Fundamentals of Programming

Lecture 9

Chamila Karunatilake

Department of Information and Communication Technology

Faculty of Technology

University of Sri Jayewardenepura

chamilakarunatilake@sjp.ac.lk

Arrays

Arrays in C

- An array is a data structure that can store a **sequential collection of elements** of the **same type**.
- All arrays consist of **consecutive memory locations**.
- Array can be thought of as a **collection of variables** of the **same type**.

10	81	34	8	95	13	19	56	75

H	E	L	L	O	!

Array Declaration

- In normal variable declaration, only the data type and the variable name is sufficient.
- When declaring an array, there is one addition to that. The **size of the array** has to be mentioned with in brackets.

  Syntax:

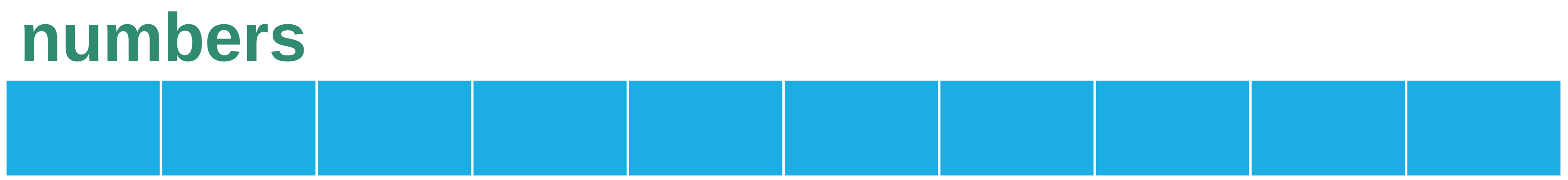
```
<type> <arrayName>[<array_size>];
 int numbers[10];
```

- The **array size** indicates the **number of elements in the array**.
- Once the array is declared, the size cannot be changed.

Array Declaration

- When an array is declared, a collection of memory locations are reserved with the name of the array.

```
int numbers[10];
```



- At this point, **no values are assigned** to the elements.
- However, the array elements **may contain garbage values**.

Array Element Identification

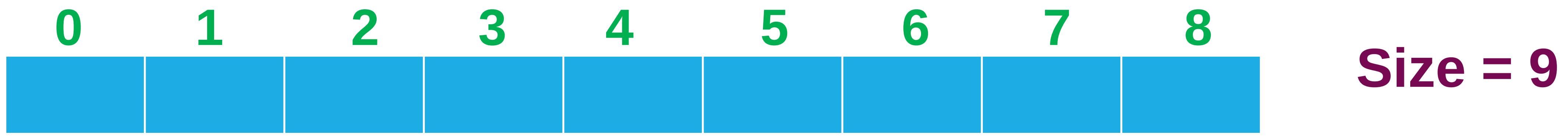
- Normal variables can be identified using their names.
- Read and write operations can be done via name of the variable.

```
int number;
number = 10;
printf(“%d”, number);
int i = number;
number++;
```

How this can be achieved with the array elements?

Array Element Identification

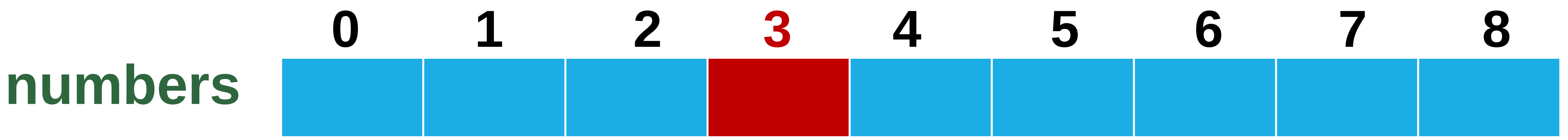
- In an array, **index** is used when working with **individual elements**.
- **Array Index** is a **sequential number** which indicates the **position** of a particular element.
- Index **starts with 0**. First element of the array is indexed with 0.
- The **last element's index** become the **size of the array - 1**.



Array Element Identification

- Element identification can be achieved using **name and index** together(indexing the name).

<arrayName>[<index>]



If we want to deal with the fourth element(index 3) of the above array, following syntax can be used.

numbers[3]

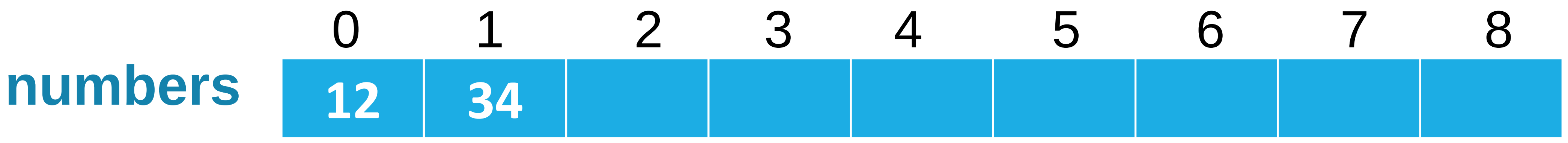
Array Initialization

- Array initialization is the process of **assigning values** to the array elements.
- There are two initialization methods:
  - Initialize one element at a time
  - Initialize whole array at once

Array Initialization

- Elements can be initialize **one at a time** using **name** of the array and the **index** of the element.

```
e.g.    numbers[0] = 12;
        numbers[1] = 34;
        ……………………..
```



Array Initialization

- Whole array can be **initialized** at once **when array is declared**.

e.g. int numbers[10] = {12,34,13,56,78,72,90,11,80,67} ;

	0	1	2	3	4	5	6	7	8	9
numbers	12	34	13	56	78	72	90	11	80	67

- When the array is initialized in this way, the **size declaration is optional**. You **can declare** the array **with empty brackets** without size inside(un-sized).

int numbers[] = {12,34,13,56,78,72,90,11,80,67} ;

Array Initialization

- There are several variations and special cases of initializing whole array at once.

```
int numbers[10] = {0} ;
```

	0	1	2	3	4	5	6	7	8	9
numbers	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

```
int numbers[10] = {} ;
```

	0	1	2	3	4	5	6	7	8	9
numbers	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Array Initialization

```
int numbers[10] = {1} ;
```

	0	1	2	3	4	5	6	7	8	9
numbers	1	0	0	0	0	0	0	0	0	0

```
int numbers[10] = {12,34,5,26} ;
```

	0	1	2	3	4	5	6	7	8	9
numbers	12	34	5	26	0	0	0	0	0	0

Accessing Array Elements

- An element of an array **can be accessed** by **indexing the array name**.

```
int myNumber = numbers[3];

int sum = numbers[3] + numbers[4];

printf(“%d”, numbers[3] );
```

```
#include <stdio.h>

int main()
{
      int numbers[4] = {12,34,13,56} ;
      printf("%d\n",numbers[0]);
      printf("%d\n",numbers[1]);
      printf("%d\n",numbers[2]);
      printf("%d\n",numbers[3]);
      float average = (numbers[0] + numbers[1] +
numbers[2] + numbers[4])/4.0 ;
      printf("%.2f\n",average);
      return 0;
}
```

Accessing Array Elements using loops

- Loops are used frequently when working with arrays.
- When printing the elements of the whole array, it could be done using for loop.

```
for(int i=0;i<10;i++){
   printf("%d\n",numbers[i]);
}
```

- When initializing an array with the same value or values with a sequential patterns, for loop can be used.

```
for(int i=0;i<10;i++){
   numbers[i] = 100 + i;
}
```

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

```c
#include <stdio.h>

int main()
{
    int numbers[10];
    for(int i=0;i<10;i++)
    {
        numbers[i] = 12 * (i+1);
    }

    for(int i=0;i<10;i++)
    {
        printf("%d\n",numbers[i]);
    }
    return 0;
}
```

Passing Arrays as Function Arguments

- As any other variables, an array can be passed **as an argument** in a function.
- Formal **parameter** can be **declared** as **a sized** or **an un-sized array**.

```
void myFunction(int param[10]) {
    .....
}
```

```
void myFunction(int param[]) {
    .....
}
```

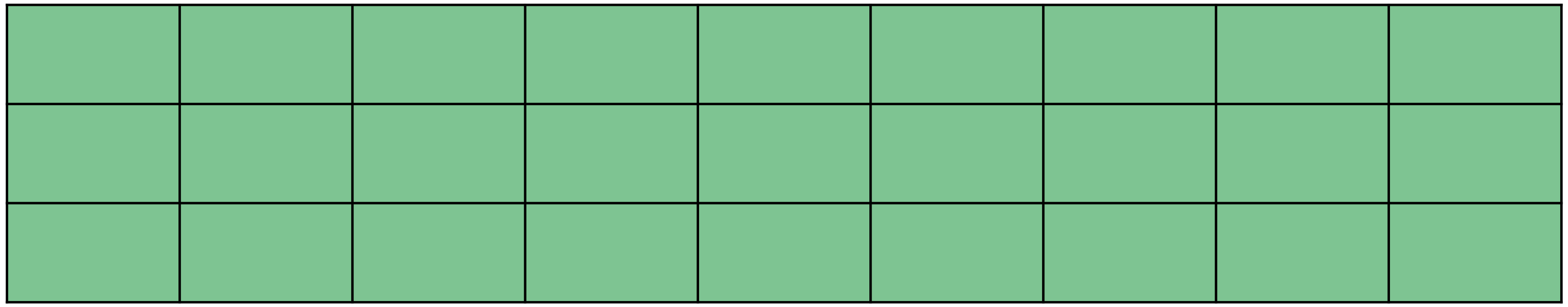
```
#include <stdio.h>

double getSum(int arr[], int size);

int main ()
{
   int numbers[5] = {1000, 2, 3, 17, 50};
   double result;

   result = getSum(numbers, 5 ) ;

   printf( “Total Sum is: %.2f ", result);
   return 0;
}
```

```
double getSum(int arr[], int size)
{
    double sum = 0;

    for (int i = 0; i < size; ++i)
    {
        sum += arr[i];
    }

    return sum;
}
```

C Multidimensional Arrays

- In C programming, you can create **an array of arrays** known as multidimensional array.
- The following example creates two-dimensional array.

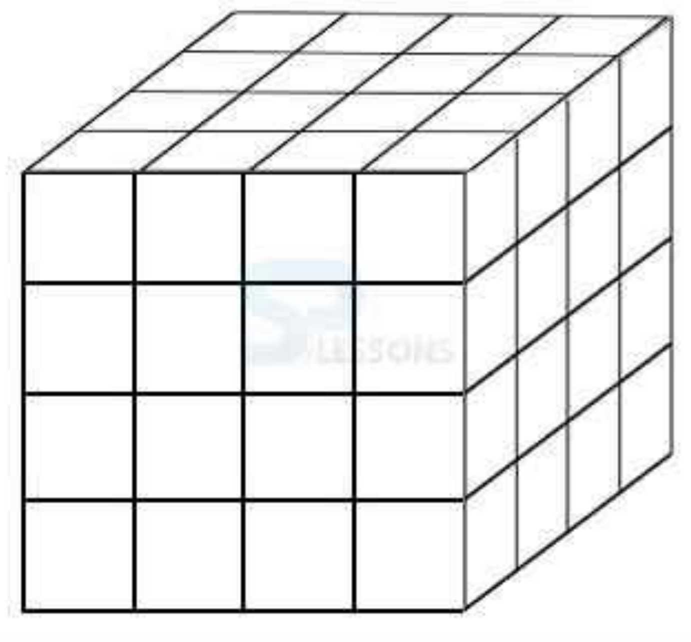
```
int a[3][9];                    // [rows][Columns]
```



C Multidimensional Arrays

- It’s possible to create higher dimensional arrays as well.

```
int ma3d[4][4][4];
```



Initializing Multi-Dimensional Arrays

- Multidimensional arrays may be initialized by specifying bracketed values for each row.
- Following is an array with 3 rows and each row has 4 columns.

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

- The nested braces, which indicate the intended row, are optional.

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

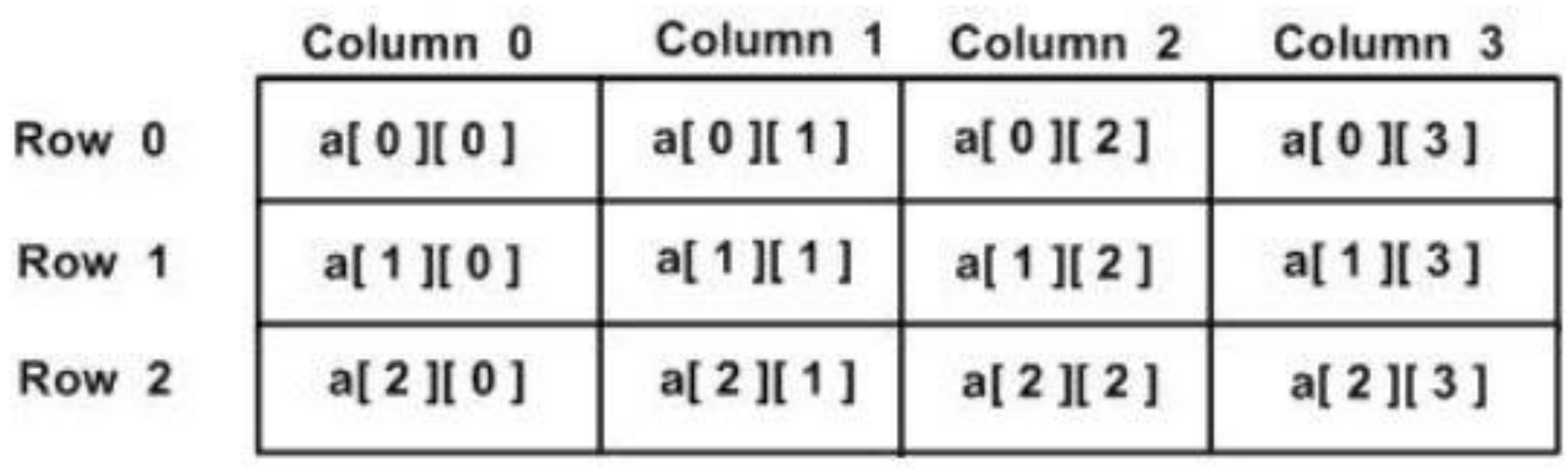
✓

Access Multidimensional Array Elements

- An element in a multi-dimensional array can be accessed by using the indices of different levels(rows and columns).

```
int val = a[2][3];
```

// [row][Column]



	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]

```c
#include <stdio.h>

int main () {
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;
   for ( i = 0; i < 5; i++ ) {
      for ( j = 0; j < 2; j++ ) {
         printf("a[%d][%d] = %d\n", i,j, a[i][j] );
      }
   }
   return 0;
}
```

Questions?