Faculty of Computers & Informatics

Introduction to Programming

Lecture 5: More about Declarations & Operators

Mahmoud El-Gayyar

elgayyar@ci.suez.edu.eg

Review of Chapter 4

- *Expression Statement*
- *Conditional*
  - if Statement
  - Nested if
  - switch Statement
- *Boolean Expressions*
- *Loops*
  - while Loop
  - for Loop
  - Continue & Break

Outline

- *Arrays*
  - Initialization
  - Multi-dimensional arrays
- *More Operators*
  - Assignment operators
  - Increment and decrement operators
  - Order of evaluation

Outline

- ***Arrays***
  - Initialization
  - Multi-dimensional arrays
- *More Operators*
  - Assignment operators
  - Increment and decrement operators
  - Order of evaluation

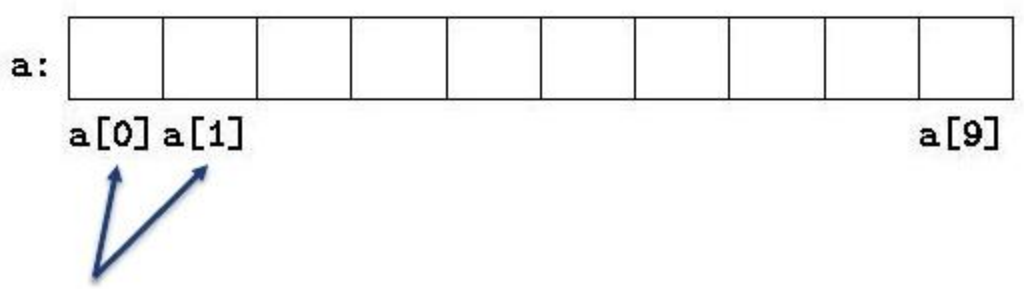
Arrays

- *Suppose you would like to store the salary of 1000 employees?!*
  - Trivial solution: define 1000 float variables, salaryEmployee1, salaryEmployee2, ...etc.
  - This will be a big miss!!
- *Arrays in programming languages allow you to solve this problem by storing multiple values with one variable name:*
  - But all values ***must be*** of the same type

Array Declaration

- *To declare an array of several elements, you need:*
  - Type of elements
  - Name of the array variable
  - Size (number of elements)

```
int a[10];
```



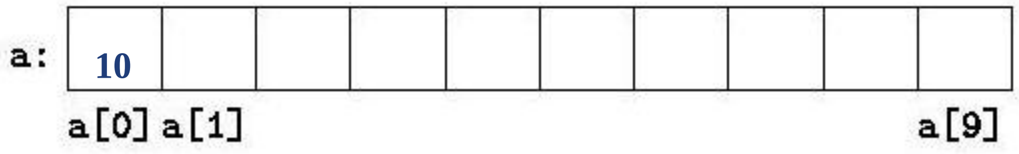
Index / subscript

- *Take care index is always start with **zero** and end with **length-1***

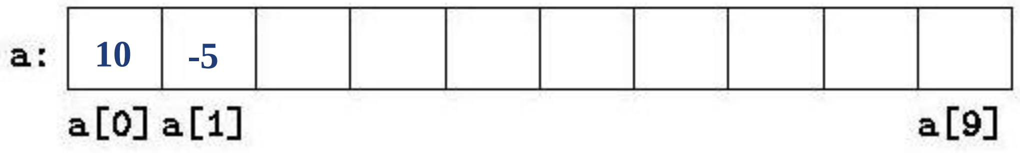
Arrays: Element Access

- *You can access array elements throgh the index / subscript*

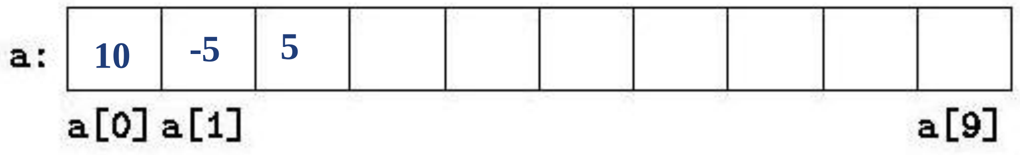
  - a[0]= 10;



  - a[1]=-5;



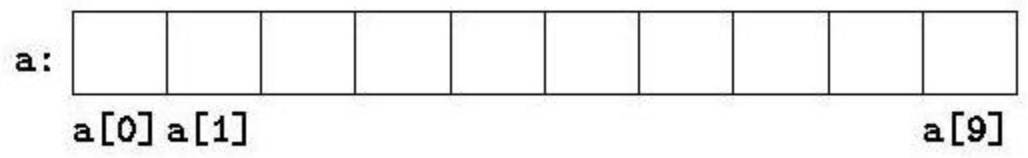
  - a[2]=a[0]+a[1];



  - a[10]=10;
    - *Out of index (memroy error)*

Arrays: Element Access

- *What if you would like to inialize all elements to 0?!!*



```
int i;
for (i=0 ; i < 10 ; i=i+1){
      a[i]=0;
}
```

- a=0;

- *What if you would like to copy array a into array b?*

- b=a;

- Again use loops?!

Arrays: Intialization

- *You can inialize arrays while declaration as in normal variables:*

```
int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

int a[] = {10, 11, 12, 13, 14};          //here array size is 5

int a[10] = {0, 1, 2, 3, 4, 5, 6};       //elements 7,8,9 are zeros

int a[100] = {0};

char s1[7] = "Hello,";                   // don't forget the null (0)

char s2[10] = "there,";                  //?

char s3[] = "world!";                    //array size?!
```

Arrays: Full Example

- *Suppose you would like to roll a pair of dice 100 times and see how often each roll (2 – 12) comes up.*

- *How to roll a dice?*
  - Use **rand( )** from **stdlib.h** that returns random integer (up to 32767)
  - You need to scale it to a value between 1 and 6: use %6 +1
    - *rand( ) % 6 + 1;*   *should be always between 1 and 6*
  - To get the outcome of 100 rolls, simply use a for loop

```
int i, outcome;
for (i=0 ; i < 100 ; i=i+1){
       d1=rand() % 6 + 1;
       d2=rand() % 6 + 1;
       outcome=d1+d2;
}
```

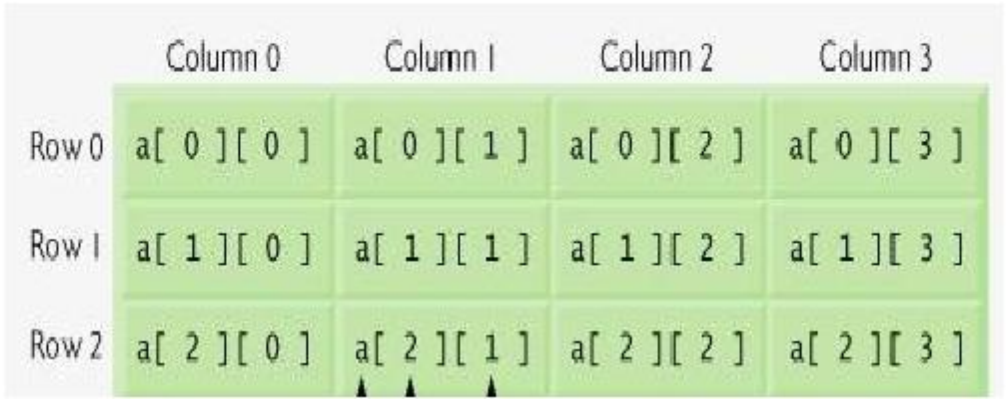
Arrays: Full Example

```
#include <stdio.h>
#include <stdlib.h>          /* for rand()*/

int main(){
       int i, d1, d2;
       int a[13]={0};        /* uses [2..12] */

       for(i = 0; i < 100; i = i + 1){
              d1 = rand() % 6 + 1;
              d2 = rand() % 6 + 1;
              a[d1 + d2] = a[d1 + d2] + 1;
       }

       for(i = 2; i <= 12; i = i + 1)
              printf("%d: %d\n", i, a[i]);

       return 0;
}
```

Multi-dimensional Arrays

- *The declaration of an array of arrays (matrix) looks like this:*

```
int a [3][4];
```



	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

- *a is a matrix with **3 rows** and **4 columns***
- *You will need **two loops** to handle all elements of a matrix*

Multi-dimensional Arrays

- *Example to set all cells in matrix a to 1:*

```
int i, j;
int a [3][4];
for(i=0 ; i < 3 ; i= i+1)
      for(j=0 ; j < 3 ; j= i+1)
            a[i][j]=1;
```

- *To print all elements on the screen in a matrix form:*

```
for(i = 0; i < 3; i = i + 1){

      for(j = 0; j < 4; j = j + 1)
            printf("%d\t", a2[i][j]);

      printf("\n");
}
```

Multi-dimensional Arrays: Intialization

- *Multidimensional arrays may be initialized by specifying bracketed values for each row:*

```
int a[3][4] = {
 {0, 1, 2, 3} ,    /*  initializers for row indexed by 0 */
 {4, 5, 6, 7} ,    /*  initializers for row indexed by 1 */
 {8, 9, 10, 11}    /*  initializers for row indexed by 2 */
};
```

- *The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to previous example:*

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

Outline

- *Arrays*
  - Initialization
  - Multi-dimensional arrays
- ***More Operators***
  - Assignment operators
  - Increment and decrement operators
  - Order of evaluation

Assignment Operators

```
i=i+1;
```

```
a[i+j+2*k] = a[i+j+2*k] + 1;
```

```
a[i+j+2*k] = a[i+j+2+k] + 1;
```

```
i+=1;
```

```
a[i+j+2*k] += 1;
```

k *= n + 1 → k=k*(n+1)

a[i] /= b → a[i] = a[i] / b

Increment & Decrement Operators

```
++i    → i=i+1;
--i    → i=i-1;
```

Prefix

```
i++    → i=i+1;
i--    → i=i-1;
```

Postfix

But take care, they are different!!!

```
i=1;
k = 2 * ++i;                //i=2, k=4
```

```
i=1;
k = 2 * i++;                //i=2, k=2
```

Increment & Decrement Operators

```
int i=1;
printf("i is %d\n", i++);
printf("i is %d\n", ++i);
```

```
1
3
```

Order of Evaluation

- *Expressions now are more complicated*

```
a[i++] = b[j++];                //?
```

```
a[i++] = b[i++];
```

- *We call this undefined expression, you have to avoid such type of expressions.*

Order of Evaluation

- *As another example, if you would like to set a[i]=i:*

```
int i, a[10];
      i = 0;
      while(i < 10)
            a[i] = i++;
```

WRONG!

- *We may end up with a[1]=0, a[2]=1,…*

- *A better form is to use a for loop in this case:*

```
for(i = 0; i < 10; i++)
        a[i] = i;
```

Problems with logical AND / OR

- *The main problem is that conditions accept arithmetic expressions*

```
if(x > 0 && x++ < 10){
}
if(x > 0 || x++ < 10){
}
```

- *The main problem here is that C first evaluates the first part of the compound logical expression.*
  - In case of AND: if the first part is false, it will not evaluate the second one
  - In case of OR: if the first part is true, it will not evaluate the second one

Summary

- *Arrays*
- *Matrices*
- *Other Operators*
  - Assignment
  - Increment / decrement
- *Don’t use ambiguous expressions*