CSE 06131223 ♦ CSE 06131224

Structured Programming

Lecture 13

Array in C (3)

Prepared by

Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering

Jatiya Kabi Kazi Nazrul Islam University, Bangladesh

www.mijanrahman.com

Contents

ARRAY IN C

Dynamic Array

- A dynamic array, also known as a resizable array or growable array, is a data structure that allows elements to be stored in contiguous memory locations and **provides the ability to resize the array dynamically during program execution.**

- **Static vs. Dynamic**
- Unlike static arrays, where the size is fixed at compile time, **dynamic arrays can grow or shrink as needed**, enabling more flexibility in handling varying amounts of data.

How Dynamic Array Works?

- Here's how a typical dynamic array works:
  1. **Initialization:** Initially, a dynamic array is created with a certain capacity (or initial size).
  2. **Adding Elements:** When an element is added to the array, it dynamically allocates a new block of memory.
  3. **Accessing Elements:** Elements in a dynamic array are typically accessed using indices, just like in static arrays.
  4. **Resizing:** Dynamic arrays resize themselves automatically when necessary.
  5. **Removing Elements:** Similarly, when elements are removed from a dynamic array, the array may shrink its capacity to save memory.

How Dynamic Array Works?

- **Initialization:**
- **Initially, a dynamic array is created with a certain capacity (or initial size).** This capacity is usually larger than the number of elements currently stored in the array to accommodate future growth without needing to resize the array frequently.

- **Adding Elements:**
- **When an element is added to the dynamic array and the current capacity is exceeded, the array dynamically allocates a new block of memory with a larger capacity (often doubling the current capacity is a common strategy).** Then, it copies the existing elements into the new memory block and inserts the new element. This process is called resizing or reallocating the array.

How Dynamic Array Works?

- **Accessing Elements:**
- Elements in a dynamic array are typically accessed using indices, just like in static arrays. Since elements are stored in contiguous memory locations, accessing elements by index is efficient.

- **Resizing:**
- Dynamic arrays resize themselves automatically when necessary, ensuring that there is enough space to accommodate additional elements.
- However, resizing operations can be relatively expensive because they involve allocating new memory, copying elements, and deallocating the old memory block. To minimize the frequency of resizing and the associated overhead, dynamic arrays often increase their capacity by a certain factor (e.g., doubling the capacity) each time they need to resize.

How Dynamic Array Works?

- **Removing Elements:**
- When elements are removed from a dynamic array, and the number of elements becomes significantly smaller than the current capacity, the array may shrink its capacity to save memory.
- However, many dynamic array implementations do not shrink the capacity immediately to avoid frequent resizing operations due to minor fluctuations in the number of elements.

Dynamic Array Implementation in C

- In C, dynamic arrays are typically implemented using **pointers and dynamic memory allocation functions like malloc(), realloc(), and free().**
- Unlike static arrays whose size is fixed at compile time, dynamic arrays allow you to allocate memory dynamically at runtime, enabling you to resize them as needed.

Dynamic Array Implementation in C

1. Allocation:

- To create a dynamic array, you first allocate memory for the array using the malloc() function. For example:

```
int *dynamicArray = malloc(initialSize * sizeof(int));
```

- This allocates memory for an array of **initialSize** integers and returns a pointer to the first element of the array. If the allocation is successful, **dynamicArray** will point to the beginning of the allocated memory block.

Dynamic Array Implementation in C

2. Accessing Elements:

- Elements in the dynamic array can be accessed using array notation or pointer arithmetic, just like in static arrays. For example:

```
dynamicArray[0] = 10;                // Assigning a value to the first element
int element = dynamicArray[0];       // Accessing the value of the first element
```

Dynamic Array Implementation in C

3. Resizing:

- To resize the dynamic array to accommodate more elements, we can use the **realloc() function**. This function allows to change the size of the previously allocated memory block. For example:

```
dynamicArray = realloc(dynamicArray, newSize * sizeof(int));
```

- This reallocates memory for the dynamic array to accommodate **newSize** elements. It may either resize the existing memory block or allocate a new block and copy the existing elements to it, depending on the available memory and the implementation of **realloc()**.
- The contents of the original array are preserved up to the minimum of the old and new sizes.

C Program using Dynamic array:

A simple C program that demonstrates the use of dynamic arrays to store integers.

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int size;
    printf("Enter the size of the dynamic array: ");
    scanf("%d", &size);

    // Allocate memory for the dynamic array
    int *dynamicArray = (int*)malloc(size * sizeof(int));
    // Check if memory allocation is successful
    if(dynamicArray == NULL) {
        printf("Memory allocation failed. Exiting...\n");
        return 1;
    }

    printf("Enter %d integer values:\n", size);
    for(int i = 0; i < size; i++) {
        scanf("%d", &dynamicArray[i]);
    }

    // Print the array elements
    printf("Array elements:\n");
    for(int i = 0; i < size; i++) {
        printf("%d ", dynamicArray[i]);
    }

    // Free the dynamically allocated memory
    free(dynamicArray);

    return 0;
}
```

Terminal

```
Enter the size of the dynamic array: 2
Enter 2 integer values:
12
23
Array elements:
12 23
```

C Program using Dynamic array:

This program demonstrates dynamic memory allocation to store names entered by the user.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    char **names = NULL;
    char input[50];
    int size = 0;

    printf("Enter names (type 'done' to stop):\n");
    while (1) {
        printf("Name %d: ", size + 1);
        scanf("%s", input);
        if (strcmp(input, "done") == 0) {
            break;
        }
        // Allocate memory for the new name
        names = (char **)realloc(names, (size + 1) * sizeof(char *));
        names[size] = (char *)malloc((strlen(input) + 1) * sizeof(char));
        strcpy(names[size], input);
        size++;
    }

    printf("\nEntered names:\n");
    for (int i = 0; i < size; i++) {
        printf("%d: %s\n", i + 1, names[i]);
        free(names[i]); // Free memory allocated for individual names
    }
    free(names); // Free memory allocated for the array of names

    return 0;
}
```

Terminal

```
Enter names (type 'done' to stop):
Name 1: Rahman
Name 2: Abdur
Name 3: done
Entered names:
1: Rahman
2: Abdur
```

C Program using Dynamic array:

This program prompts the user to enter integers dynamically, calculates their average, and displays it.

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *numbers = NULL;
    int size = 0;
    int sum = 0;
    float average;

    printf("Enter integers (enter 0 to stop):\n");
    while (1) {
        int input;
        printf("Integer %d: ", size + 1);
        scanf("%d", &input);
        if (input == 0) {
            break;
        }
        // Allocate memory for the new number
        numbers = (int *)realloc(numbers, (size + 1) * sizeof(int));
        numbers[size] = input;
        size++;
    }

    for (int i = 0; i < size; i++) {
        sum += numbers[i];
    }
    if (size > 0) {
        average = (float)sum / size;
        printf("\nAverage: %.2f\n", average);
    } else {
        printf("\nNo numbers entered.\n");
    }
    free(numbers);
    return 0;
}
```

Terminal

```
Enter integers (enter 0 to stop):
Integer 1: 10
Integer 2: 20
Integer 3: 30
Integer 4: 0
Average: 20.00
```

C Program using Dynamic array:

This program reads a string from the user, stores it in a dynamic array, reverses the string, and prints the reversed string.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    char *input = NULL;
    int length = 0;

    printf("Enter a string: ");
    char ch = getchar();
    while (ch != '\n') {
        input = (char *)realloc(input, (length + 1) * sizeof(char));
        input[length++] = ch;
        ch = getchar();
    }
    input = (char *)realloc(input, (length + 1) * sizeof(char));
    input[length] = '\0';
    printf("Original string: %s\n", input);
    // Reverse the string
    for (int i = 0; i < length / 2; i++) {
        char temp = input[i];
        input[i] = input[length - i - 1];
        input[length - i - 1] = temp;
    }
    printf("Reversed string: %s\n", input);

    free(input);
    return 0;
}
```

Terminal

```
Enter a string: UNIVERSITY
Original string: UNIVERSITY
Reversed string: YTISREVINU
```

THE END