L10 - Pointers and Arrays

December 14, 2019

1 Pointers and Arrays

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

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

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

             for (i = 0; i < h; i++)
             {
                 sum = sum + data[i];
             }

             return sum / h;
         }

         int main()
         {
             int samples[10] = { 10, 11, 9, 12, 8, 9, 20, 23, 31, 9 };

             double m = average(samples, 10);
             printf("The average of the array is %3.2f.", m);
         }
```

```
The average of the array is 14.20.
```

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 [12]: #include <stdio.h>

         double average( int *data, int n )
         {
             double sum = 0;
             int i;

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

             return sum / n;
         }

         int main()
         {
             int samples[10] = { 10, 11, 9, 12, 8, 9, 20, 23, 31, 9 };
```

```
    double m = average(samples, 10);
    printf("The average of the array is %3.2f.", m);
}
```

```
The average of the array is 14.20.
```

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 [15]: #include <stdio.h>

         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;
         }

         int main()
         {
             int samples[10] = { 10, 11, 9, 12, 8, 9, 20, 23, 31, 9 };

             double m = average(samples, 10);
             printf("The average of the array is %3.2f.", m);
         }
```

```
The average of the array is 14.20.
```

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)`.