L10 - Pointers and Arrays

November 1, 2019

1 Pointers and Arrays

The declaration for an `int` array defines and array that can store 10 integers.

```
In [3]: #include <stdlib.h>
        #include <stdio.h>

        int main()
        {
            int a[10];
            int* pa; // pointer
            pa = &a[0]; // points to first value in a

            int x = 3;
            *pa = x; // makes a[0] into a 3

            printf("%d", a[0]); //see?
        }
```

```
3
```

1.0.1 Pointer Arithmetic

If `pa` points to any element in `a[]`: * `pa+1` is the address of the next element * `pa+i` is the address of the i'th element after the element `pa` points to

For example, after executing `pa = &a[0]`:

- `pa+1` is the address of `a[0]`
- `*pa+1` is the contents of `a[0]`
- `pa+i` is the address of `a[i]`
- `*pa+i` is the contents of `a[i]`

And the name of an array is a synonym for the address of its zeroeth element, so `pa = &a[0];` is the same as `pa = a;`. That means that `*(a+i)` is *also* the value of the i'th element of `a[]`.

```
In [ ]: double average(int data[], int h)
        {
            double sum = 0;
            int i;

            for (i = 0; i < n; i++)
            {
                sum = sum + data;
            }

            return sum / n;
        }

        int main()
        {
            int samples[100];

            average(samples, 100)
        }
```

In the code above, the paramater `data[]` is actually a pointer to the first element of an array of integers. The call `average(samples, 50)` is converted to the call `average(&samples[0], 50)`.

We can rewrite the function using pointer expressions:

```
In [ ]: double average( int *data, int n )
        {
            double sum = 0;
            int i;

            for(i = 0; i < n; i++)
            {
                sum = sum + *(data+i);
            }

            return sum / n;
        }
```

Note that the `data` parameter now has the type *pointer to* `int` and the *pointer-plus-offset* expression is used to access array elements.

```
In [ ]: double average( int *data, int n )
        {
            double sum = 0;
            int i;

            for(i = 0; i < n; i++)
            {
                sum = sum + *data;
                data += 1; // walking pointer
```

```
    }

    return sum / n;
}
```

As **parameters** in a function definition, `int data[]` and `int *data` are equivalent. The function can treat `data` as an array of integers, and access elements using subscripts; or it can treat `data` as a pointer to a block of integers and access integers using pointer notation: `*data` or `*(data + 1)`.