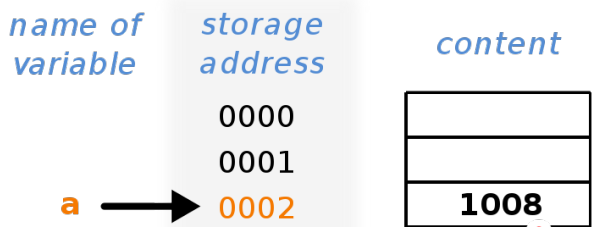
# Punteros

March 25, 2020

## ¿Qué son los punteros?

Cualquier cosa que almacenamos en la computadora va a parar a la memoria. La memoria está formada por "celdas" que guardan parte de la información. Estas celdas están identificadas con un *storage address* como se puede ver en la figura.

Un puntero es un objeto que lo que hace es señalar a alguna de estas celdas, o posiciones de memoria. Siguiendo con la imagen cuando yo defino el puntero "a", lo que hace este puntero es apuntar a la posición de memoria con el *storage address* 0002. Si yo quisiera saber qué es lo que hay en la posición de memoria a la que apunta el puntero "a", simplemente tengo que escribir `*a`. El asterisco se traduce al español como "el contenido de la posición de memoria correspondiente al puntero".




Analizando un poco la figura, vemos que el puntero `a`apunta a la posición de memoria 0002. Si queremos ver el contenido de esta posición de memoria escribimos `*a`y esto nos muestra que el contenido de esa posición de memoria es el número 1008.

Si queremos cambiar el contenido de esa posición de memoria, por ejemplo por el número 23, simplemente escribimos `*a = 23`. De este modo, al utilizar punteros, uno opera directamente sobre la memoria de la computadora, cosa que es imposible en lenguajes de alto nivel como Python, y que nos va a resultar muy útil en el futuro cercano.

## ¿Para qué queremos los punteros?

Habrán visto que cuando programamos en C usamos funciones, por ejemplo `int main()`,etc. Todas las funciones que definimos son del tipo int, float, etc. Esto significa que la fución devuelve algo que es un entero o un número de punto flotante. Ahora, ¿qué pasa si yo quiero que una sola función me devuelva dos números? Supongamos que hago una función que calcula la energía de algún sistema, y yo quiero que me devuelva por un lado la energía cinética y por otro la potencial, ¿cómo hago? Se podrían hacer dos funciones `float cinetica() y float potencial()`donde cada una devuelva el valor de interés, pero eso no es muy práctico (o elegante). En esta situación vienen bien los punteros. Si yo defino dos variables `T` y `V` (energía cinética y potencial) no tengo forma de que una sola función me calcule los dos valores, ya que la función va a devolver un sólo número de punto flotante. Pero si yo defino `T` y `V` como punteros en una sóla función puedo modificar a los dos, ya que haciendo `*T= algo`y `*V= otra cosa`ya estoy directamente modificando el contenido de la posición de memoria correspondiente a T y a V. Ni siquiera necesito que la función tenga `return T; o return V;`Al modificar el contenido de un puntero dentro de una función, este queda modificado para siempre, lo que nos permite modificar todas las variables que querramos dentro de una misma función. Esto hace que los punteros sean muy útiles. Por ejemplo, defino una función cualquiera que tenga como inputs a T y a V.

```
int ejemplo(float *T, float *V)
{
```

```
*T = 62;
*V = 18;
return 0;
}
```

Luego de correr esta función, cuyo return es 0, por lo que pareciera que no hace nada, los valores de T y V fueron modificados por lo que yo escribí en la función. Esto va a ser muy útil para todas las funciones que hagan, donde en general van a querer calcular energías, velocidades, presión, temperatura, etc. Si no usan punteros, deberían hacer una función para cada cosa, lo que además de ser poco elegante, haría que el programa se vuelva pesado y lento rápidamente.

Otra cosa para la que resultan muy útiles los punteros es para trabajar con vectores y matrices. Habrán visto que cuando definen variables en C pueden ser `int o float`, pero no hay una opción `vector o matrix`. ¿Entonces cómo creamos vectores? De nuevo nos salvan los punteros. Para hacer vectores uno hace lo siguiente:

```
float *v;
v = (float*) malloc(N * sizeof(float));
```

Esto que a simple vista no se entiende significa lo siguiente. Defino el puntero v, que apuntará a alguna posición de memoria libre. Pero como yo quiero guardar un vector de N componentes, voy a necesitar N veces el tamaño de un float. Entonces aclaro que quiero que mi puntero me guarde las N-1 posiciones de memoria siguientes a la correspondiente a v. Esto es lo que hace la función malloc(). Como los números int y float ocupan distinta cantidad de memoria, yo le tengo que aclarar a la computadora qué tipo de variable estoy usando, malloc(N*sizeof(float)) significa, guardame las posiciones de memoria necesarias para llenar con N floats.

Con estas dos lineas de código se construye un puntero v que apunta a una posición de memoria determinada, y además me guarda las posiciones de memoria siguientes para que yo pueda escribir mi vector de N componentes. Ejemplo para que se entienda:

En la Figura vemos que el puntero "a" apunta a la posición de memoria 0002. Ahora supongamos que yo quiero crear un vector de 3 componentes, entonces hago lo siguiente:

```
float *a;
a = (float*) malloc(3 * sizeof(float));
```

Ahora "a" apunta a la posición 0002 igual que antes, pero también tengo disponibles para mi vector las posiciones 0003 y 0004. Si yo quiero escribir el vector (v,w,z) simplemente hago:

```
*a = v;
*(a + 1) = w;
*(a + 2) = z;
```

Si yo quiero apuntar a la posición 0003 lo que tengo que hacer es apuntar primero a la posición 0002 (la correspondiente al puntero a) y después correrme un lugar más (a+1).

Si yo defino un vector, siempre la primera componente va corresponderse con el puntero, o sea `*a = primera componer` Si ahora el vector que hice no me gusta y lo quiero cambiar por (v,x,z) simplemente escribo `*(a + 1) = x;`Esto me cambia esa componente y no le hace nada a las otras.

A la hora de hacer matrices hacemos exactamente lo mismo. En vez de trabajar con arrays de dos dimensiones, como hacíamos en python, es más fácil y eficiente convertir las matrices en arrays unidimensionales, lo que uno suele llamar vector. Entonces la matriz

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

se convierte en el vector (1,2,3,4,5,6,7,8,9). Definimos entonces el puntero:

```
float *M;
M = (float*) malloc(N*N * sizeof(float));
```

Entonces tenemos un vector de $N^2$componentes, las necesarias para escribir la matriz. Como estamos acostumbrados a referirnos a los elementos de matriz como $M_{ij}$ en lenguaje de punteros vamos a referirnos a ellos como `*(M + N*i + j)`.

Con esto ya les alcanza para empezar a crear y manipular punteros, vectores y matrices. Les mandamos un ejemplo de un programita para transponer matrices de 2x2 usando punteros. Para practicar, armen programas que hagan productos escalares, productos vectoriales y productos de matrices, basándose en el

ejemplo. Una vez que le agarren la mano a crear punteros y a manipular correctamente las componentes de vectores y matrices van a ver que lo demás no es mucha ciencia.