Estructura de Datos y de la Información

Tema 1: Gestión dinámica de la memoria

LIDIA
Laboratorio de Investigación y desarrollo en Inteligencia Artificial

Departamento de Computación
Universidade da Coruña, España

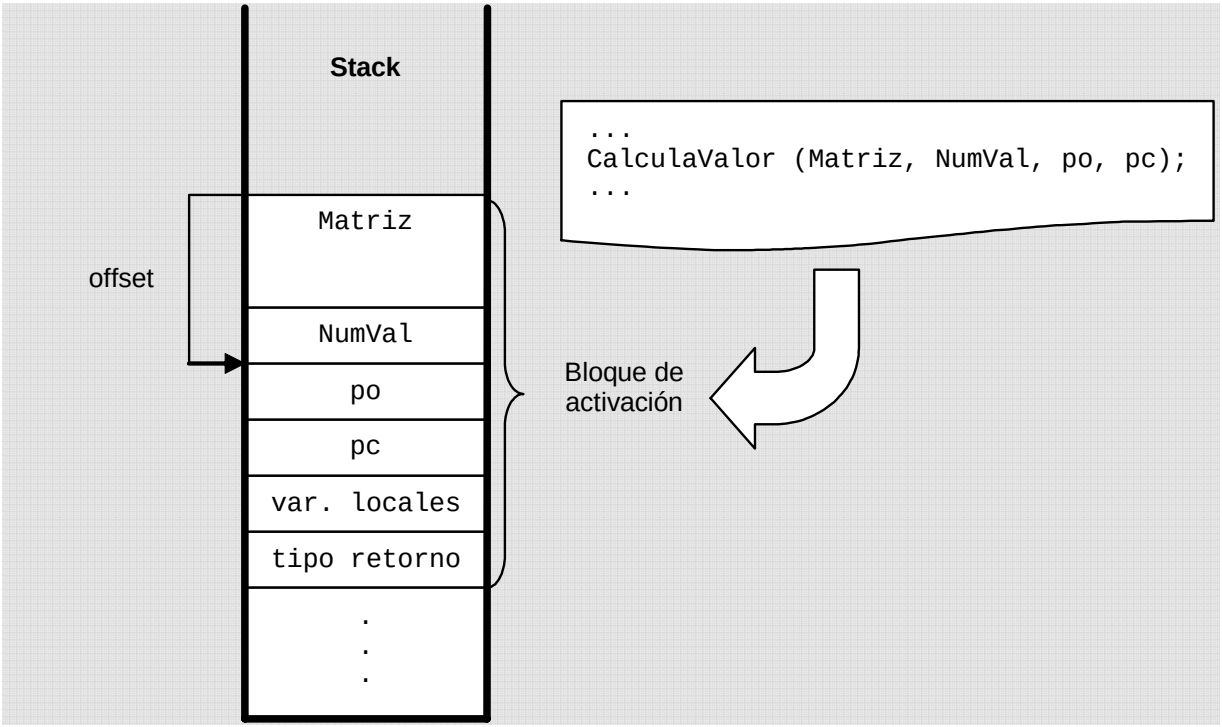
Índice

1. Organización de la memoria de un programa
2. Definición de variables de tipo puntero
3. Reserva y destrucción dinámica de memoria
4. Asignación y comparación de punteros

Organización de la Memoria

- **Existen dos lugares en la memoria de un computador que pueden utilizarse para almacenar elementos: la pila (stack) y el montículo (heap).**
- **Pila (stack)**
  - **Es una estructura en la cual solo se pueden incorporar o eliminar elementos por un solo punto denominado cima.**
  - **Se utiliza para las llamadas a funciones, cuando se llama a una función se reserva espacio en la pila para los valores locales, los parámetros de dicha función, el tipo de retorno, etc.**
  - **Estos valores existen mientras se esté ejecutando la función y se eliminan y se recupera la memoria que ocupaban cuando la función termina.**
  - **Si una función llama a otras funciones sus valores se van amontonando en la cima de la pila.**

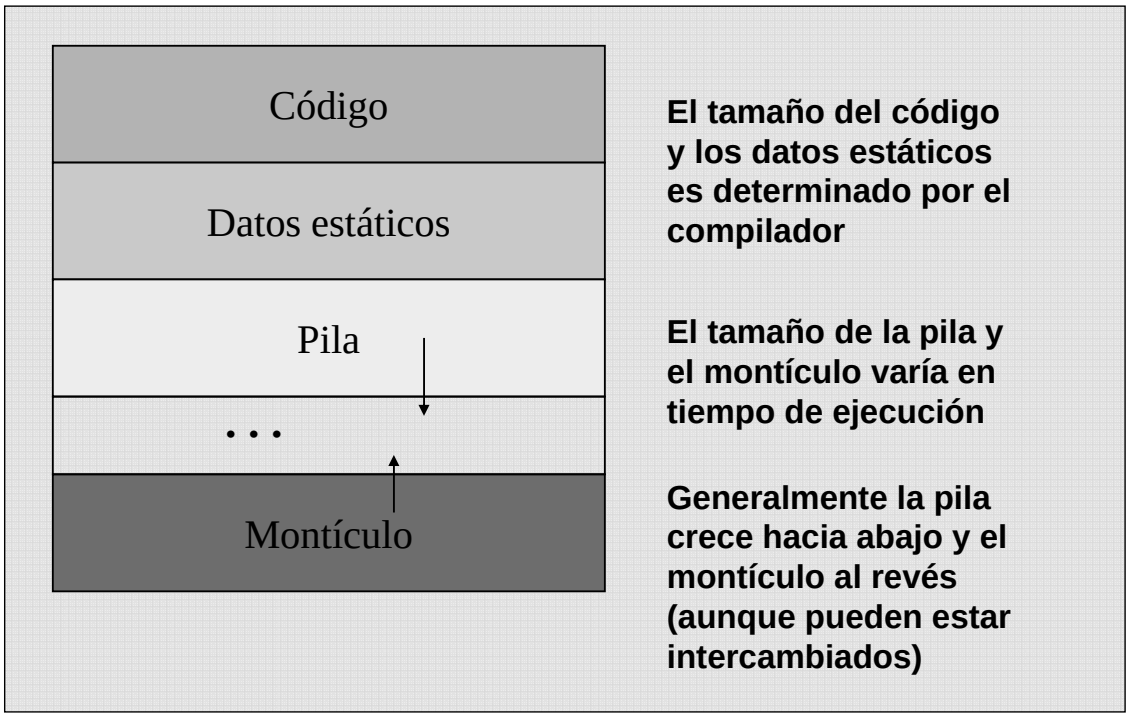
Organización de la Memoria



Organización de la Memoria

- **Montículo (heap)**
  - **Parte de la memoria que no está ligada a la llamada de funciones**
  - **La memoria se asigna cuando se solicita expresamente (a través de la función new).**
  - **Se utiliza para almacenar elementos dinámicos (cuya memoria se reserva en tiempo de ejecución)**
  - **Los elementos dinámicos son accesibles a través de un tipo básico, el puntero, cuya memoria sí que se reserva en tiempo de compilación ya que lo único que hace es almacenar una dirección de memoria correspondiente al heap**

Organización de la Memoria

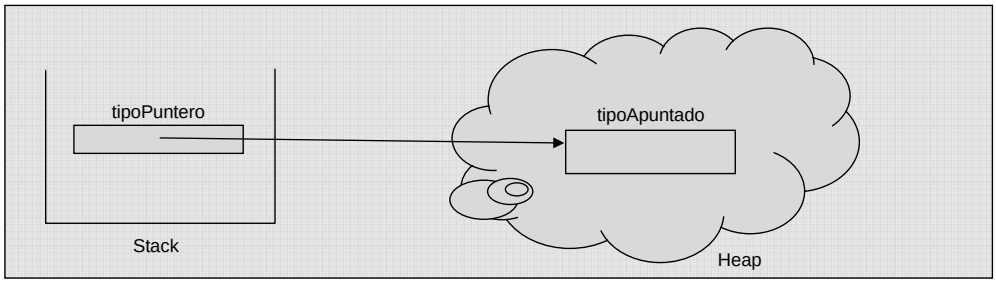


Definición de punteros

- **Punteros en Pascal**
  - **Un puntero es un tipo básico en Pascal (como integer, boolean, char, real, etc.) y como tal ocupa en memoria una cantidad determinada conocida en tiempo de compilación (generalmente 4 bytes)**
  - **Una variable de tipo puntero contiene una dirección en memoria en la cual se almacena una variable de otro tipo**
  - **Las variables de tipo puntero en Pascal son "tipadas" lo que quiere decir que un puntero se declara que apunta a un tipo particular de datos y no puede apuntar a ningún otro**

Definición de punteros

- **Definición de un tipo puntero**
  - **Sintaxis: `type tipoPuntero = ^tipoApuntado`**
  - **Estamos declarando que tipoPuntero es una variable de tipo puntero que apunta a elementos del tipo tipoApuntado**
  - **El símbolo "^" tiene un significado de flecha "→", estamos diciendo que tipoPuntero apunta a una variable de tipoApuntado**
  - **Representación:**



- **Declaración de variables tipo puntero**
  - **Exactamente igual que cualquier otra variable**
  - **Sintaxis: `var variablePuntero = tipoPuntero`**

Definición de punteros

- **Ejemplo de utilización de tipos puntero**
  - **Codigo**

```
type
  punteroEntero = ^integer;
var
  P1: punteroEntero;
  P2: ^integer;
```

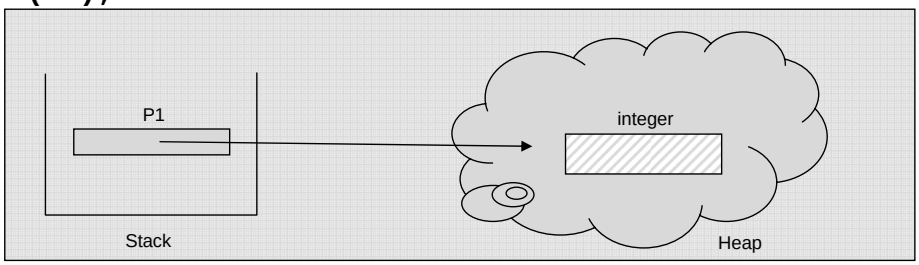
  - **Explicación**
    - **He declarado un tipo punteroEntero indicando que las variables definidas de ese tipo albergarán direcciones de memoria en las cuales vamos a encontrar un entero**
    - **La variable P1 se declara del tipo punteroEntero por lo cual se reserva memoria en la pila (4 bytes) para albergar una dirección de memoria que apuntará a un entero almacenado en el montículo**
    - **Para usar los punteros no es necesario declarar un tipo y pueden usarse directamente, tal y como hace P2. Sin embargo, crear un tipo aumenta la abstracción del código y elimina problemas de comprobación de tipos entre los punteros ¿P1 es del mismo tipo que P2?.**

Definición de punteros

- **Inicialización de punteros**
  - **El proceso de reserva de memoria lo único que hace es marcar esa memoria como reservada, pero no elimina ni inicializa el contenido de la memoria reservada, por lo tanto P1 contendrá inicialmente un valor “basura”**
  - **Utilizar un puntero cuyo contenido es “basura” generalmente tendrá efectos fatales para la ejecución del programa (p. ej. el contenido basura puede representar una dirección de memoria a la cual el usuario no tiene acceso).**
  - **Por eso se recomienda inicializar los punteros a un valor “nulo” que es común para todos (independientemente del tipo en el que se hayan declarado).**
  - **Este valor se representa por la constante “nil” en Pascal. En otros lenguajes como C y Java se representa por “null”**
  - **Ejemplo:**
    - `P1 := nil;`

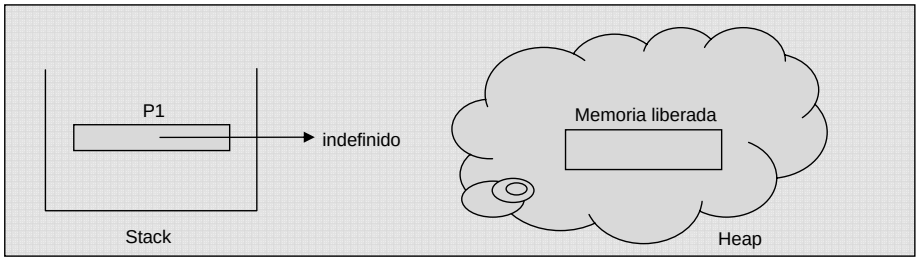
Reserva y destrucción dinámica de memoria

- **Creación de una variable dinámica**
  - **Para reservar memoria a la variable dinámica apuntada por un puntero se utiliza la función "new" definida en la unit "System"**
  - **Sintaxis:**
    - `procedure new (var P:Pointer)`
  - **Funcionamiento**
    - **Reserva memoria para el tipo de la variable apuntada por el puntero que se pasa por parámetro e inicializa el valor del puntero a la dirección de dicha memoria reservada**
    - **En caso de no existir memoria suficiente el procedimiento inicializa la variable a "nil"**
  - **Ejemplo**
    - `new (P1);`



Reserva y destrucción dinámica de memoria

- **Destrucción de una variable dinámica**
  - **Para liberar la memoria a la variable dinámica apuntada por un puntero se utiliza la función "dispose" definida en la unit "System"**
  - **Sintaxis:** `procedure dispose (var P:Pointer)`
  - **Funcionamiento**
    - **Marca la memoria apuntada por el puntero como liberada, de forma que pueda volver a ser utilizada**
    - **¿Qué ocurre con el valor de P después de la ejecución de un dispose?**
      - **El funcionamiento concreto depende de cada compilador**
      - **Lo que sí se suele considerar es que el valor de P no está definido (o queda indefinido) después de la ejecución de un dispose**
      - **Un funcionamiento por ejemplo sería no alterar el contenido de la variable puntero, por lo que esta sigue apuntado a la misma dirección de memoria, pero la memoria se ha marcado como liberada, es decir, la dirección está obsoleta o es una dirección basura**
      - **Acceder a un puntero cuya dirección no está definida generalmente produce un error**
  - **Ejemplo:** `Dispose (P1);`

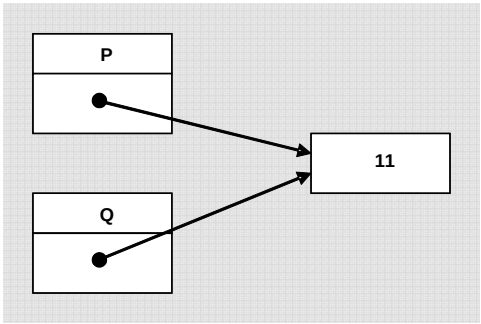


Reserva y destrucción dinámica de memoria

- **Acceso a una variable dinámica**
  - **Se hace a través del operador de dirección "^" que recordemos representa a una flecha**
  - **En las variables estáticas**
    - **Declaración: i: integer;**
    - **Escritura: i := 5**
    - **Lectura: j := i**
  - **En las variables dinámicas**
    - **Declaración: i: ^integer;**
    - **Escritura: i^ := 5;**
    - **Lectura j := i^;**

Asignación y comparación de punteros

- **Sentencias válidas de asignación de punteros**
  - **P := nil**
    - **Le asignamos al puntero el puntero vacío**
  - **P := Q**
    - **Le asignamos al puntero el valor de otro puntero**
    - **Es una operación que hay que realizar con cuidado porque ahora todo cambio en P afectará a Q y viceversa**
    - **Cuál es el resultado de**
      - **Q^ := 4**
      - **P^:= Q**
      - **P^:= 11**
      - **WriteLn (Q^);**
  - **new (P)**
    - **Reservamos memoria y asignamos al puntero la dirección de inicio de esa memoria**



Asignación y comparación de punteros

- **Tipos de comparación de variables dinámicas:**
  - **Identidad:**
    - **Dos variables dinámicas son idénticas si y solo si están en la misma dirección de memoria.**
  - **Igualdad:**
    - **Dos variables dinámicas son iguales si, a pesar de estar en direcciones de memoria distintas, su contenido es el mismo**