Tecnologías de la Información

Universidad de Huelva

Tema 1: PUNTEROS Y ARCHIVOS

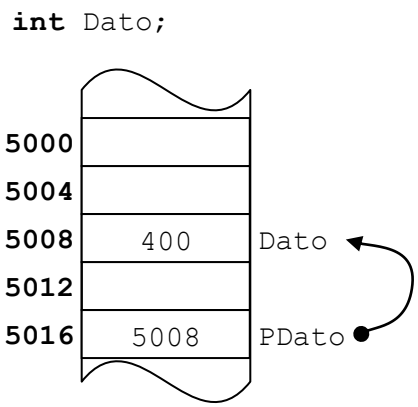
1. Punteros
2. Archivos lógicos y físicos
3. Tipos de archivos
4. Operaciones con archivos
5. Consideraciones sobre el acceso secuencial y directo

1. Punteros

1.1. Concepto de puntero.

Un **puntero**, hablando de manera aproximada, es una variable de tipo entero que contiene un valor numérico que es en realidad la **dirección de memoria** donde está situada otra variable del programa.

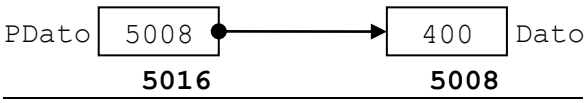
La distinción entre una variable de tipo entero y un puntero es la interpretación semántica del valor que contiene.



Ejemplo 1. *Representación de la memoria interna*

La diferencia entre ambas variable es:

- La variable `Dato` se va a utilizar (su valor) para una operación de tipo aritmético
- La variable `PDato` se utilizará para guardar la dirección de memoria donde se encuentra la variable `Dato`.

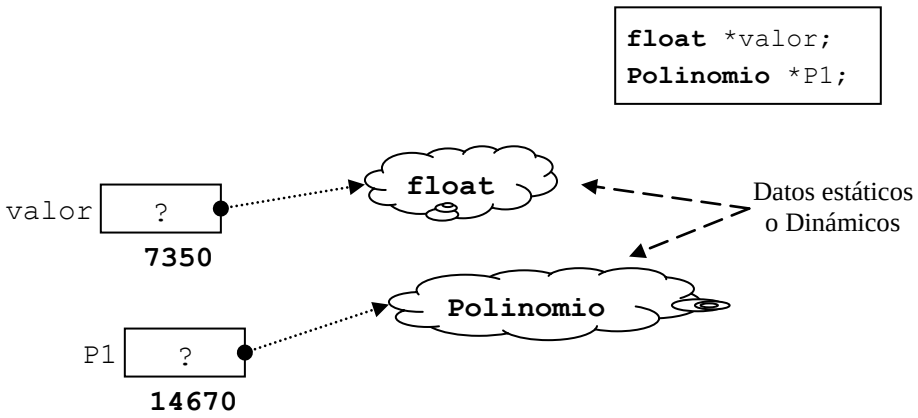


Ejemplo 2. *Representación Gráfica.*

1.2. Declaración de variables de tipo puntero.

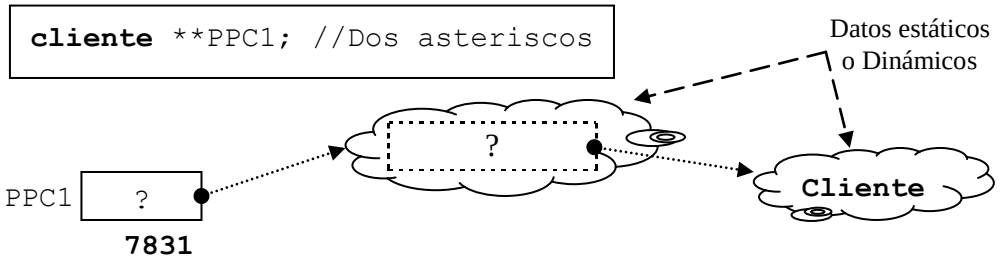
Una variable puntero se declara exactamente igual que una variable normal añadiendo antes del identificador uno o más asteriscos '*****'.

Cuando se declara un puntero **sólo** se está reservando una posición de memoria donde se va a almacenar la dirección de otra variable, **pero no** se reserva memoria para la variable apuntada.



Ejemplo 3. *Representación gráfica de la declaración de dos punteros.*

La zona de memoria donde apunta un puntero **<u>puede ser</u>** además otro puntero que apunte a otra zona de memoria. Esto se conoce como **encadenamiento de punteros**.

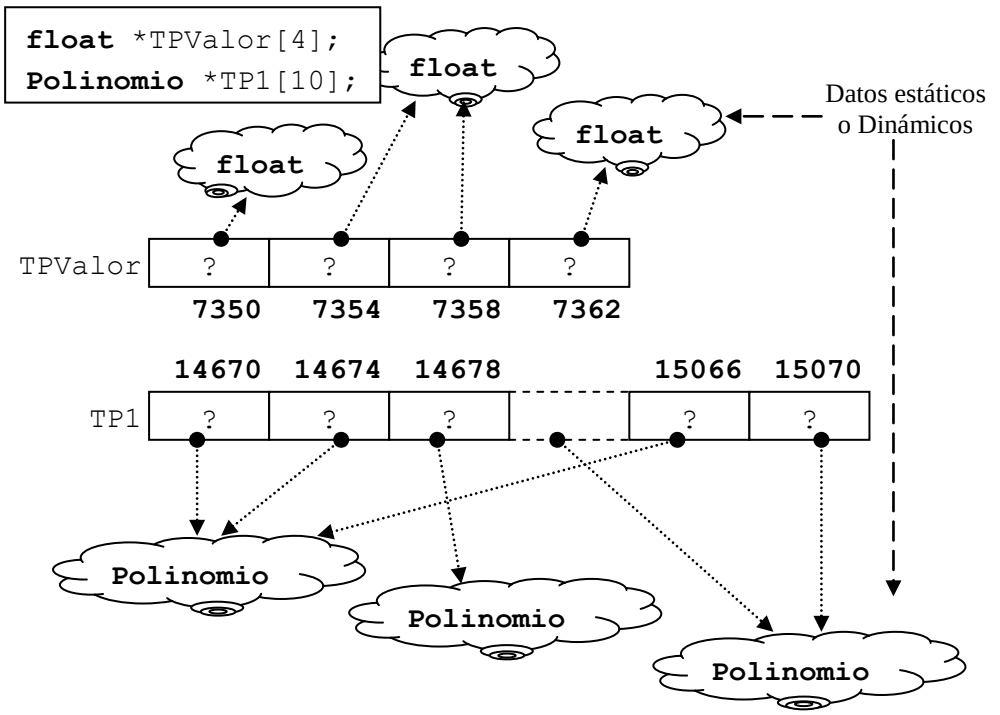


Ejemplo 4. *Representación gráfica del encadenamiento de punteros.*

Este tipo de estructuras se le denominan estructuras de dos o más niveles, ya que para acceder al dato (en este caso un objeto Cliente) se necesita pasar por dos punteros, uno la variable puntero (`PPC1`) y otro el puntero que es apuntado por `PPC1`. Existen estructuras de más de dos niveles.

1.3. Declaración de tablas de punteros.

Se declara una tabla de punteros añadiendo a la declaración de un puntero el tamaño de cada dimensión. Hay que tener en cuenta que cuando se declara una tabla de punteros sólo se reserva memoria para la tabla y no para los datos a los que apuntará cada puntero.



Ejemplo 5. *Representación gráfica de dos tablas de punteros.*

En el ejemplo, se está declarando una tabla de cuatro punteros a reales (**no cuatro reales**) y una tabla de 10 punteros a objetos Polinomio (**no 10 objetos Polinomio**).

Se puede dar el caso de que varios punteros apunten a la misma variable. Para ello el valor de la dirección de memoria será el mismo para ambos punteros. Tampoco es infrecuente encontrar tablas de punteros de más de un nivel como por ejemplo:

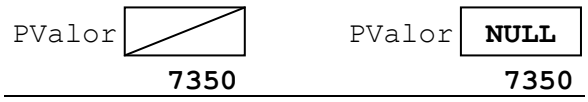
```
cliente **TPC1[10]; //Cada puntero de esta tabla apuntará a
                    //otro puntero que a su vez apuntará a
                    //un objeto cliente
```

1.4. Operaciones con punteros.

1.4.1. Inicialización.

Las variables puntero han de ser inicializadas con direcciones de memoria o con un valor especial que indica una dirección nula, la constante **NULL**.

```
float *PValor=NULL;
Polinomio **PPolinomio=NULL;
```



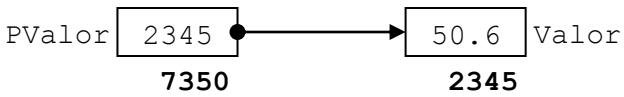
Ejemplo 6. *Representación gráfica*

1.4.2. Asignación.

Existe el ***operador de referencia*** '**&**' que nos devuelve la dirección de una variable si éste se coloca justo delante del identificador de la variable.

```
float *PValor=NULL, Valor=50.6;
```


```
Pvalor = &Valor;
//&Valor devuelve la
//dirección de Valor, 2345
```



Ejemplo 7. *Ejemplo de declaración, inicialización y asignación de variables puntero.*

1.4.3. Acceso.

En todo momento se puede acceder a dos tipos de información: al **propio valor** de la variable puntero (mediante el identificador de la misma) y a la **información a la que apunta** mediante el operador denominado ***desreferencia*** '*****'.

Utilizando el ejemplo anterior:

```
float *PValor=NULL, Valor=50.6;
float *Copia_de_PValor=NULL, Suma;

Pvalor = &Valor;
Copia_de_Pvalor = PValor;  /* PValor devuelve su contenido, es
        decir 2345, que es la dirección de la variable Valor */
Suma = *Pvalor + 10;       //Suma contiene 60.6
```

Nota: Dependiendo del lugar donde aparece, el símbolo '*****' tiene un significado distinto.

```
float Producto = *Pvalor * *PValor; //Contiene 50.6*50.6=2560.36
```

Para acceder a los campos (métodos) de una estructura (clase) apuntados por una variable puntero se puede utilizar el operador punto '**.**' .

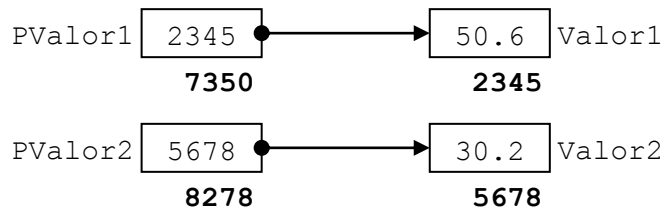
```
Cliente *PCli=NULL, Cli;
PCli = &Cli;
(*PCli).Edad = 20;
cout << (*PCli).Nombre;
```

El operador '**->**' simplifica la escritura de programas. Como podemos ver en el siguiente ejemplo, el operador '**->**' nos indica que el nombre del cliente está apuntado y accedido a través el puntero `PCli`.

```
Cliente *PCli=NULL, Cli;
PCli = &Cli;
PCli->Edad = 20;
cout << PCli->Nombre;
```

1.4.4. Comparación

Una variable puntero puede ser comparada con otra variable puntero o con la constante **NULL**; aunque todos los operadores relacionales están permitidos, sólo los operadores != y == tienen una verdadera utilidad práctica.



Ejemplo 8. *Representación gráfica del ejemplo*

```
float *PValor1 = NULL, *PValor2 = NULL, Valor1 = 50.6, Valor2 = 30.2;
PValor1 = &Valor1;
PValor2 = &Valor2;
if (PValor1<PValor2)  //Comparación 1ª
  cout << "Valor1 está situada en la memoria antes de Valor2";
else
  cout << "Valor2 está situada en la memoria antes o en la misma
          posición de Valor1";
if (*PValor1 < *PValor2) //Comparación 2ª
  cout << *PValor2 << " es mayor que " << *PValor1;
else
  cout << *PValor2 << " es menor o igual que " << *PValor1;
```

La salida por pantalla del código del ejemplo anterior es el siguiente:

- Comparación 1ª:

  `Valor1 está situada en la memoria antes de Valor2`

- Comparación 2ª

  `30.2 es menor o igual que 50.6`

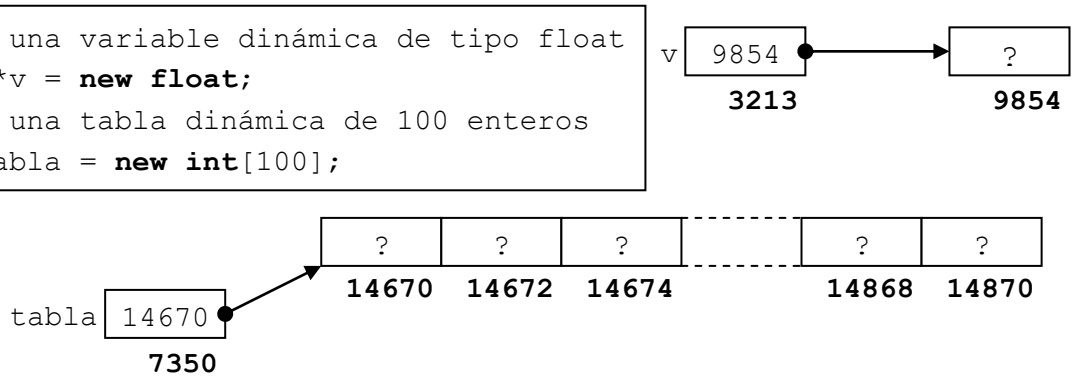
La primera condición es verdadera ya que el contenido de `PValor1` es menor que el contenido de `PValor2` (`2345 < 5678`) y la segunda condición es falsa (`50.6 < 30.2`).

1.5. Creación de variables dinámicas.

Mediante el uso de punteros se nos permite disponer de nuevas variables en tiempo de ejecución. Las variables dinámicas tienen como identificador la dirección de memoria donde están situadas. El programador tiene la responsabilidad de solicitar la creación y eliminación de variables dinámicas.

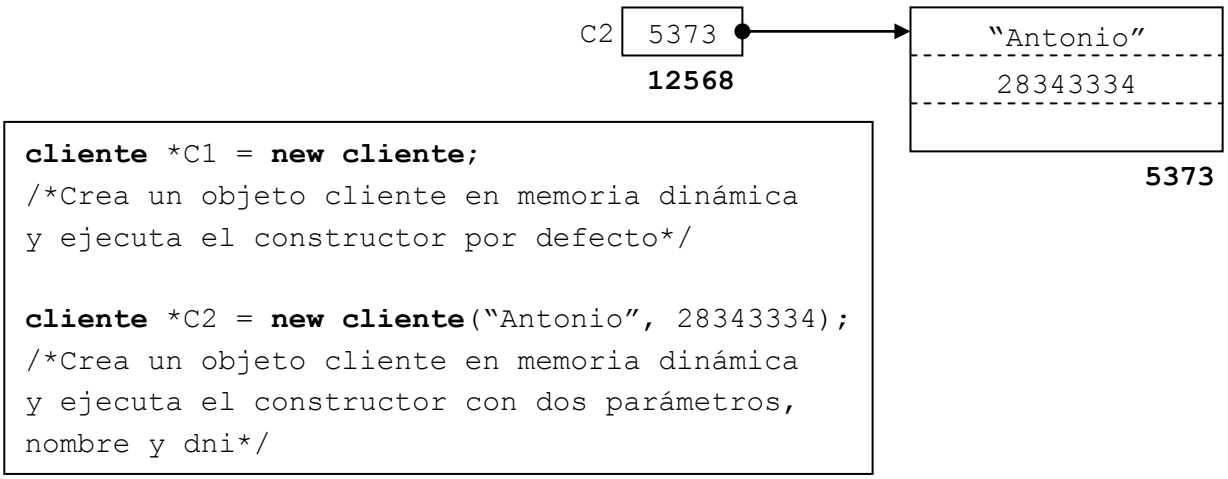
En el lenguaje C++ se cuenta con el operador **new** para solicitar el uso de memoria dinámica. Posee la capacidad de ejecutar los constructores definidos en la clase del objeto que se instancia. Además permite traspasar valores a los parámetros definidos en los constructores de la clase.

```
//Crea una variable dinámica de tipo float
float *v = new float;
//Crea una tabla dinámica de 100 enteros
int *tabla = new int[100];
```



Ejemplo 9. *Ejemplos de creación de variables dinámicas y de tablas con memoria dinámica*

Todas las funciones y operadores que reservan memoria, devuelven o bien la dirección de memoria reservada o bien la constante **NULL** para indicar que el S.O. no ha encontrado ninguna zona de memoria con el tamaño que ha sido solicitado.



Ejemplo 10. *Ejemplo en C++ de creación de estructuras de datos en memoria dinámica.*

1.6. Eliminación de variables dinámicas.

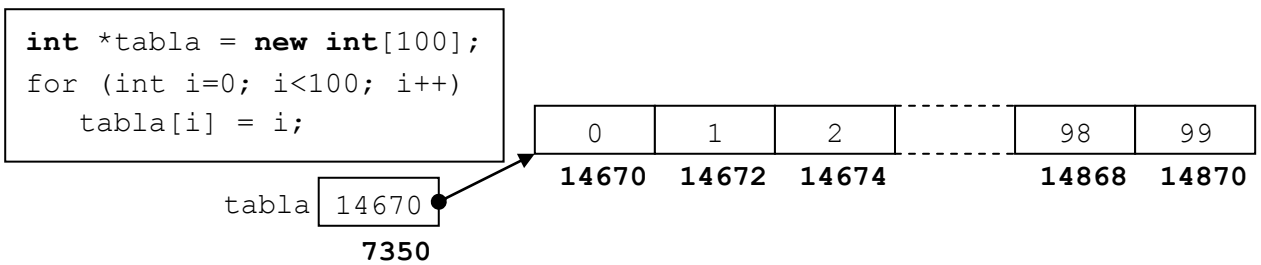
C++ permite la liberación de memoria dinámica con el operador **delete**. Posee la capacidad de ejecutar el destructor definido en la clase del objeto que se destruye.

```
delete v;

delete [] tabla;
// para liberar un array creado dinámicamente con new
// invoca al destructor de cada elemento del array
```

1.7. Aritmética de punteros.

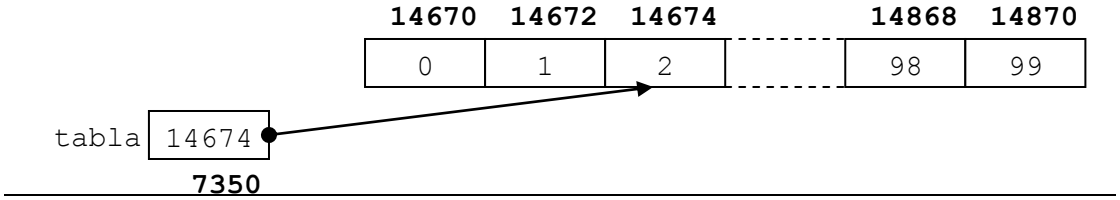
Como toda variable numérica, las variables de tipo puntero permiten realizar ciertas operaciones aritméticas sobre la dirección de memoria que almacenan. Las operaciones normales son la suma y resta de posiciones de memoria.



Ejemplo 11. *Creación e inicialización de una tabla dinámica.*

Una vez que se agrega un valor a un puntero, puede que se produzcan resultados inesperados, efectos colaterales e incluso la pérdida de información.

```
tabla++;        //contiene 14672 el siguiente elemento
cout << *tabla; //escribe 1 ( *tabla es igual que tabla[0])
tabla++;        //contiene 14674 el tercer elemento de la tabla
cout << *tabla; //escribe 2
```



Ejemplo 12. *Ejemplo que muestra la utilización de la aritmética de punteros.*

El estado anterior de la tabla es tal que el primer elemento de la tabla es el valor **2** y no el **0** y además el número de elementos de la tabla es de **98** elementos y no de **100**.

```
cout << tabla[0];  //escribe 2
cout << tabla[-1]; //escribe 1
```

Este funcionamiento es **independiente** del tipo de datos de la variable puntero. La agregación de valores a una variable puntero **incrementa** su contenido en el **mismo número** de bytes que ocupa el tipo de la variable puntero.