So why use pointers? …

(From http://www.codingunit.com/c-tutorial-how-to-use-pointers)

To make full use of the C Programming language, you have to have a very good understanding of pointers. For most people it will take some time to fully understand pointers. **So be patient**. You have to learn pointers because they are used everywhere in the C language. Once you master the use of pointers, you will use them everywhere.

Pointers are used (in the C language) in three different ways:

- To allocate dynamic data memory.
- To pass and handle variable parameters passed to functions.
- To access information stored in arrays. (Especially if you work with link lists).

Pointers are also used by experienced programmers to **make the code more efficient and thus faster**. With an array you have to declare its maximum size (for every dimension) at the beginning. Let's say you create an array that can hold a maximum of twenty megabytes. When the array is declared, the twenty megabytes is claimed. Now this time you have only data for ten megabytes. (And next time it could be fifteen megabytes or five megabytes). So in this case ten megabytes of memory is wasted, because only ten megabytes from the twenty is used. This is where pointers come in. With pointers, you can create ***dynamic*** data structures. Instead of claiming the memory up-front, the memory is allocated (from the heap) while the program is running. So the exact amount of memory is claimed and **there is no waste**. Even better, memory not used will be returned to the heap. (Freed memory can be used for other programs).

Look at how short the code is for the following two functions using Pointers!

```
/* Example 1
============================================================
     String Concatenation via Pointers – stores s1 & s2 into s3
============================================================*/
void concat (char *s1, char *s2, char *s3)
{    while ( *s1 != '\0')
        *s3++ = *s1++;                // Put characters in s1 into s3, excluding null

    while ( *s2 != '\0')
        *s3++ = *s2++;                // Put characters in s2 into s3, excluding null

    *s3 = '\0';                       // add null at the end of s3
}

/* Example 2
============================================================
     String Copy Function using Pointers – copies s2 into s1
============================================================*/
void scopy(char *s1, char *s2)
{  while (*s1++ = *s2++);             // puts characters in s2 into s1 until null is reached
                                      // while (Nonzero) ... returns 0 when hitting NULL
}
```