

FAST-National University of Computer & Emerging Sciences Peshawar

Programming Fundamentals

Engr.Zakria Bacha

zakria.bacha@nu.edu.pk

Lab: Dynamic Memory Allocation in C++

Section 0: Array With Pointer

- 1. Pass Array to function:**
- 2. Print elements of Array through pointer**
- 3. Return the address of the array from function through pointer.**

Section 1: Introduction to Dynamic Memory Allocation

Objective:

Understand the basics of new and delete operators for dynamic memory allocation.

Concepts:

1. Dynamic memory allocation allows runtime memory allocation.
2. The new operator allocates memory, and the delete operator deallocates it

3.What is a Dynamic Array?

1. A dynamic array is quite similar to a regular array, but its size is modifiable during program runtime. DynamArray elements occupy a contiguous block of memory.

2. Once an array has been created, its size cannot be changed. However, a dynamic array is different. A dynamic array can expand its size even after it has been filled.

The new Keyword

In C++, we can create a dynamic array using the new keyword. The number of items to be allocated is specified within a pair of square brackets. The type name should precede this. The requested number of items will be allocated.

Syntax

The new keyword takes the following syntax:

```
pointer_variable = new data_type;
```

The `pointer_variable` is the name of the [pointer variable](#).

The `data_type` must be a valid C++ data type.

The keyword then returns a pointer to the first item. After creating the dynamic array, we can delete it using the delete keyword.

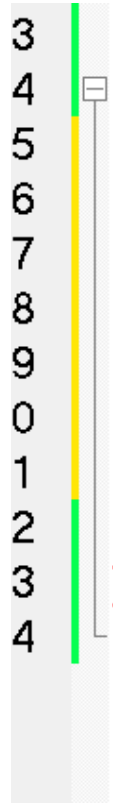
Dynamically Deleting Arrays

A dynamic array should be deleted from the computer memory once its purpose is fulfilled. The delete statement can help you accomplish this. The released memory space can then be used to hold another set of data. However, even if you do not delete the dynamic array from the computer memory, it will be deleted automatically once the program terminates.

Dangling Pointer

Problem:

Using a pointer after the memory it points to has been deallocated leads to undefined behavior.



```
3
4 int main() {
5
6     int* arr = new int[5];
7     delete[] arr;
8     arr[0] = 10; // Dangling pointer
9     delete[] arr;
10    arr = nullptr;
11
12    return 0;
13 }
```

Section 2: Dynamic Arrays

Objective:

Learn how to create and manage dynamic arrays.

Concepts:

1. Dynamic arrays are created using new with an array size.
2. Deallocate memory using
3. delete [].

```
3
4 int main() {
5     int n;
6     cout << "Enter the size of the array: ";
7     cin >> n;
8
9     int* arr = new int[n]; // Allocate memory for an array
10
11     for (int i = 0; i < n; i++) {
12         arr[i] = i + 1; // Initialize array
13     }
14
15     cout << "Array elements: ";
16     for (int i = 0; i < n; i++) {
17         cout << arr[i] << " ";
18     }
19     cout << endl;
20
21     delete[] arr; // Free memory
```

Section 3: Resizing a Dynamic Array

Objective:

Understand manual resizing of dynamic arrays.

Concepts:

1. Allocate new memory for a larger array.
2. Copy data from the old array to the new array.
3. Free old memory using delete [].

Lab Tasks:

Program Description:

1. The program takes two inputs: the first and last numbers. It calculates the size for storing all even numbers between them. A dynamic array is created to store the even numbers. A function populates the array and returns it to the main function. The program prints the array using a pointer.
2. The Education Department wants to hire 50 accountants for the Accounts section. Write a program that takes the number 50 as input and creates a dynamic array to store their IDs (from 1 to 50). After this, the MD wants to hire 50 Office Assistants for the organization. Create another function to store additional IDs (even numbers from 51 to 100)