Chapter 5: Static Arrays

Lesson 1: One-dimensional array

Learning Outcomes:

- Declare a one-dimensional array and initialize, modify and manipulate its content.
- Solve problems that would make use of one-dimensional arrays.

So far the variables that are used would only store a single data item for a single variable. An array in C would act as a storage of multiple data items under as single name, referenced by the index (or subscript). Think of it as a simple list or grouping for variables of the same type, thus making the programmer organize related data efficiently. For this particular chapter only fixed sized array during runtime would be covered.

Declaring and Initializing

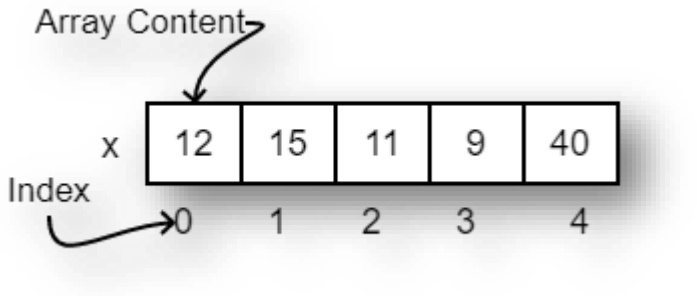
In C the arrays are declared and initialized with the syntax below.

```
datatype arrayName [size] = {var0, var1, …var_size-1};
```

For example if you would declare and initialize an array with 5 elements.

```
int x[5] = {12,15,11,9,40};
```

This example can be illustrated by a simple list shown below.



When the size of the array would be omitted, it would automatically allocate the size for the initialized data.

```
int y[] = {0,1,2,3,4,5}; //the array size would now be 6
```

If the dimension is specified, but not all elements are initialized, the remaining uninitialized elements will be allocated with 0.

```
int z[5] = {150}; // the array content would be {150,0,0,0,0}
```

Accessing and modifying array content

To access and modify the content of the array, you need to specify the array name and its index. The syntax format for accessing array content is:

```
variable = arrayName[index]
```

While the syntax for modifying the array content is:

```
arrayName[index] = variable/constant
```

For example by the 5 element array below is accessed and modified.

```
int w [5] = {3,5,1,7,9};
int var;
var = w[0];//var would now contain 3
w[1] = var;//index 1 of array w will now contain 3
```

Take note that C has no out of bound checking, so by accessing an index beyond the size of the array would cause runtime error. For example

```
var = w[6]// this is out of bound, so runtime error would ensue
```

It is also convenient to use a for loop to access and modify the content of the array. For example, by using a loop to display the content of the array, you would stray away from the tedious task of accessing and displaying the content one by one.

```
int i=0;
for (i=0;i<5;i++)
{
    printf("%d,",w[i]);
}
//output: 3,3,1,7,9
```

Another example is you can modify the content of the array by using a loop, for instance if you would change the content of the array all to zero during runtime.

```
for (i=0;i<5;i++)
{
    w[i] = 0;
}
```

Lesson 2: Two-dimensional array

Learning Outcomes:

- To declare a two-dimensional array and initialize, modify and manipulate its content.
- To solve problems that would make use of two-dimensional arrays.

A 2 dimensional array can be created by stacking up 1 dimensional arrays. Think of it as stacking rows of items to create a table. Although by applying the same idea, you could create a 3 dimensional array by stacking 2 dimensional array and a 4 dimensional array by stacking 3 dimensional array and so on. For practical purposes only 2 dimensional array would be covered.

Discussion

Declaring and Initializing

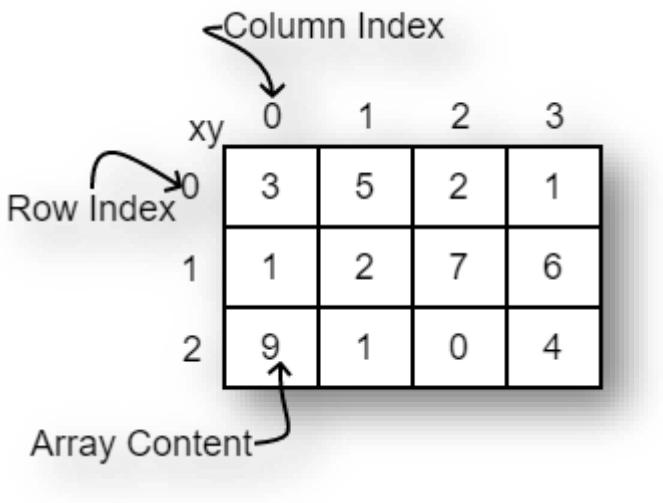
A two-dimensional array is declared and initialized using the syntax format:

```
datatype arrayName [rowSize][colSize] = {{row1Init},{row1Init}…};
```

For example a 3 row, 4 column integer two-dimensional array with initialized with content:

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

This two-dimensional array can be illustrated in table for as shown below.



Accessing and modifying two-dimensional array content

When accessing and modifying an array content, the row location and column location must be indicated. The syntax format is as follows:

```
variable = arrayName[row][col]; //accessing
arrayName[row][col] = variable/constant; //modifying
```

By using the same declaration from the previous example, accessing and modifying array content.

```
int vars;
vars = xy [0][0];//assign content of 0,0 (row, col)to vars
xy [0][1] = 10;//modifying 0,1 (row, col) content with 10
```

A convenient way of accessing and modifying the content of a two-dimensional array is to use nested loops (i.e. typically for loops). For example, when displaying the content of two-dimensional array xy can be done as follows:

```
int i=0,j=0; // i is for row index, j is for column index
for (i=0;i<3;i++)
{
    for (j=0;j<4;j++)
    {
        printf("%d,",xy[i][j]);
    }
    printf("\n");
}
```

Moreover, if you want to modify all the content of two-dimensional array xy with all 1, it could be done as shown on the code below

```
for (i=0;i<3;i++)
{
    for (j=0;j<4;j++)
    {
       xy[i][j]=1;//modify content @ (row,col)
    }
}
```