



UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT



Data Structure and Algorithm

Laboratory Activity No. 5

Implementation of Arrays

Submitted by:
Talagtag Mark Angel T.

Instructor:
Engr. Maria Rizette H. Sayo

August 16, 2025

I. Objectives

Introduction

Array, in general, refers to an orderly arrangement of data elements. Array is a type of data structure that stores data elements in adjacent locations. Array is considered as linear data structure that stores elements of same data types. Hence, it is also called as a linear homogenous data structure.

This laboratory activity aims to implement the principles and techniques in:

- Writing algorithms using Array data structure
- Writing a python program that can implement Array data structure

II. Methods

- Write a Python program to create an array of 10 integers and display the array items. Access individual elements through indexes and compute for the sum.
- Write a Python program to append a new item to the end of the array. Original array: numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- Write a Python program to insert a new item before the second element in an existing array. Original array: numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- Write a Python program to reverse the order of the items in the array. Original array: numbers = [5, 4, 3, 2, 1]

Write a Python program to get the length of the array. Original array: numbers = [5, 4, 3, 2, 1]

III. Results

1. This code print array items and give them element and index then the code sum all the number on the arrays

```
numbers = [12, 34, 56, 78, 90, 23, 45, 67, 89, 12]
print("Array items:")
for num in numbers:
    print(num)

print("\nAccessing individual elements:")
for i in range(len(numbers)):
    print(f"Element at index {i}: {numbers[i]}")

total_sum = sum(numbers)

print("\nSum of the array elements:", total_sum)
```

```
⇒ Array items:
12
34
56
78
90
23
45
67
89
12

Accessing individual elements:
Element at index 0: 12
Element at index 1: 34
Element at index 2: 56
Element at index 3: 78
Element at index 4: 90
Element at index 5: 23
Element at index 6: 45
Element at index 7: 67
Element at index 8: 89
Element at index 9: 12

Sum of the array elements: 506
```

ALGORITHM

1. All array elements printed
2. Each element accessed by index and printed
3. The sum of all elements printed

2. this code input a new number in the end of the array

```

> numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

new_item = int(input("Enter a number: "))

numbers.insert(10, new_item)

# Print the updated array
print("Updated array:", numbers)

Enter a number: 100
Updated array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100]
```

ALGORITHM

- 1. Input a number
- 2. print the array with a new number on the end

3. just like in number 2 this code was meant to input a new number between 1 and 2

```

> numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

new_item = int(input("Enter a number: "))

numbers.insert(1, new_item)

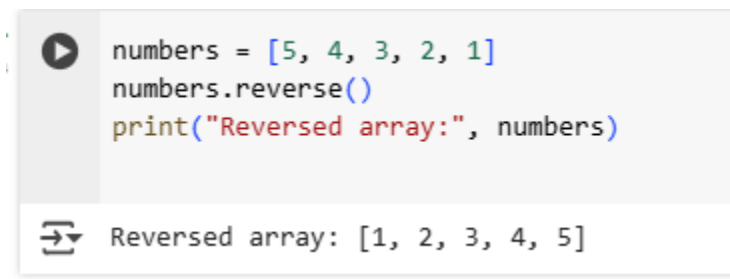
# Print the updated array
print("Updated array:", numbers)

Enter a number: 123
Updated array: [1, 123, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

ALGORITHM

- 1. Input a number
- 2. print the array with a new number between the 1 and 2

4. this code rearrange the arrays of numbers (reverse).

A screenshot of a code editor with a light gray background. On the left, there is a vertical toolbar with a play button icon. The code area contains three lines of Python code: `numbers = [5, 4, 3, 2, 1]`, `numbers.reverse()`, and `print("Reversed array:", numbers)`. Below the code, there is a console output area with a double arrow icon and the text `Reversed array: [1, 2, 3, 4, 5]`.

```
numbers = [5, 4, 3, 2, 1]
numbers.reverse()
print("Reversed array:", numbers)
```

Reversed array: [1, 2, 3, 4, 5]

- ALGORITHM
- 1. Print the array numbers
 - 2. Reverse the array numbers

Figure 1 Screenshot of program

If an image is taken from another literature or intellectual property, please cite them accordingly in the caption. Always keep in mind the Honor Code [1] of our course to prevent failure due to academic dishonesty.

IV. Conclusion

Throughout this laboratory, I learned a lot about arrays (lists) in Python and how to arrange and manipulate them. These fundamental concepts have helped me gain a better understanding of how to work with data structures in a more organized and efficient way.

One of the first things I learned was how to append new items to a list. By using the `append()` method, I realized how easy it is to add elements at the end of an array. This is useful in real-world scenarios where I may need to dynamically build lists as my program runs. It gave me a solid foundation in managing the growth of an array.

Another key lesson was about inserting elements at specific positions within the array using the `insert()` method. This was a game-changer, as I learned that I don't have to add elements only at the end but can place them exactly where I need them. This is especially helpful when dealing with ordered data or when precise data insertion is required.

References

- [1] Co Arthur O.. “University of Caloocan City Computer Engineering Department Honor Code,” UCC-CpE Departmental Policies, 2020.