11th Chapters

Data Structure Basics (Array)

In the general context, the term “Array” mean an ordered, structured collection of items, commonly used to organize data or objects in a specific arrangement. For example, during a buffet where food were arranged side by side in orderly manner. We could respond to it with: There was a splendid array of food on the table. The array were used to describe any object or item or even person that were arranged. We can even see how word “arranged” were influenced by “array” at some way but we are not going to talk about the word array in general context any longer.

In the context of programming, array were one of the essential data structure that could be utilized for better code management. That is correect, unlike the previous chapters where we talked about programming paradigm like branching and looping, array were mainly a data structure. How does array work? Now that is what we going to learn on this chapter.

Pre-Test : Understanding Array Concept

Evaluation Score

Instruction:

- Answer all questions.
- For multiple-choice questions, choose the best answer.
- For problem-solving questions, write your logic clearly.

Multiple-Choice

1. What is an array in programming?

A. A variable that stores only one value

C. A collection of elements of the same data type stored in contiguous memory

B. A collection of variables of different types

D. A loop structure

2. What is the index of the first element in an array in C?

A. 0

C. -1

B. 1

D. Depends on user input

3. Which of the following is the correct declaration of an array in C?

A. `int arr;`

C. `array int arr[5];`

B. `int arr[5];`

D. `int arr = [5];`

4. How many elements are there in the following array?

int arr[5];

A. 4

C. 6

B. 5

D. Undefined

5. What will be the output of the following code?

```
int arr[3] = {10, 20, 30};
printf("%d", arr[1]);
```

A. 10

C. 30

B. 20

D. Eror

Short Answer Questions

6. Explain in your own words what an array is.

7. Why do programmers use arrays instead of multiple variables?

8. What is the meaning of “index” in an array?

Basic Logic Recognition

09. If you want to store the scores of 30 students, which is better:

- 30 separate variables
- One array?

Answer:

10. Write a simple declaration of an integer array that can store 10 elements.

Answer:

Introduction to Array in Programming

An array is a fundamental data structure in programming used to store multiple values of the same data type in a single variable. Instead of declaring many variables individually, an array allows programmers to manage a collection of data efficiently.

In programming, arrays are stored in contiguous memory locations, meaning each element is placed next to each other in memory. This allows fast access using an index.

To make a proper analogy in real life examplee, array in programming is something like a locker. One locker could store multiple things at once. All of thesee lockers are of the same type, each of the locker have identifier, for example a number or perhaps owner's name. To access one of the locker, first we need to know the identifier for which locker we want to access.

Key Characteristic of Array

- Same Data Type - All elements must be of the same type (e.g., all integers).
- Indexed Structure - Each element is accessed using an index (starting from 0).
- Fixed Size - The size of an array is defined when it is created.
- Efficient Access - Elements can be accessed directly using their index.

Array Declaration in C

The format to declare an *array* in C will be

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

If we break down the code, *data_type* mean the type of data for the array and *array_name* will be the name of the *array variable*. For example, when we would like to create an integer type of array with the name "numbers" then the code would be:

```
int numbers[5];
```

Then we have create an array of "numbers" with total row of 5. Remember, while in other programming array could have dynamic number of rows, but due to technical limit in C, the number of row cannot be modified once it has been declared. However, we could declare an array along with it's values, for example we could write:

```
int numbers[5] = {10, 20};
```

The code will declare an array and two first rows, while the other remain 0 value. Or perhaps, if we don't know how many rows we need to initialize but we know the value required, we could declare it with:

```
int numbers[ ] = {10, 20};
```

But by doing so, will automatically initialize the row for numbers into only two.

Accessing Array Element

To acces an array, we write:

```
printf("%d", numbers[0]); // Output: 10

printf("%d", numbers[2]); // Output: 0
```

Remember that array index always start with 0 instead of 1. Now, for example, we would like to change the value of 2nd row of array. Then we write:

```
numbers[1] = 100;
```

The previous value for 2nd row (20) now changed into 100. For better understanding on how array work, try to run this program:

```
#include <stdio.h>

int main() {

    int numbers[3] = {5, 10, 15};


    printf("%d\n", numbers[0]);

    printf("%d\n", numbers[1]);

    printf("%d\n", numbers[2]);

    return 0;

}
```

What is the output of this program? Has we finally understand how array work?

Common Mistakes in Array

1. Forgot the starting point – As mentioned before, array always start with 0. Mean if we declare a 4 row of array, the index will consists of: “0 , 1 , 2 and 3” instead of 4.
2. Wrong Initialization Size – In C, if we declare an array that consist of 4 row, then we should declare it only with 4 value.
3. Mistaking Array and Variable – There are difference in how we initialize Array and Variable. Trying to access a row from a common variable would lead to an error.Mini-Project – Error Handling Menu-Based Program Using Looping

One Dimensional Array

A one-dimensional array is the simplest form of an array. It stores elements in a single line (or list), where each element can be accessed using a single index. This type of array is widely used for storing lists of data such as student scores, numbers, or simple datasets.

Our previous example with *numbers* array are one of the proper example on how to utilize an One Dimensional Array. But the utilization of array didn't end just in there. For example, we could combine array with a loop to create a program to input array value:

```
#include <stdio.h>

int main() {

    int arr[5];

    // Input

    for(int i = 0; i < 5; i++) {

        printf("Enter value %d: ", i);

        scanf("%d", &arr[i]);

    }

    // Output

    printf("Array elements:\n");

    for(int i = 0; i < 5; i++) {

        printf("%d ", arr[i]);

    }

    return 0;

}
```

Common usages of One Dimension Array

- Traversing (printing elements)
- Searching (finding a value)
- Summation
- Finding maximum/minimum

Two Dimensional Array

A two-dimensional array is an array of arrays. It is used to store data in the form of a table consisting of rows and columns, similar to a matrix.This structure is useful when dealing with data that naturally fits into a grid format, such as:

- Student scores (students × subjects)
- Tables
- Game boards
- Matrices in mathematics

How does Two Dimensional Array Work?

If we visualize 2D array, it will be like:

```
         Column
         0   1   2
Row 0   10  20  30
Row 1   40  50  60
Row 2   70  80  90
```

And if we want to access one of the row, we use:

```
arr[row][column]
```

For example, we fill the row with 1 and column with 2 and we will got 60.

Now, to have better understanding on how 2D Array work, try to run this program of inputing array that has been combine with loop like the previous one:

```
#include <stdio.h>
int main() {
    int arr[2][2];
    // Input
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++) {
            printf("Enter value [%d][%d]: ", i, j);
            scanf("%d", &arr[i][j]);
        }
    }
    // Output
    printf("Matrix:\n");
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
```

```
    return 0;
}
```

Multi Dimensional Array

A multidimensional array is an extension of arrays that allows storing data in more than two dimensions. While a one-dimensional array represents a list and a two-dimensional array represents a table, a multidimensional array can represent more complex data structures such as cubes or higher-dimensional data. While in some common programing activity this array is rarely used. The multi dimensional array is very useful in processing a lot of data such as in Game Development or Data Science.

Concept of Multidimentional Array

A multidimensional array can be seen as an array of arrays of arrays. If we visualize it, it will be like:

```
Layer 0:
[1 2]
[3 4]

Layer 1:
[5 6]
[7 8]
```

When we want to access one of the element, we use:

```
arr[x][y][z];
```

To process a multidimensional array, we need *nested looping* to access each of the layer. For example, run this 3 Dimension Array:

```
#include <stdio.h>
int main() {
    int arr[2][2][2] = {
        {
            {1, 2},
            {3, 4}
        },
        {
```

```
            {5, 6},
            {7, 8}
        }
    };
    for(int i = 0; i < 2; i++) {
        for(int j = 0; j < 2; j++) {
            for(int k = 0; k < 2; k++) {
                printf("%d ", arr[i][j][k]);
            }
            printf("\n");
        }
        printf("\n");
    }

    return 0;
}
```

Parallel Array and Indexing Method

A parallel array is a technique where multiple arrays are used together to store related data. Each array stores a different type of information, but all arrays share the same index to represent one complete data record. This method could be alternative when we only want to use one dimensional array for simpler management or in case of different data type.

Concept of Paralel Array

Instead of using a complex structure, we use multiple arrays:

```
Index :    0        1        2
Name  :  Alice    Bob     Charlie
Score :   85       90       78
```

Data at the same index represents one entity despite they are 3 different 1 dimensional array:

- Index 0 → Alice, 85
- Index 1 → Bob, 90

- Index 2 → Charlie, 78

This way, if we want to call all of the student, we could use the looping index (i) to all of the one dimension array variable.

Why should we use it?

Parallel arrays are useful when:

- You need to store related data
- Data types are different (e.g., string and integer)
- Structures (struct) have not been introduced yet

For an example, run this program to input different value to each array:

```
#include <stdio.h>

int main() {

    char names[3][20];

    int scores[3];

    for(int i = 0; i < 3; i++) {

        printf("Enter name: ");

        scanf("%s", names[i]);


        printf("Enter score: ");

        scanf("%d", &scores[i]);

    }


    printf("\nStudent Data:\n");


    for(int i = 0; i < 3; i++) {

        printf("%s - %d\n", names[i], scores[i]);

    }


    return 0;

}
```

Mini-Project: Student Score Management System

Project Tittle: Student Score Management System Using Arrays

Objective:

- Apply one-dimensional array and parallel array
- Manage and process multiple data entries
- Use looping and basic decision-making
- Build a simple real-world application

Flow of Program:

1. 3 feature -> Input, Search, Sorting
2. When clicking input, user input number of student
3. Each student have name, ID and score
4. Program doing loop for user to input the student's data
5. After done, put user back to input/search/sorting menu
6. Upon clicking search, program ask the index wanted by user
7. User input index
8. Program show the data searched by user
9. User click enter to return to main menu
10. Upon clicking sorting, the program will sort the student score based on the highest
11. Program show the newly sorted student list
12. User click enter to return to main menu

Expected Output:

Menu

```
===== MENU =====

1. Input

2. Search

3. Sort

Choose an option: 2
```

Scenario: Choosing search

```
==Index Search==

Input user index: 2
==Search Result==
Name: Budi
ID: 2234010076
Score: 80
```

Post-Test : Data Structure Basics (Array)

Evaluation Score

Instruction:

- Answer all questions.
- For multiple-choice questions, choose the best answer.
- For problem-solving questions, write your logic clearly.

Multiple-Choice

1. What is the main purpose of using an array in programming?

A. To execute loops

C. To define conditions

B. To store multiple values of the same data type

D. To print output

2. What is the index of the last element in an array of size 5?

A. 5

C. 3

B. 4

D. 1

3. What will be the output of the following code?

```
int arr[4] = {2, 4, 6, 8};
printf("%d", arr[3]);
```

A. 2

C. 6

B. 4

D. 8

4. Which of the following correctly accesses the element in row 1, column 2 of a 2D array?

A. `arr[1][2]`

C. `arr(1,2)`

B. `arr[2][1]`

D. `arr[1,2]`

5. What is the main idea of a parallel array?

A. Using one array for all data

C. Using multiple arrays with the same index representing related data

B. Using multiple arrays with unrelated data

D. Using loops inside arrays

Short Answer Questions

6. Explain the difference between one-dimensional and two-dimensional arrays.

7. Why is indexing important in arrays?

8. What is one disadvantage of using parallel arrays?

Code Analysis

09. Determine the output of the following code:

```
int arr[3] = {5, 10, 15};
printf("%d", arr[0] + arr[2]);
```

Answer:

10. What is the output? What does the index [1][1] represent?

```
int arr[2][2] = {
    {1, 2},
    {3, 4}
};

printf("%d", arr[1][1]);
```

Answer: