Resumen de *Mastering Algorithms in C*

Martin Noblia

March 15, 2016

Pointers(Punteros)

En C para cualquier *type* `T` existe un correspondiente *type* que contiene la direccion de memoria en donde el objeto de type `T` se guardan. Por eso a estas variables se las llama *Punteros* ya que es como si apuntaran a. Por ello los punteros son simples variables que que guardan la direccion de memoria donde esta guardada ciertos datos, en lugar de guardar los datos en si mismos.

Una de las mejores formas para comunicar y entender la informacion de los punteros es dibujar diagramas, en lugar de listar las direcciones de memoria dibujamos flechas que unen una direccion con otra, cuando el puntero no apunta a nada(o sea a NULL) se dibuja una doble linea(Como si fuera masa o tierra en electronica).

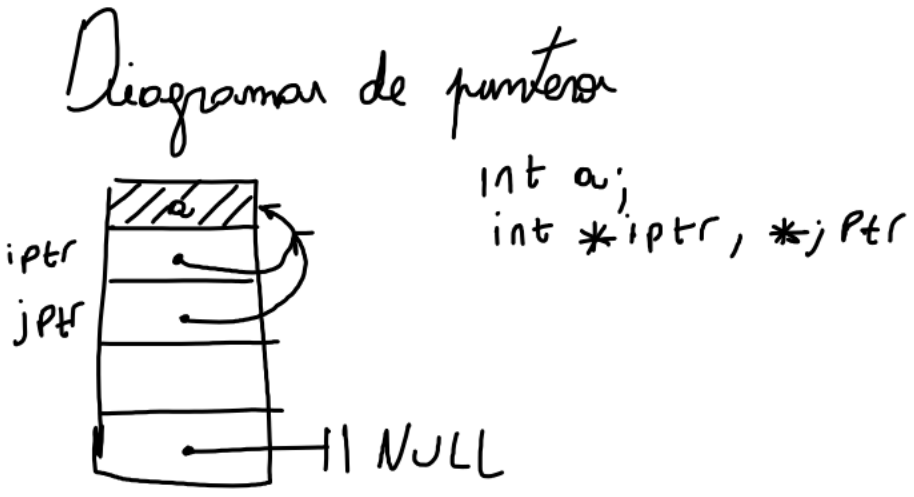

Figure 1: Diagramas de punteros

Cuando declaramos un puntero cierto espacio en memoria es guardado para el. Es importante notar que solo se guarda espacio para el puntero(variable) y no hacia donde apunta. Existe dos maneras de guardar(*allocate*) memoria para nuestros datos: una es declarando una variable, la otra dinamicamente utilizando `malloc()` o `realloc()` como sabemos cuando declaramos una variable, su *type* le dice al compilador cuanta memoria guardar, esta memoria se guarda automaticamente pero puede que no dure toda la vida del programa, esto es importante cuando tratamos con punteros a *variables automaticas*, estas variables son aquellas cuyo almacenamiento es *allocate* y *deallocate* cuando entran o dejan una funcion o un bloque delimitador.

Por ejemplo ya que en el siguiente caso `iptr` es asociado a una variable automatica `a` en la siguiente funcion `f` entonces `iptr` se convierte en un puntero colgado(*dangling pointer*)

```
int f(int **iptr)
{
    int a = 10;
    *iptr = &a;

    return 0;
}
```

O sea que le estamos dando la direccion de memoria de una variable que va a desaparecer!!!

Agrupamiento de datos en `C`

Uno de los usos mas frecuentes de los punteros en `C` es el de referenciar datos `C` soporta dos tipos de agrupamiento: *structs* y *arrays*. Como sabemos las *structs* son agrupamientos de datos heterogéneos que pueden ser tratados juntos como un solo tipo de dato. Los punteros a *structs* son una parte importante para construir los tipos de datos mas conocidos.

Arrays

Los *arrays* son secuencia de elementos homogeneos distribuidos consecutivemente en memoria. Como sabemos en `C` los *arrays* estan muy ligados a los punteros ya que cuando una expresion de *array* ocurre el lenguage automaticamente convierte a este en un puntero que apunta al primer elemento del *array*. O sea que existe una relacion entre: `a[i]` y `*(a + i)` que tambien vale para matrices

```
*(*(a + i) + j)
```

Punteros como parametro de funciones

Los punteros son utilizados para pasar a las funciones parametros por referencia en lugar de por valor(ya que si por ejemplo tenemos un array grande este primero se copia y luego se pasa a la funcion).