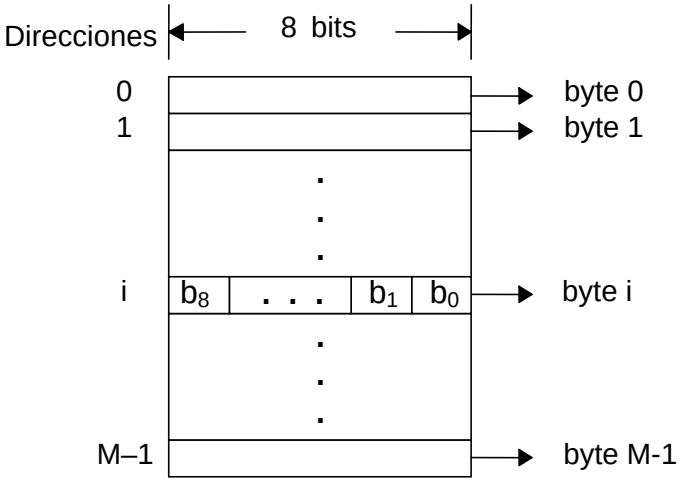
UNIVERSIDAD DE CARABOBO
FACULTAD EXPERIMENTAL DE CIENCIAS Y TECNOLOGÍA
DEPARTAMENTO DE COMPUTACIÓN
ALGORITMOS Y PROGRAMACIÓN II

# APUNTADORES

## La memoria del computador

La memoria principal de un computador consiste de un gran número de posiciones secuenciales de almacenamiento denominadas *celdas de memoria*, cada una de las cuales puede almacenar una porción de información. En los últimos años, la mayoría de los fabricantes de computadores han establecido como estándar una celda de n=8 *bits* que se denomina *byte*. Cada *byte* de memoria tiene un número único asociado, denominado *dirección de memoria*. Las direcciones de memoria en un computador cuya memoria consiste de M *bytes* van desde 0 hasta M-1.




## Las variables y la memoria

Al declarar una variable en un lenguaje de programación, implícitamente se está dando la orden de que se reserve un espacio contiguo de memoria lo suficientemente grande para guardar un valor del tipo especificado en la declaración, es decir, una cantidad fija de bytes. A este espacio de memoria se hace referencia a través del nombre de la variable. Internamente, el compilador asocia a cada nombre de variable la dirección del primer byte del bloque de memoria que utiliza esa variable.

## Apuntadores

Un apuntador es una variable que contiene una dirección de memoria. Normalmente, esa dirección es la posición en memoria de otra variable. Si una variable contiene la dirección de otra variable, entonces se dice que la primera variable apunta a la segunda. También se dice que un apuntador hace referencia a otra variable.

El tipo de dato “apuntador” es un tipo de dato primitivo. La sintaxis para la declaración de una variable de tipo Apuntador tiene la siguiente estructura:

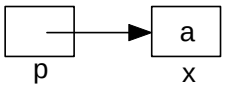
```
var ptr = pointer to T0
```

Donde:

- “ptr” es el identificador de la variable.
- “$T_0$” es el tipo de dato al que “ptr” hace referencia. “$T_0$” puede ser un tipo primitivo o estructurado.

Nótese que, aunque un apuntador es una dirección de memoria y ésta es un número entero, se mantiene una diferencia de tipos entre un valor entero y una dirección de memoria. Además, se limitan los apuntadores a que sólo apunten a datos de algún tipo específico, indicado en la declaración.

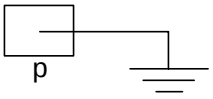
Gráficamente, las variables se representan como rectángulos que llevan a un lado su nombre y adentro su valor. En el caso de los apuntadores, su valor se representa como una flecha que parte desde adentro del rectángulo y llega hasta el borde de otro rectángulo correspondiente a otra variable. Por ejemplo, si tenemos un apuntador “p” que apunta a una variable “x” cuyo valor es “a”, se dibuja así:




## La dirección nula

Dado que una variable siempre tiene algún valor, un apuntador siempre apunta a algún espacio de memoria aun cuando no se le haya especificado que apunte a alguna parte. Cuando se desee que un apuntador no apunte a nada, se utiliza una dirección especial llamada “la dirección nula” o “el apuntador nulo”. En los algoritmos, llamaremos a este valor “NULL”.

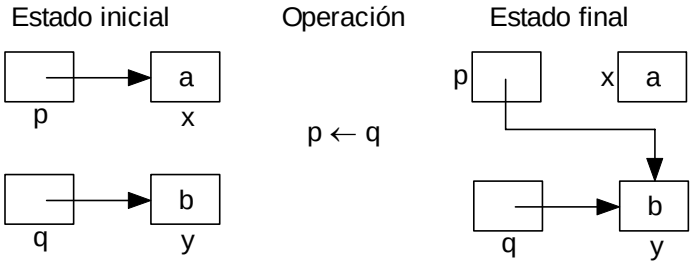
De manera gráfica, cuando un apuntador “p” tenga el valor “NULL”, se representa de la siguiente manera:




## Asignación

A una variable de tipo Apuntador se le puede asignar otra variable de tipo Apuntador siempre y cuando ambas estén declaradas como apuntadores al mismo tipo.

Sean “p” y “q” dos variables de tipo Apuntador que hacen referencia a dos variables “x” y “y”, cuyos valores son “a” y “b”, respectivamente. La operación de asignar “q” a “p” produce el siguiente resultado:

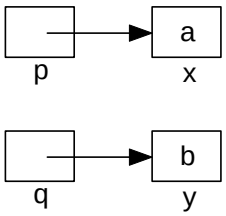



También se le puede asignar a un apuntador el valor “NULL”.

## Pruebas de igualdad / desigualdad

Se pueden aplicar operaciones de igualdad (=) y desigualdad (≠) entre apuntadores para determinar si contienen la misma dirección, o dicho de otra forma, si apuntan a la misma variable, retornando un valor booleano.

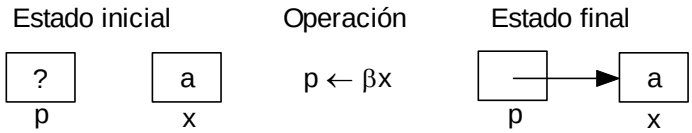
Sean “p” y “q” dos variables de tipo Apuntador que hacen referencia a dos variables “x” y “y”, cuyos valores son “a” y “b”, respectivamente:




Entonces, se cumple que (p = q) = *false* y (p ≠ q) = *true*

## Referenciación (β):

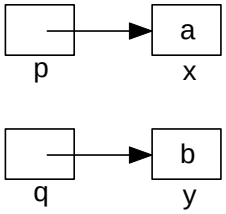
La operación de referenciación consiste en obtener la dirección de memoria correspondiente a una variable. Esta operación se representa con el operador **β** (beta). Si “x” es una variable de tipo $T_0$, entonces “βx” retorna la dirección de memoria correspondiente a la variable “x”. Al asignar este resultado a un apuntador “p” ocurre lo siguiente:




## Desreferenciación (↑):

La operación de desreferenciación consiste en obtener el valor de la variable a la cual apunta un apuntador. Esta operación se representa con el operador "↑". Si p es un apuntador que hace referencia a objetos de tipo $T_0$, entonces "p↑" retorna el valor almacenado en la dirección de memoria contenida en "p".

Sean "p" y "q" dos variables de tipo Apuntador que hacen referencia a dos variables "x" y "y" de tipo $T_0$, cuyos valores son "a" y "b", respectivamente:



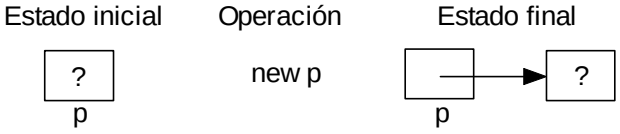

Entonces se tiene que p↑ = a y q↑ = b. Sobre p↑ y q↑ se aplican todas las operaciones asociadas al tipo $T_0$.

Desreferenciar un apuntador sólo es válido cuando el mismo ha sido inicializado.

## Creación dinámica de objetos

Los apuntadores se utilizan principalmente para la creación dinámica de variables u objetos durante la ejecución de un algoritmo. Los objetos creados de forma dinámica existen en memoria hasta que se decida eliminarlos; en contraste con los creados de forma estática, que existen sólo en la función en la que fueron declarados.

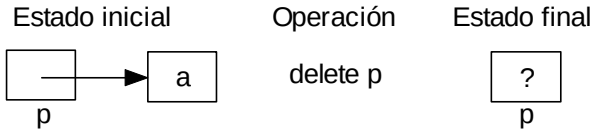
Para crear un nuevo objeto de forma dinámica se define el operador "new", el cual se aplica a un apuntador. Dado un apuntador "p" que hace referencia a objetos de tipo "$T_0$", la instrucción "new p" crea una nueva variable u objeto (sin nombre) de tipo $T_0$ y hace que "p" apunte a ella, como se muestra en el gráfico:



## Eliminación de objetos

Un objeto creado de forma dinámica se debe eliminar cuando ya no se necesite, con el objetivo de liberar la memoria ocupada por dicho objeto para un uso posterior.

Para este fin se define el operador “delete”, el cual se aplica a un apuntador. Si “p” es un apuntador que hace referencia a un objeto creado de forma dinámica (a través de “p” o cualquier otro apuntador del mismo tipo), entonces la instrucción “delete p” elimina dicho objeto liberando la memoria que ocupaba, como se muestra en el gráfico:




Nótese que después de eliminar el objeto referenciado por “p”, la dirección contenida en “p” deja de ser válida.

Algunos entornos de ejecución disponen de un sistema llamado “recolector de basura”, el cual libera automáticamente todos los objetos creados dinámicamente que ya no sean referenciados por ningún apuntador en determinados momentos durante la ejecución de un programa, lo cual permite que el programador no se tenga que preocupar por liberar la memoria reservada para cada objeto creado de forma dinámica.