C

SWE110

Lesson 7

Prof. Zachi Baharav
zbaharav@cogswell.edu

Lesson 7

- In previous lesson:
  - Functions, program control
- **In this lesson:**
  - Arrays!!
- Next Lesson:
  - More on arrays
  - And then command line arguments, strings, etc..
- Lab

Array

- Collection of elements of the SAME type.

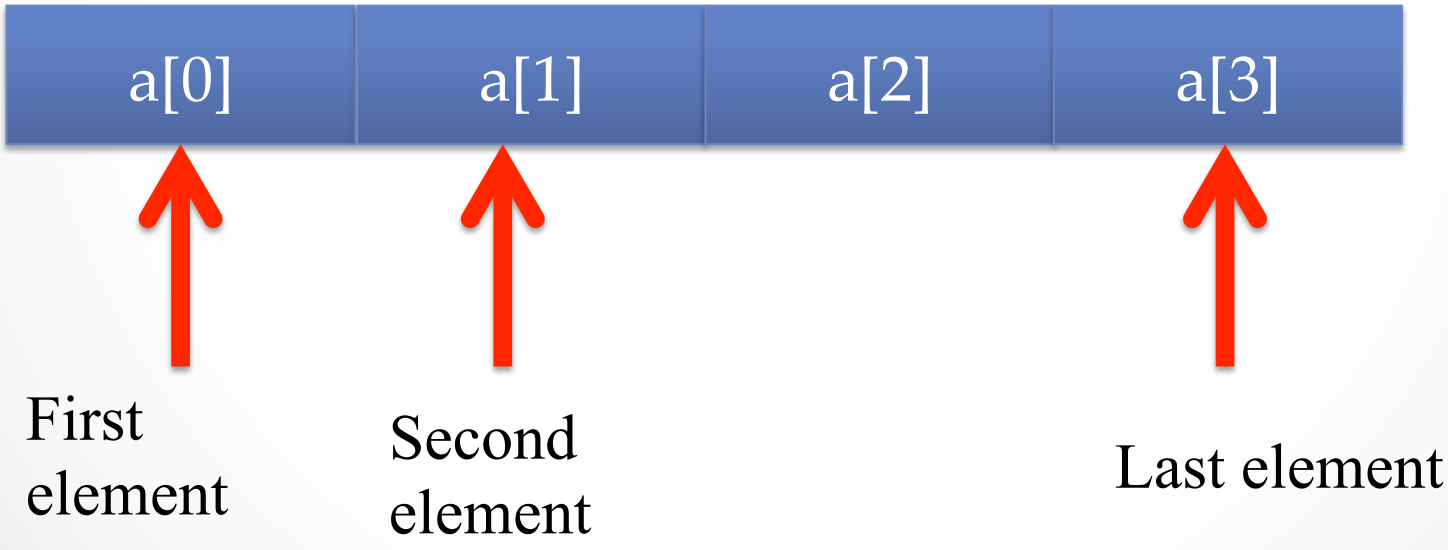
5 Elements.
num[0], num[1], … , num[4]

```
int     num[5] ;
char  a[80];
```

Memory

- Contiguous set of fixed-sized memory sections.
- Example:

int a[4] ;



Accessing and Assigning

- a[0] = 5;
- a[1] = 7;
- a[2] = a[1] – 1;

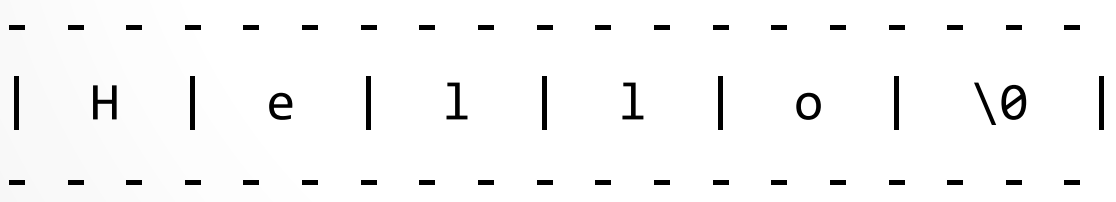
- int b[3] = {0, 1, 2}; // initialized
- int c[] = {0, 2, 4}; // implied 3 elements

- Things get tricky… :
  - You can also write and try to access **a[100]**, even though you asked for smaller space!! Run time error.
  - You can access **uninitialized** memory locations!

Character arrays (aka Strings)

- C does NOT have a string type.
- C saves strings as an array of characters
  - **Terminated by the null-character**

- "Hello" is saved in memory as:
  - An array of 6 elements



Declaring and Initializing

- `char str1[6] = “Hello”; // explicit`
- `char str2[] = “Hello”; // implicit`
- `Char str3[80];`

- Note (we’ll talk more in two slides):
  - `scanf(“%s”,str3);`

Passing arrays to functions

```
void    sample( char a[] , int num[6] ,   int*
num2 )
{
   … your code here …
}
```

- The length of the array is NOT passed by C.
  - You have to take care of this!

Memory and arrays

- The array-name evaluates to it's **memory location**!
- Thus, the computer actually does the following:

a [5] ==> '5' elements after memory location 'a'
or i.e, memory location **a+5**.

Since array-name is same as it's memory location

```
char   singleChar;
char   str[80];

scanf (" %c ",   &singleChar );

scanf (" %s ",   &str[0] );

scanf (" %s ",   str );
```

Time to program!!

END