Dynamic Arrays

In this lesson, you will learn how to allocate the memory to an array dynamically.

We'll cover the following

Dynamic arrays

In the arrays lesson, we discussed the static arrays. In a static array, a fixed amount of memory is allocated to the array during the compile time. Therefore, we cannot allocate more memory to the arrays during program execution.

Suppose we have declared an array that can store five integer values.

What if we want to store more than five values in an array? Here is where, dynamic arrays come in!

***Dynamic arrays** can grow or shrink during the program execution.*

Declaration

The general syntax for declaring dynamic arrays is given below:

```
DataType  *ArrayName  = new DataType [size] ;
```

Initialization

We can initialize the dynamic array just like the static array.

ArrayName [Index] = Value ;

Deallocating array

The basic syntax for deallocating a dynamic array is given below:

delete [ ] ArrayName ;

Print all elements of a dynamic array

The code will initialize the dynamic array and then print all its elements using for loop.

```
#include <iostream>

using namespace std;

int main(){
  int size = 5;
  //Declare dynamic array
  int *Array = new int[size];
  //Initialize dynamic array
  for(int i = 0 ; i< size; i++){
    Array[i] = i;
  }
  //Prints dynamic array
  for(int i = 0 ; i< size; i++){
    cout << Array[i] << " ";
  }
 // Deletes a memory allocated to dynamic array
   delete[] Array;
}
```

Line No. 8: Reserves space for 5 integers and stores the starting address of allocated space in a pointer `Array`

Line No. 10: Traverses an array and initializes its elements

Line No. 14: Traverses an array and prints its value

Line No. 18: Frees the space pointed by an `Array`

Resizing dynamic array

Let's write a program in which we will increase the size of an array and store some new elements.

```
#include <iostream>

using namespace std;

// printArray function
void printArray(int arr[], int size) {
  for (int i = 0; i < size; i++) {
    cout << arr[i] << " ";
  }
  cout << endl;
}

// main function
int main() {
  // Initialize variable size
  int size = 5;
  // Create Array
  int * Arr = new int[size];
  // Fill elements of an array
  for (int i = 0; i < size; i++) {
    Arr[i] = i;
  }
  // Call printArray function
  printArray(Arr, size);
  // Create new array
  int * ResizeArray = new int[size + 2];
  // Copy elements in new arary
  for (int i = 0; i < size; i++) {
    ResizeArray[i] = Arr[i];
  }
  // Delete old array
  delete[] Arr;
  // Pointer Array will point to ResizeArray
  Arr = ResizeArray;
  // Store new values
  Arr[size] = 90;
  Arr[size + 1] = 100;
  // Call printArray function
  printArray(Arr, size + 2);
}
```

Explanation

Suppose we want to resize the already declared array `Arr` in a program. We will follow the following steps:

Line No. 26: Creates a new array `ResizeArray` with the new size

Line No. 28: Copies the elements of the old array `Arr` into the new array `ResizeArray`

Line No. 32: We don't need the memory pointed by the `Arr` anymore. Therefore, we delete it

Line No. 34: We still want our array to be called `Arr`. Therefore we change the pointer.

Line No. 36: Stores the new values in an array `Arr`

Tada! We have just resized the original array using pointers.

Quiz

Q

Which of the following statement creates a dynamic array of type `double` and `size = 20`?

Retake Quiz

That’s all about dynamic memory allocation. Let’s get our hands dirty on the few coding challenges in the upcoming lesson.