CADT
បណ្ឌិត្យសភាបច្ចេកវិទ្យាឌីជីថលកម្ពុជា
Cambodia Academy of Digital Technology

IDT
វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល
Institute of Digital Technology

Week 09

Array and String

Prepared by: **Leangsiv HAN**

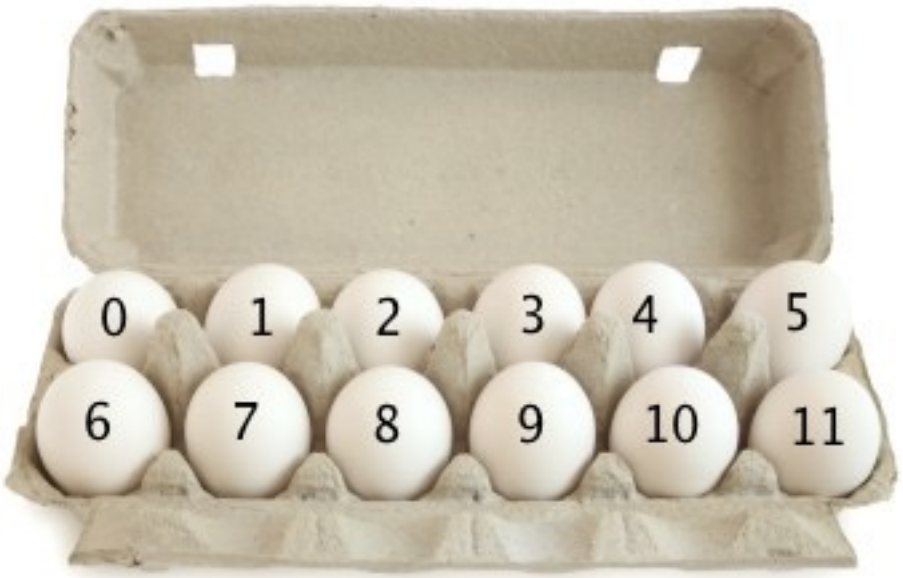
26th April 2024

Before Khmer New Year

You have learned about:

- ❑ Introduction to CPF
- ❑ Number Systems
- ❑ Variable, Data types, and Operators
- ❑ Input and Output
- ❑ Flowchart
- ❑ Control Statements – Selection and Iteration

Let’s get started with this image!



How about this image?

Learning Objectives

By the end of this lesson, you will be to:

- ❑ Define an array and identify its types.
- ❑ Differentiate between one-dimensional and multi-dimensional arrays.
- ❑ Define a string and demonstrate its usage.

Array

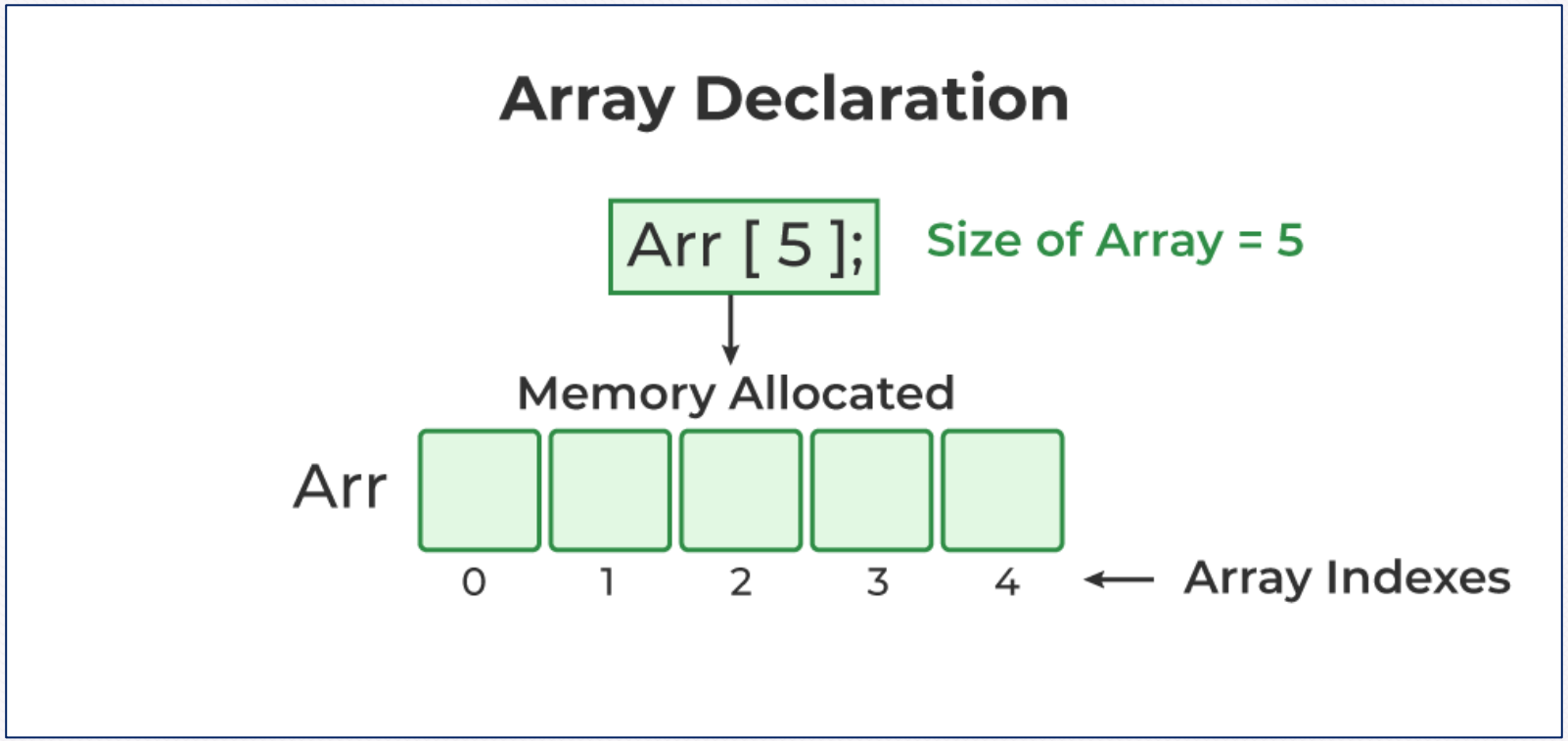
- **An array** is a *collection of individual data elements* that are ordered, fixed in size, and homogeneous.
- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
- There are **two types of arrays**, including:
  - One-dimensional or Single-dimensional array (1D Array)
  - Multi-dimensional array

One-Dimensional Array (1/6)

- The array is **one-dimensional or single-dimensional**, there will be a single subscript or index whose value refers to the individual array element which ranges from 0 to (n–1), where n is the total number of elements in the array.
- ***Syntax of Array Declaration***:

```
data_type array_name [size];
```

One-Dimensional Array (2/6)



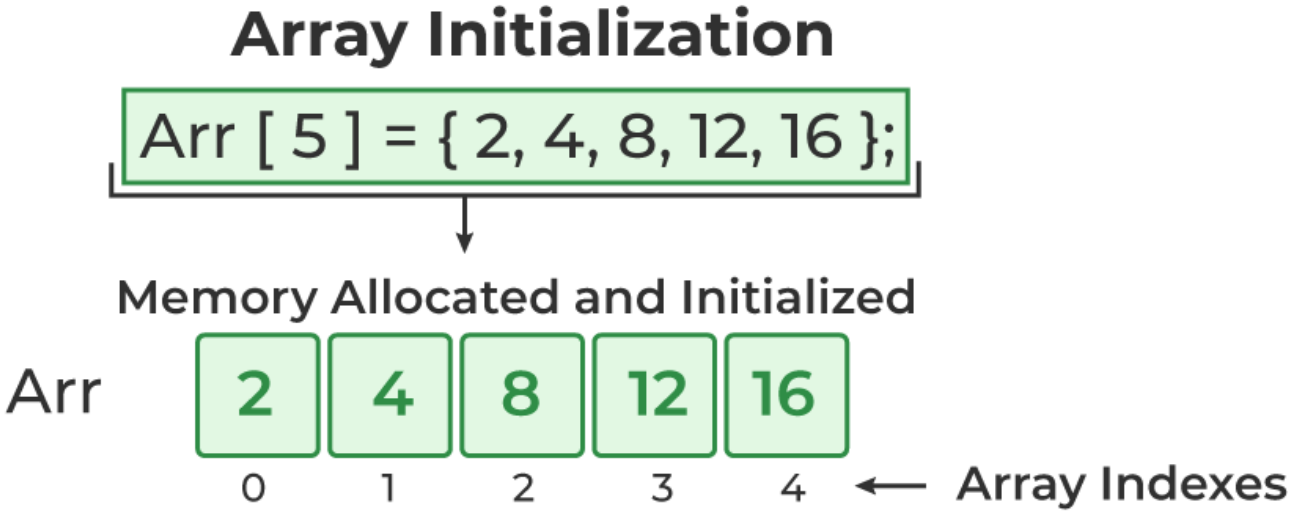
One-Dimensional Array (3/6)

- ***Syntax of Array Initialization with Declaration***:

```
data_type array_name [size] = {value1, value2, ... valueN};
```

In this method, we initialize the array along with its declaration. We use an initializer list to initialize multiple elements of the array. An initializer list is the list of values enclosed within **braces { }** separated with a comma.

One-Dimensional Array (4/6)



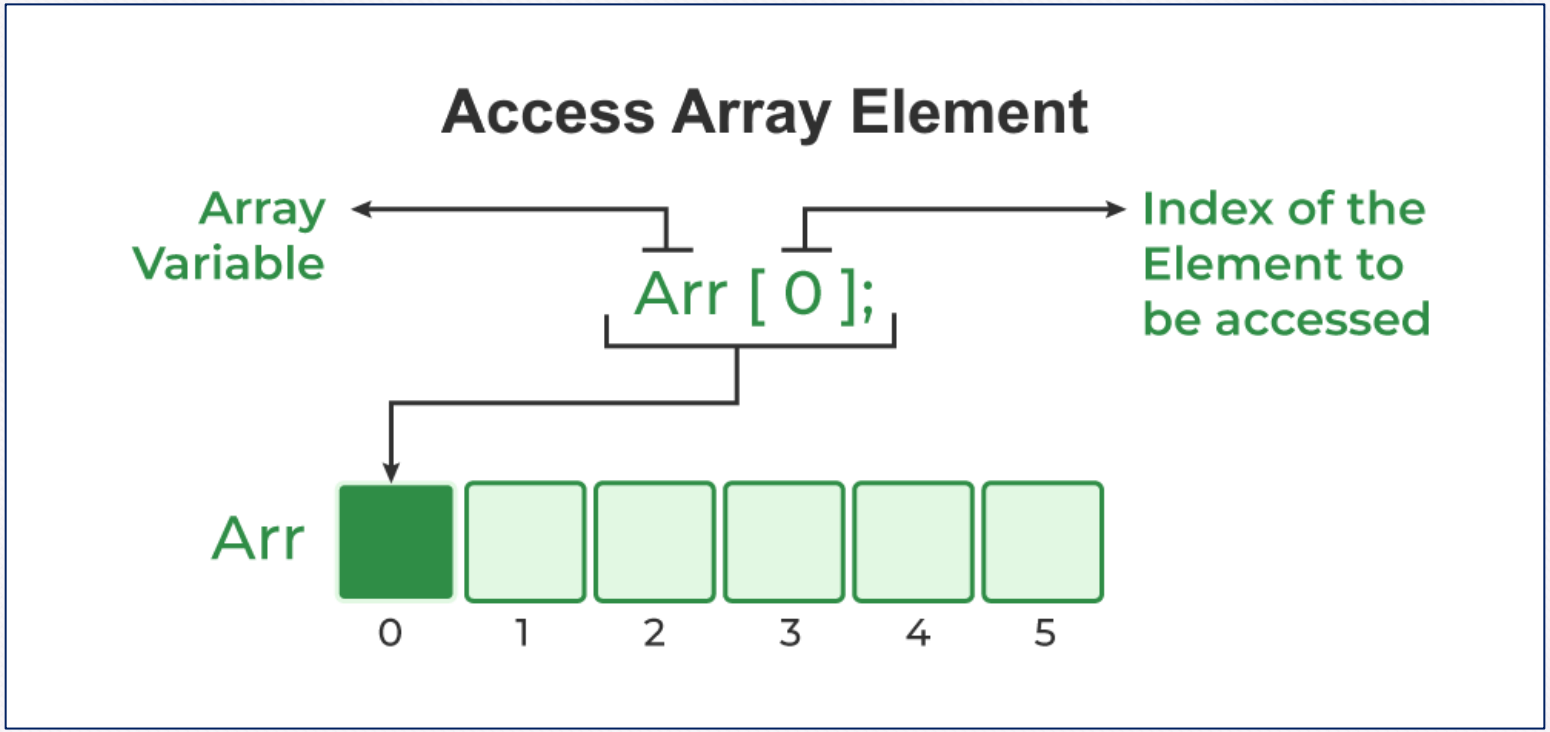
One-Dimensional Array (5/6)

- ***Syntax of Access Array Elements***:

```
array_name [index];
```

- We can access any element of an array in C using the array subscript operator [ ] and the index value i of the element.
- One thing to note is that the indexing in the array always starts with 0, i.e., the first element is at index 0 and the last element is at N – 1 where N is the number of elements in the array.

One-Dimensional Array (6/6)



Self-study

One-dimensional Array

- Array Initialization with Declaration without Size.
- Array Initialization after Declaration (Using Loops).
- C Array Traversal.

Multi-Dimensional Array (1/3)

- Arrays with *more than one dimension* are called **multidimensional arrays**. Although humans cannot easily visualize objects with more than three dimensions, representing multidimensional arrays presents no problem to computers.
- ***Syntax of Array Declaration***:

```
data_type array_name [size1] [size2]...[sizeN];
```

Where:

- data_type: Type of data to be stored in the array.
- array_name: Name of the array.
- size1, size2,…, sizeN: Size of each dimension.

Multi-Dimensional Array (2/3)

Example:

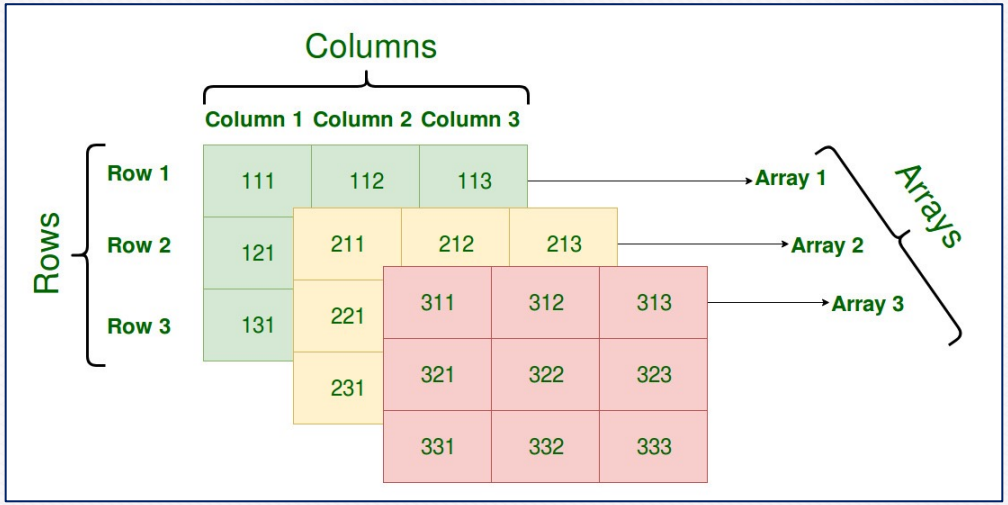
- **Two-dimensional array (2D Array):** int two_d[3][3];
- *Graphical Representation of Two-Dimensional Array of Size 3 x 3*

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

Multi-Dimensional Array (3/3)

Example:

- **Three-dimensional array (3D Array):** int two_d[3][3][3];
- *Graphical Representation of Three-Dimensional Array of Size 3 x 3 x 3*



Self-study

Multi-dimensional Array

- Array Initialization with Declaration without Size.
- Array Initialization after Declaration (Using Loops).
- C Array Traversal.

String

- **Strings** (one-dimensional character arrays) in C are represented by *arrays of characters*.
- The end of the string is marked with a special character, the null character, which is a character whose bits are all zero, i.e., a NUL (not a NULL).
- ***Syntax:***

```
char string_name[size];
```

String Initialization

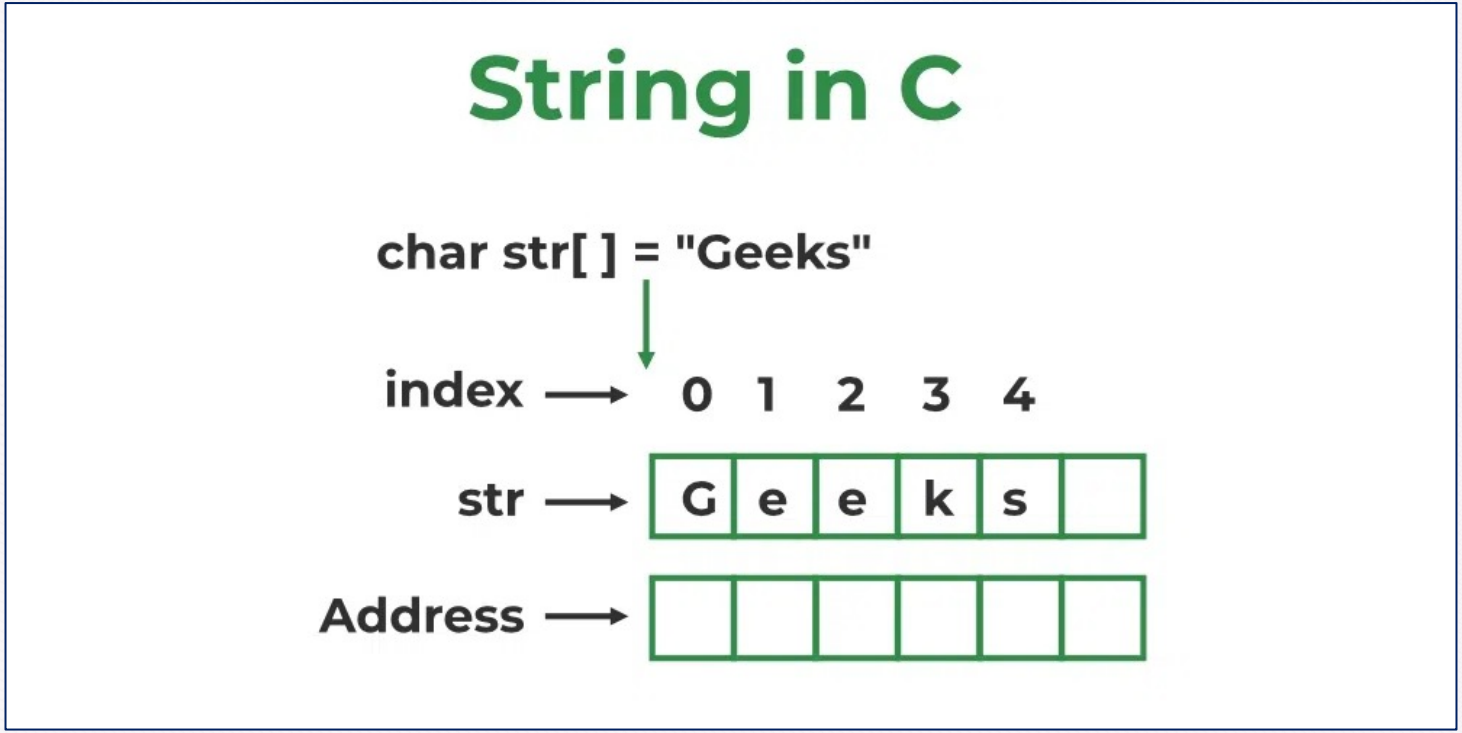
Character arrays or strings allow a shorthand initialization, for example,

```
char str[9] = “I like C”;
```

which is the same as

```
char str[9] = {‘I’,‘ ’,‘l’,‘i’,‘k’,‘e’,‘ ’,‘C’,‘\0’};
```

An example of String in C



Self-study

Function	Description
`ialnum(c)`	Returns a non-zero if c is alphabetic or numeric
`isalpha(c)`	Returns a non-zero if c is alphabetic
`scntrl(c)`	Returns a non-zero if c is a control character
`isdigit(c)`	Returns a non-zero if c is a digit, 0 – 9
`isgraph(c)`	Returns a non-zero if c is a non-blank but printing character
`islower(c)`	Returns a non-zero if c is a lowercase alphabetic character, i.e., a – z
`isprint(c)`	Returns a non-zero if c is printable, non-blanks and white space included
`ispunct(c)`	Returns a non-zero if c is a printable character, but not alpha, numeric, or blank
`isspace(c)`	Returns a non-zero for blanks and these escape sequences: '\f', '\n', '\r', '\t', and '\v'
`isupper(c)`	Returns a non-zero if c is a capital letter, i.e., A – Z
`isxdigit(c)`	Returns a non-zero if c is a hexadecimal character: 0 – 9, a – f, or A – F
`tolower(c)`	Returns the lowercase version if c is a capital letter; otherwise returns c
`toupper(c)`	Returns the capital letter version if c is a lowercase character; otherwise returns c

Character functions in <ctype.h> where c is the character argumen

Function	Description
`strcpy(s1,s2)`	Copies `s2` into `s1`
`strcat(s1,s2)`	Concatenates `s2` to `s1`. That is, it appends the string contained by `s2` to the end of the string pointed to by `s1`. The terminating null character of `s1` is overwritten. Copying stops once the terminating null character of `s2` is copied.
`strncat(s1,s2,n)`	Appends the string pointed to by `s2` to the end of the string pointed to by `s1` up to *n* characters long. The terminating null character of `s1` is overwritten. Copying stops once *n* characters are copied or the terminating null character of `s2` is copied. A terminating null character is always appended to `s1`.
`strlen(s1)`	Returns the length of `s1`. That is, it returns the number of characters in the string without the terminating null character.
`strcmp(s1,s2)`	Returns 0 if `s1` and `s2` are the same. Returns less than 0 if `s1<s2`. Returns greater than 0 if `s1>s2`.
`strchr(s1,ch)`	Returns pointer to first occurrence ch in `s1`.
`strstr(s1,s2)`	Returns pointer to first occurrence `s2` in `s1`.

String manipulation functions available in string.h

Key Takeaways

You are now able to:

- ✓ Define an array and identify its types.
- ✓ Differentiate between one-dimensional and multi-dimensional arrays.
- ✓ Define a string and demonstrate its usage.

References

- Dey, P., & Ghosh, M. (2013). *Computer fundamentals and programming in C*.
- *C arrays*. (2023, August 23). GeeksforGeeks. https://www.geeksforgeeks.org/c-arrays/
- *C arrays*. (n.d.). W3Schools Online Web Tutorials. https://www.w3schools.com/c/c_arrays.php
- *Strings in C*. (2024, March 7). GeeksforGeeks. https://www.geeksforgeeks.org/strings-in-c

Thank you !

Questions or Feedbacks?

Contact Me via:

leangsiv.han@cadt.edu.kh

@leangsiv

leangsivhan