CSCD240

C and Unix Programming

Today’s topic

- Introduction to Array;
- How Array is defined in C;
- Address of an Array;
- Determining the size of an array;
- Array types;
- Array Initialization;
- Some examples om Array

Array

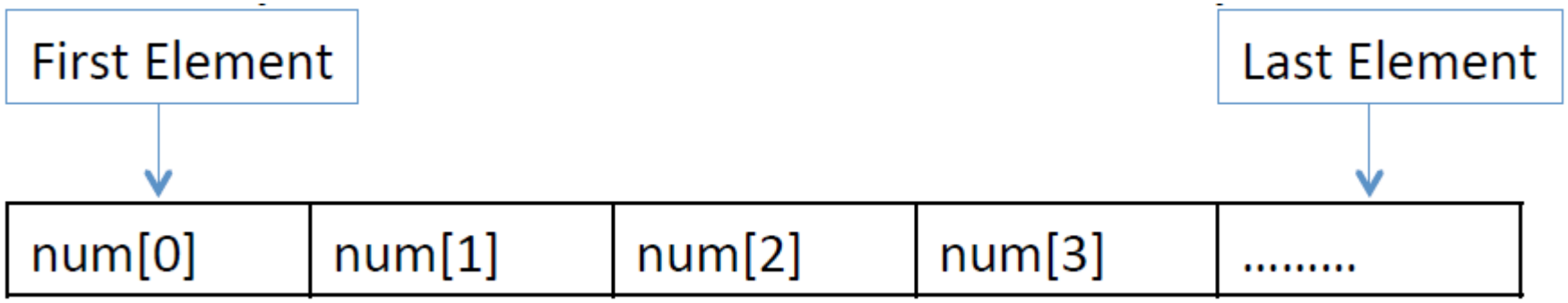
- Array in C is a data structure.
- It stores a **fixed-size** sequential collection of elements of the same type.
- We can assign/re-assign different values to each array element.
- The entire array consists of contiguous memory locations.

Array

- To define an array in C, we specify the type of the elements and the number of elements required by an array as follows:
  - **type arrayName** [**constantArraySize**];
  - Note here: the array size between bracket should be an integer constant greater than 0;
- unlike Java:
  - type [] arrayName; ← doesn’t work in C

Array

- Interestingly, in C, the **array name** is used as the address of the first element,
  - called array base address;
  - &(num[0]) and **num** represent the same value, the array base address. **num** is the array name here.



Array

- double balance[10];

- Now, ***balance*** is available to hold up to 10 double numbers.
- The maximum number of elements in balance is fixed,
  - We have to know the maximum capacity before **compile**.
  - If more than 10 values in a file, then balance array cannot hold them all.
  - If only 5 values in a file, then half of array memory is wasted.

Array

- Define means create the array entirely, including allocating memory space.
- If you do **int arr[10];** inside a function or in main(), the memory space for 10 integers is allocated **automatically** when the function is invoked.
    - You **need not** worry about the memory allocation in this case.

- This memory is deallocated **automatically** when the function (in which the array is defined) returns.

Array

- How to find the length (number of elements) of an array;
  - int a[MAXSIZE];
  - int numer_of_elements = sizeof(a)/sizeof(int)
- Different from java, isn’t it?

Arrays

- Can be global, automatic or static;

```
#include <stdio.h>
int id[50];
int main(void){

    double wage[50];
            :
            :

  return 0;
}
```

Arrays

- Globally defined arrays are implicitly initialized;

- Demo (GlobalvsLocal.c)

Array

- You can access array elements the same way as you do in java.

  - double salary = balance[9];
  - This accesses the tenth element in balance array.
  - Array index starts at 0 as we already learned in Java.

Array

- Can be 1D, 2D , 3D
- Can you give me examples of an 1D, 2D and 3D array?

- char x[i] ← 1D
- double y[i][j]← 2D
- char z[i][j][k]← 3D

Array Initialization

- float width[10] = {10.1, 12.6, 12.3, 10.0, 4.2, 2.3, 2.1, 3.9, 3.8, 5.4,};

- float width[10] = {10.1, 12.6, 12.3};← elements [0], [1] and [2] are explicitly initialized; all other elements are implicitly initialized to 0

- float width[10] = {0.0} ← first element explicitly initialized, rest of the elements are implicitly initialized;

Array Initialization

- int age[3] = { 40, 54, 63, 70} ← Initializing list contains too many items! Error!

- int age [] = { 40, 54, 63, 70}; ← sets the size of age array to 4 by default

Array: A few examples

- char color[] = “RED”; ←
- char color[4] = “RED” ←

- Both null terminated, i.e., ‘\0’

- What about the following one?
- char color[3] = “RED” ← not NULL terminated;