Tarea 20

Elementos de ciencias de la computación

Tulio Muñoz Magaña

October 31, 2020

Reporte de lectura de los capítulos I y II del texto dado.

Las variables en general son ”objetos” que se componen esencialmente de 2 valores, el valor por la izquierda (lvalue) y el valor por la derecha (rvalue). El valor por la izquierda contiene la dirección donde está almacenada la variable, el valor por la derecha contiene la ”información” de la variable, es decir, lo que está almacenado en la dirección dicha. Estos nombres provienen de la convención de asignación de valores en C, si escribimos `a = b;` la variable `a` representa su dirección ya que está a la izquierda, y la variable `b` representa su contenido porque está a la derecha. Los rvalues no se pueden colocar a la izquierda de una asignación.

La cantidad de espacio necesaria para almacenar una dirección de memoria ha variado a lo largo de la historia, y depende de la cantidad de memoria que contenga la máquina, las variables que almacenan direcciones de memoria son llamadas apuntadores. Para declarar una variable apuntador, colocamos un asterisco, además que definimos el tipo de dato al que apuntará, por ejemplo `int* pt;`. Si un apuntador es declarado fuera de una función, se le asigna el valor ”NULL”, que es el apntador que no apunta a ningún lado. Este valor puede no ser una trama de ceros, esto depende de la máquina, por lo que por lo general se define una macro con ese nombre para hacer referencia a él.

Para obtener la dirección de una variable ordinaria, utilizamos el operador unitario `&` antes de la variable, es decir de la siguiente forma: `&a`. Otro

operador relacionado con los apuntadores es el * como operador de desreferencia, lo usamos para modificar u obtener el valor que está almacenado en el lugar al que estamos apuntando, por ejemplo `*ap = 2;` cambia el valor del contenido de la dirección a la que apunta `ap` a 2.

La razón por la que indicamos el tipo de dato al que se apuntará al definir un apuntador, es para que el programa sepa la cantidad de bytes que tiene que copiar. Además, nos permite definir la "aritmética de apuntadores", donde el puntero `pt + i` nos proporcionará la i-ésima dirección sucesiva a `ap`, considerando el tipo de dato al que apunta. Asimismo, también funcionan las operaciones `ptr++` y `++ptr`, la única diferencia entre estas dos operaciones es que son diferentes en algunas situaciones relativas al orden de operadores. Por ejemplo si escribo `a = *ptr++;` se asignará a la variable `a` el valor desreferenciado de `ptr` y luego se avanzará una posición en el apuntador, pero si escribo `a = *(++ptr);` primero se aumentará en la posición del apuntador, y luego se asignará a la variable `a` el valor desreferenciado de la nueva posición.

En C existe una convención para declarar un apuntador que apunta al inicio de un arreglo, esta convención es asignando el apuntador de la siguiente forma, si el arreglo es `arr` y el apuntador es `ap`, entonces al escribir `ap = arr;` apuntamos con `ap` al inicio del arreglo, es decir, es lo mismo que escribir `ap = &arr[0];`. Es importante notar, sin embargo, que `arr` no es un apuntador variable al que le podamos cambiar el valor, lo podemos pensar como un apuntador constante que siempre apunta al mismo lugar, y cuya dirección puede ser asignada a un apuntador ordinario.

En ciertas ocasiones, puede haber problema al asignar el valor de un apuntador a otro que apunte a un tipo de dato distinto al primero, de igual manera que podría haber problema al asignar un entero a una variable de tipo caracter, para resolver este problema existe el tipo de apuntador `void*`, los apuntadores de este tipo son genéricos y pueden ser comparados con cualquier tipo de apuntador, mientras que por ejemplo un apuntador de tipo `int*` no puede ser comparado con uno de tipo `float*`.

El código del inciso b está en el proyecto "Tarea 20", las capturas del funcionamiento son captura1.png y captura2.png, el código efectivamente muestra la trama de bits predicha al ingresar el número 0.15625, y cambia el

primer bit a 1 al ingresar el número -0.15625.