is Initialization



Nuestra situación: trabajamos con **TDA**

Modularización en:

{

Ficheros

Funciones

TDA

*En general, no nos sirve un modelo en el que **cada** función deba, necesariamente, reservar-usar-liberar memoria dinámica.*

TDA

{


Representación interna

Funciones

Las funciones no tienen sentido individualmente


Las funciones están en el ámbito/contexto del TDA

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)
6



La memoria dinámica

RAII: Resource Acquisition Is Initialization



TDA VectorDinámico:

```


Get(v,i)  // Devuelve el elemento de la posición i-ésima
Set(v,i,p) // Asigna p a la posición i-ésima
Reserva(v,n) // Reserva memoria para n elementos
  
```

```

VectorDinamico v;
...
Reserva(v,6);      // Reservamos memoria para 6 elementos
// Supongamos que ponemos en v {4,7,6,2,3,1}
...
elem = Get(v,3);   // Obtiene el 2 y lo asigna a elem
...
Set(v,2,34);       // Cambia el 6 por 34
...
  
```


Get y Set no reservan ni liberan memoria, sólo hacen uso del recurso
Reserva no usa ni libera, sólo reserva memoria

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)
7



La memoria dinámica

RAII: Resource Acquisition Is Initialization



```

VectorDinamico v;
...
Reserva(v,6);      // Reservamos memoria para 6 elementos
// Supongamos que ponemos en v {4,7,6,2,3,1}
...
elem = Get(v,3);   // Obtiene el 2 y lo asigna a elem
...
Set(v,2,34);       // Cambia el 6 por 34
...
  
```

En este caso: {

- Nuestro objeto es v
- Su ámbito es este:
- Cada función es parte del objeto v

Aplicamos el modelo RAII al TDA y no a cada función:

Es VectorDinamico el que hace uso de los recursos (no las funciones)
 VectorDinamico usa funciones para reservar/liberar/usar recursos.

(En el ejemplo faltaría liberar el recurso al final del ámbito de v)

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)
8

La memoria dinámica
RAII: Resource Acquisition Is Initialization

```

VectorDinamico v;
...
Reserva(v,6);      // Reservamos memoria para 6 elementos
// Supongamos que ponemos en v {4,7,6,2,3,1}
...
elem = Get(v,3);   // Obtiene el 2 y lo asigna a elem
...
Set(v,2,34);      // Cambia el 6 por 34
...
Libera(v);
  
```

```

graph TD
    A[Reserva()] --> B[Otras funciones: Get, Set, ...]
    B --> C[Libera()]
  
```

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 9

La memoria dinámica
RAII: Resource Acquisition Is Initialization

Malas ideas para aplicar el modelo RAII:

```

int *ptr;
ptr = new int [10];
... procesamos prt ...
Liberar(ptr);
  
```

```



int *ptr;
ptr = Reservar(10);
... procesamos prt ...
delete [] ptr;
  
```

Distinto nivel de detalle: Cuando vemos un new a un nivel, esperamos ver un delete al mismo nivel ...

- Al ver un new podemos estar tentados a hacer un delete (*doble liberación*).
- Al no ver new podemos olvidar hacer delete (*memory leak*).

Este uso del modelo, implica conocer las interioridades de Reservar y Liberar para usarlas correctamente → **Esto va en contra de la abstracción y el encapsulamiento.**

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 10



La memoria dinámica

RAII: Resource Acquisition Is Initialization

En resumen ...

- Debéis tener siempre claro cual es el ámbito de cada objeto.
- Se reservan recursos (inicializar) al comienzo de la vida del objeto.
- Se liberan recursos (destruir) al final de la vida del objeto.

En mitad de la "vida" del objeto no se deben reservar/liberar recursos.

*Una función de un TDA puede liberar memoria por algún motivo, pero debe dejar el objeto en un estado **consistente** tras esa liberación:*

- Reservando nueva memoria.*
- Inicializando el objeto de nuevo.*
- ...*

Más que liberar recursos, estaríamos optimizando o modificando el uso de dichos recursos.

Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 11