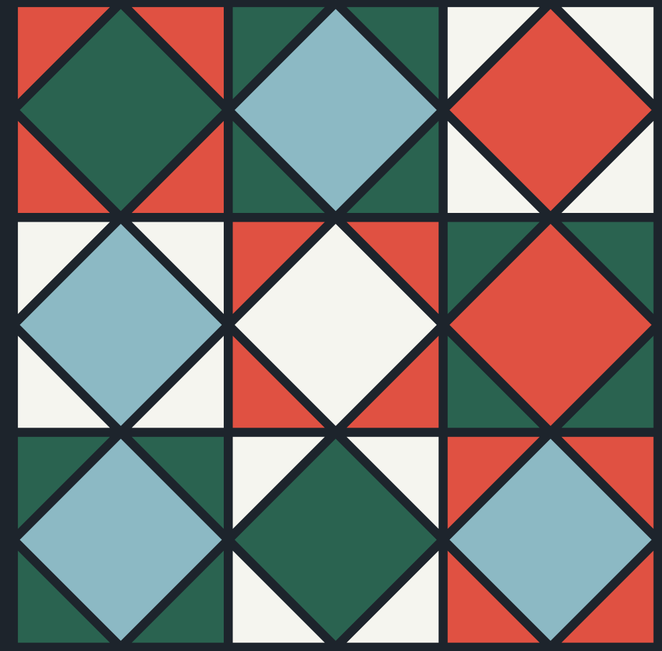
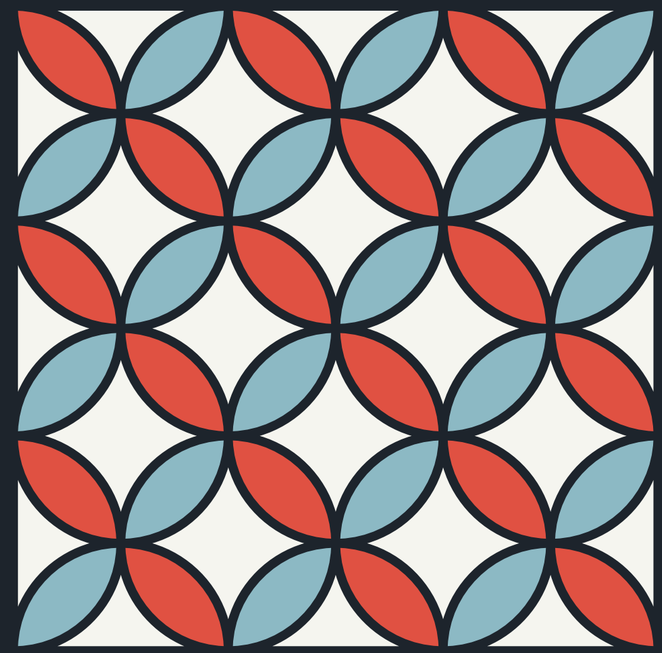
11

# DATA STRUCTURE BASICS (ARRAY)

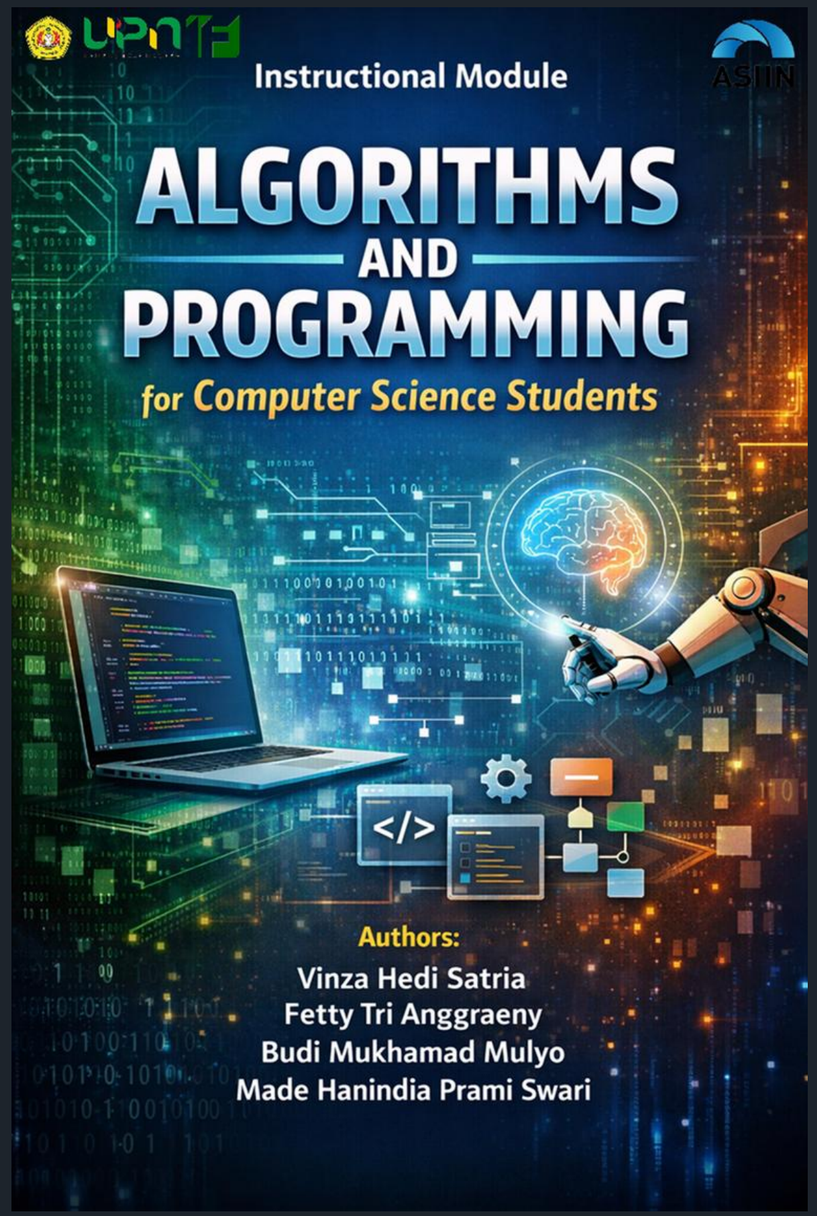




# BEFORE WE BEGIN

PLEASE FILL THIS PRE-TEST GOOGLE FORM ABOUT ARRAY

SCAN ME



# TODAY'S AGENDA

- PRE-TEST
- INTROUCTION TO ARRAY IN PROGRAMMING CONTEXT
- ONE-DIMENSIONAL ARRAY
- TWO DIMENSIONAL ARRAY
- MULTI-DIMENSIONAL ARRAY
- PARALLEL ARRAY AND INDEXING METHOD
- POST-TEST
- MINI-PROJECT: STUDENT SCORE SORTING PROGRAM

# WHAT IS AN ARRAY?

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.

# CHARACTERISTIC OF ARRAY IN C

## SAME DATA TYPE

In C, All elements of Array must be of the same type. (e.g., all integers).

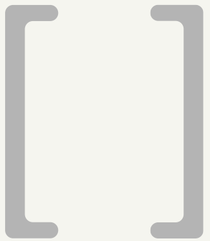
- Same Data Type

## INDEXED STRUCTURE

Each element is accessed using an index (starting from 0).

## FIXED SIZE

In C, The size of an array is defined when it is created.



## EFFICIENT ACCESS

Elements can be accessed directly using their index.

# HOW TO DECLARE ARRAY IN C?

## DECLARING THE ROW

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

By declaring the amount of rows, our array will have fixed amount of rows with empty value

## DECLARING THE VALUE

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

Or we could empty the bracket [ ] and just write the value. This way, C will automatically assumed our array have two rows

## DECLARING THE ROW AND VALUE

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

By declaring both, we could have an fixed amount of rows and some value, while the other row remain empty

# ACCESSING ARRAY

## EXAMPLE OF 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;
}
```

DECLARE

PRINT

**QUICK QUESTION! IS THIS PROGRAM EFFICIENT?**

ARRAY

# ONE DIMENSION

# WHAT IS IT?

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.

**Common usages of One Dimension Array**

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

# ONE DIMENSION ARRAY INPUT

## EXAMPLE OF PROGRAM

```
#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;
}
```

# TWO DIMENSIONAL ARRAY

TWO DIMENSIONAL ARRAY

# 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.

# DATA VISUALIZATION

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

# TO ACCESS IT

WE USE ARR[ROW][COLUMN]

# 2 DIMENSION ARRAY

## EXAMPLE OF PROGRAM

```
#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;
}
```

# TWO DIMENSIONAL ARRAY

## STRING EDITION

# ARRAY OF STRINGS IN C

FOR CHAR TYPE DATA, THERE ARE VARIOUS METHOD TO CREATE AN ARRAY


**EXAMPLE OF PROGRAM**

```
#include <stdio.h>

int main() {
    char *course[] = {"Alpro",
"Alima", "ADSI", "Game"};

    for(int i = 0; i < 4; i++)
{
        printf("%s\n",
course[i]);
    }

    return 0;
}
```


**EXAMPLE OF PROGRAM**

```
#include <stdio.h>

int main() {
    char course[4][10] = {
        "Alpro",
        "Alima",
        "ADSI",
        "Game"
    };

    for(int i = 0; i < 4; i++) {
        printf("%s\n", course[i]);
    }

    return 0;
}
```

# SO WHAT IS THE DIFFERENCE?

## WHEN USING ONE DIMENSION

THE STRING “ALPRO”, “ALIMA” AND SUCH WERE TREATED AS ONE SINGLE VALUE. (THAT STILL CAN BE SEPERATED DUE TO STRING BEHAVIOR IN C)

## WHEN USING TWO DIMENSION

WITH LENGTH OF STRING MANAGEMENT, WE ABLE TO ALLOCATE MEMORY EFFICIENTLY AND RESTRICT THE LENGTH OF THE STRING

# MULTI DIMENSION ARRAY

# 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.

# DATA VISUALIZATION

```
Layer 0:

[1 2]

[3 4]


Layer 1:

[5 6]

[7 8]
```

# TO ACCESS IT

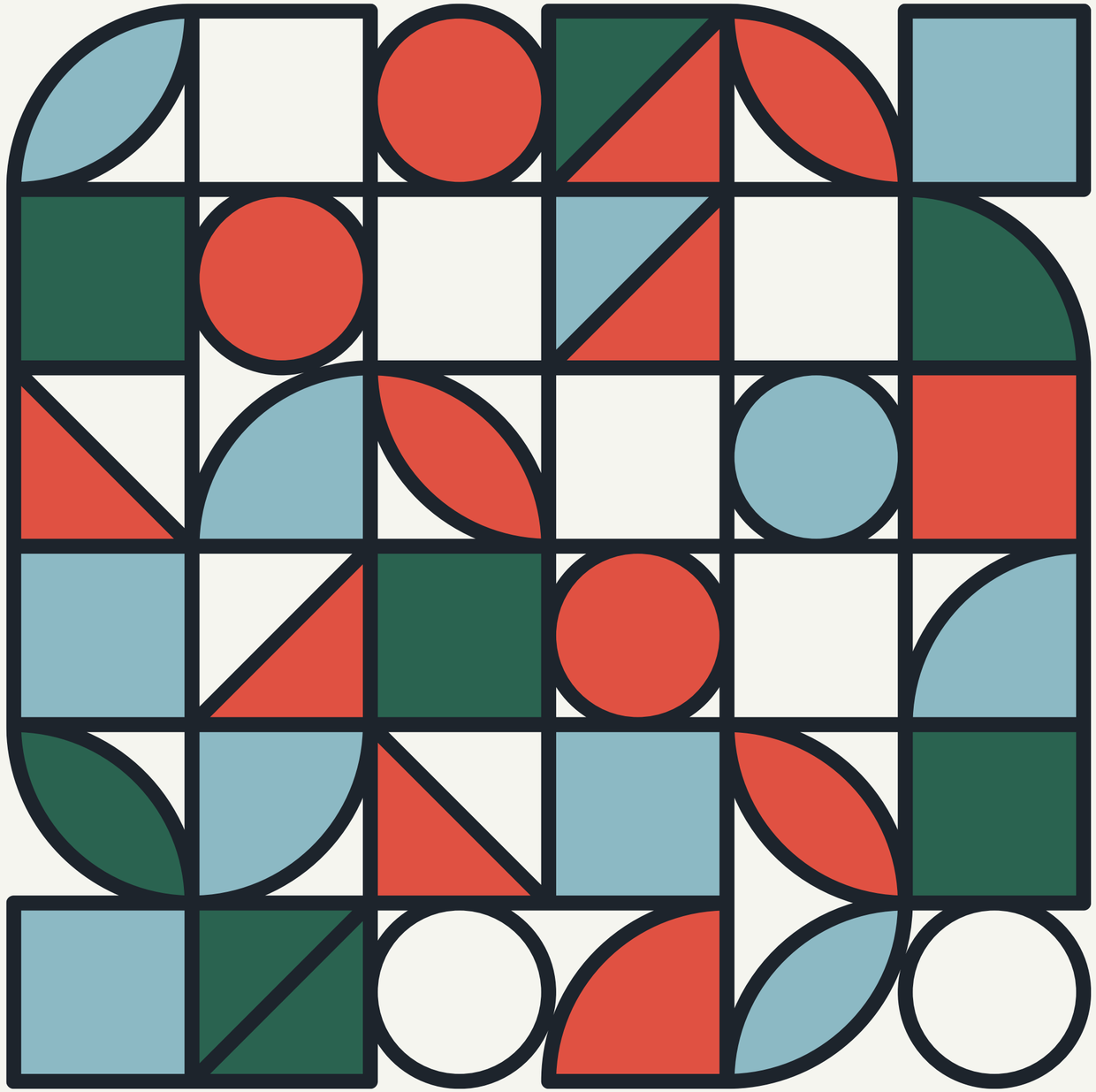
ARR[X][Y][Z];

# 3 DIMENSION ARRAY

## EXAMPLE OF PROGRAM

```
#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;
}
```



# PARRALLEL ARRAY

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 to bypass C Weakness, where they could only work on same type of data for an array

# EXAMPLE

WE HAVE 2 DIFFERENT ARRAY

## NAME

Store name of person

(string)

## SCORE

Store score of person

(Integer)

Despite being two different arrays, we ensure that each index were meant for same identity

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

# THIS WAY

We could call all of the array in one single loop

Example: we could use i for Name[i] and Score[i]

# EXAMPLE OF PARRALLEL ARRAY

```c
#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;
}
```

**TRY TO EXECUTE IT ON YOUR OWN C IDE!**

# THANK YOU!

BEFORE WE FINISH, PLEASE FILL THIS POST-TEST GOOGLE FORM!