CSCU9V4 Pract. 6: It's *not rude to point!

Introduction

First a little Q&A to better understand pointers in C.

Q: I have yet to understand pointer arithmetic. If a pointer is an address, does that mean that an expression like `p+j` adds j to the address stored in p?

A: No. Integers used in pointer arithmetic are scaled depending on the type of the pointer. If p is type `int*`, for example, then `p+j` adds 4 x `j` to p (assuming `int` values are 4 bytes). If p has type `double*`, then `p+j` adds 8 x `j` (for 8 byte doubles).

Q: When writing a loop to process an array, is it better to use subscripting or pointer arithmetic?

A: There's no easy answer to this question, which often depends on the machine a compiler. In earlier days, absolutely. Today's machines and compilers are optimized, and sometimes even do better with subscripting! The bottom line is learn both, and use the one that best suits the occasion or preference.

Q: I read somewhere that `i[a]` is the same as `a[i]`. Is this true?

A: Oddly, yes! The compiler treats `i[a]` as `*(i+a)`, which is the same as `*(a+i)`. (Note that pointer addition, like regular addition, is commutative.) Taking this one step further, we know from lecture that `*(a+i)` is equivalent to `a[i]`! Please refrain entirely from this convention, that is, unless there are plans to enter the next Obfustacted C contest. Welcome to C – who is buying my next coffee?

Pen and Paper

Attempt the following **on your own first**, before consulting with others.

1. Suppose that the following declarations are in effect:

```
int a[] = {5, 15, 34, 54, 14, 2, 52, 72};
int p = &a[1], q = &a[5];
```

   a) What is the value of `*(p+3)`?
   b) What is the value of `*(q-3)`?
   c) What is the value of `q-p`?
   d) Is the condition `p<q` true or false?
   e) Is the condition `*p<*q` true or false?

2. [Challenging] Suppose that high, low, and middle are all pointer variables of the same type, and that low and high point to elements of an array. Why is the following statement illegal, and how can it be fixed?

```
    middle = (low + high) / 2;  /* Aside: have you seen this before? */
```

Back to the Source

Modify the following program so that the max_min function uses a pointer instead of an integer to keep track of the current position in the array. Note that the function declaration **should not change** (although it could, in one small respect).

```
#include <stdio.h>

#define N 10


void max_min(int a[], int n, int *max, int *min);


int main(void)
{
  int b[N], i, big, small;

  printf("Enter %d numbers: ", N);
  for (i = 0; i < N; i++)
    scanf("%d", &b[i]);

  max_min(b, N, &big, &small);

  printf("Largest: %d¥n", big);
  printf("Smallest: %d¥n", small);

  return 0;
}

void max_min(int a[], int n, int *max, int *min)
{
  int i;

  *max = *min = a[0];
  for (i = 1; i < n; i++) {
    if (a[i] > *max)
      *max = a[i];
    else if (a[i] < *min)
      *min = a[i];
  }
}
```
