



ARRAY

ONE DIMENSIONAL

prepared by:

Gyro A. Madrona
Electronics Engineer

TOPIC OUTLINE

Declaring an Array

Initializing Array

Accessing Array Elements



ARRAY



ARRAY

An array is a collection of elements of the same data type stored in contiguous memory locations. Arrays are used to store multiple values in a single variable, making it easier to manage and manipulate data.



DECLARING AN ARRAY

The syntax for declaring an array:

```
data_type array_name[array_size];
```

Example:

```
int number[5];
```

```
char letter[4];
```

Key Features of Arrays:

1. **Fixed Size**: The size of an array is fixed at the time of declaration and cannot be changed during runtime.



INITIALIZING AN ARRAY

Arrays can be initialized at the time of declaration.

```
data_type array_name[array_size] =  
{value_1, value_2...valueN};
```

Example:

```
int number[5] = {5,8,1,1,3};  
  
char letter[4] = {'a','b','c','d'};
```

Key Features of Arrays:

1. Fixed Size: The size of an array is fixed at the time of declaration and cannot be changed during runtime.
2. Homogeneous Elements: All elements in an array must be of the same data type (e.g., **int**, **float**, **char**, etc.).



ACCESSING ARRAY ELEMENTS

Array elements are accessed using the index.

```
array_name[index];
```

Example:

```
int number[5]={5,8,1,1,3};
```

```
int first = number[0];
```

```
int second = number[1];
```

```
int third = number[2];
```

Key Features of Arrays:

1. Fixed Size: The size of an array is fixed at the time of declaration and cannot be changed during runtime.
2. Homogeneous Elements: All elements in an array must be of the same data type (e.g., **int**, **float**, **char**, etc.).
3. Zero-Based Indexing: Array indices start from 0. The first element is at index 0, the second at index 1, and so on.



EXERCISE

Determine the output of this code snippet:

```
int number[]={4,7,1,5};
```

```
int value = 0;
```

```
value = number[3];
```

```
cout << value;
```

output:

Determine the output of this code snippet:

```
int number[]={4,7,1,5};
```

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

```
    cout << number[i]<< endl;
```

```
}
```

output:



EXERCISE

Determine the output of this code snippet:

```
int arr[]={4,7,1,5};

int sum = 0;

for(int i = 0; i < 4; i++){
    sum = sum + arr[i];
}

cout << "Sum of array elements: ";
cout << sum;
```

output:

Determine the output of this code snippet:

```
int arr[]={4,7,1,5};
int min = arr[0];
for(int i = 1; i < 4; i++){
    if(arr[i] < min){
        min = arr[i];
    }
}

cout << "Minimum element: ";
cout << min;
```

output:



sizeof OPERATOR

The sizeof() operator is a compile-time unary operator that determines the size, in bytes, of a data type, variable, or object.

Example:

```
int number = 0;  
  
int size = sizeof(number) ;  
  
// size = 4 bytes
```

```
int number[] = {0,3,1};  
  
int size = sizeof(number) ;  
  
// size = 12 bytes
```



EXERCISE

Determine the output of this code snippet:

```
int arr[]={4,7,1,5};

int size =
sizeof(arr)/sizeof(arr[0]);

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

    cout << arr[i] << endl;

}
```

output:

Determine the output of this code snippet:

```
int arr[]={4,7,1,5,8,2};

int size =
sizeof(arr)/sizeof(arr[0]);

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

    if(arr[i] % 2 == 0){

        cout << arr[i] << " ";

    }

}
```

output:



LABORATORY

