Data Structures

BS (CS) _Fall_2024

Lab_02 Manual

Learning Objectives:

1. Arrays
2. Arrays with Templates

Lab Manual 02

Arrays

In C++, an array is a data structure that is used to store multiple values of similar data types in a contiguous memory location.

Initialization of Array in C++

In C++, we can initialize an array in many ways, but we can initialize an arrayat the time of declaration or after declaration.

1. Initialize array with values in C++

```
int arr[5] = {1, 2, 3, 4, 5};
```

Array with template

```
#include <iostream>
using namespace std;

template <typename T, size_t N>
void printArrayElements(const T (&arr)[N]) {
    cout << "Array Elements: ";
    for (size_t i = 0; i < N; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    int arr[5] = {4, 8, 7, 3, 6};
    printArrayElements(arr);

    double arr2[4] = {1.1, 2.2, 3.3, 4.4};
    printArrayElements(arr2);

    char arr3[3] = {'A', 'B', 'C'};
    printArrayElements(arr3);

    return 0;
}
```

2. Initialize array without size

```
int arr[] = {1, 2, 3, 4, 5};
```

```cpp
#include <iostream>
using namespace std;

template <typename T, size_t N>
void printArrayElements(const T (&arr)[N]) {
    cout << "Array Elements: ";
    for (size_t i = 0; i < N; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    int arr[] = {4, 8, 7, 3, 6}; // No need to specify size
    printArrayElements(arr);

    double arr2[] = {1.1, 2.2, 3.3, 4.4}; // No need to specify size
    printArrayElements(arr2);

    char arr3[] = {'A', 'B', 'C'}; // No need to specify size
    printArrayElements(arr3);

    return 0;
}
```

3. Passing arrays to function

Passing array as a pointer

```
return_type function_name ( data_type *array_name ) {
          // set of statements
}
```

```cpp
#include <iostream>
using namespace std;

template <typename T>
void printArrayElements(T* arr, int size) {
    cout << "Array Elements: ";
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    int arr[] = {4, 8, 7, 3, 6};
    int size = sizeof(arr) / sizeof(arr[0]);
    printArrayElements(arr, size);
    double arr2[] = {1.1, 2.2, 3.3, 4.4};
    int size2 = sizeof(arr2) / sizeof(arr2[0]);
    printArrayElements(arr2, size2);

    return 0;
}
```

Passing array as unsized array

return_type *function_name* **(data_type array_name[]**) {
// set of statements
}

```
#include <iostream>
using namespace std;

template <typename T>
void printArrayElements(T arr[], int size) {
    cout << "Array Elements: ";
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    int arr[] = {4, 8, 7, 3, 6};
    int size = sizeof(arr) / sizeof(arr[0]);
    printArrayElements(arr, size);

    double arr2[] = {1.1, 2.2, 3.3, 4.4};
    int size2 = sizeof(arr2) / sizeof(arr2[0]);
    printArrayElements(arr2, size2);

    return 0;
}
```

Passing array as sized array

```
return_type  function_name(data_type  array_name[size_of_array]){
          //  set  of  statements
}
```

```
#include <iostream>
using namespace std;

template <typename T, size_t N>
void ArrayElements(T (&arr)[N]) {
    cout << "Array Elements: ";
    for (size_t i = 0; i < N; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    int arr[] = {4, 8, 7, 3, 6}; // Array of size 5
    ArrayElements(arr);

    double arr2[] = {1.1, 2.2, 3.3}; // Array of size 3
    ArrayElements(arr2);

    return 0;
}
```

Defining the array class template

Class templates in C++ allow you to define classes that can operate with anydata type.

Header file:

```
#ifndef ARRAY_H_
#define ARRAY_H_

#include <iostream> // Missing include
#include <iomanip> // Include for setw
#include <typeinfo> // Include for typeid

using std::cout;
using std::endl;
using std::setw;

// Define a class template array of type T
// The type is not known yet and will be
// defined by instantiation of an object
// of the class array from main
template<typename T>
class array {
private:
    int size;
    T* myarray;

public:
    // Constructor with user-defined size
    array(int s) {
        size = s;
        myarray = new T[size];
    }
```

```cpp
    // Destructor to free allocated memory
    ~array() {
        delete[] myarray;
    }

    // Member function to set an element of myarray
    void setArray(int elem, T val) {
        if (elem >= 0 && elem < size) {
            myarray[elem] = val;
        } else {
            cout << "Index out of bounds!" << endl;
        }
    }

    // Member function to display all elements of the array
    void getArray() const {
        for (int j = 0; j < size; j++) {
            // typeid will retrieve the type for each value
            cout << setw(7) << j << setw(13) << myarray[j]
                 << " type: " << typeid(myarray[j]).name() << endl;
        }
        cout << "------------------------------------------------------" << endl;
    }
};

#endif
```

Cpp file

```cpp
#include "ArrayClass_Template.h"
int main()
{
// instantiate int_array object of class array with size 3
array< int > int_array(3);
// set value to a first element
// call to array class member function to set array elements
int_array.setArray(0,2);
// set value to a second element
int_array.setArray(1,3);
// set value to a third element
// NOTE: any attempt to set float to an int array will be translated to int value
int_array.setArray(2,4.4);

// call to array class member function to display array elements
int_array.getArray();

// instantiate float_array object of class array with size 3
array< float > float_array(3);
```

```
// set value to a first element
// call to array class member function to set array elementsfloat_array.setArray(0,3.4);
// set value to a second element
float_array.setArray(1,2.8);

// call to array class member function to display array elementsfloat_array.getArray();

// instantiate float_array object of class array with size 5array< char >
char_array(5);

// set value to a first element
// call to array class member function to set array elementschar_array.setArray(0,'H');
// set value to a other array elements
char_array.setArray(1,'E'); char_array.setArray(2,'L');
char_array.setArray(3,'L'); char_array.setArray(4,'O');

char_array.getArray();
```

Lab Task

Q1: Write a template in C++ program to find the most frequent element in an array of integers.

Prototype: T findMostFrequentElement(T arr[], int size)

Q2: Write a template for C++ program to rearrange a given sorted array of positive integers. Note: In final array, first element should be maximum value, second minimum value, third second maximum value , fourth second minimum value, fifth third maximum and so on.

Prototype: void rearrangeArray(T arr[], int size)

Q3. Write a C++ Program to find Sum of two equal length Array using function template. Also do the unit test for this part.

Prototype: T* sumOfArrays(const T arr1[], const T arr2[], int size)

Q4 Define a template Container in C++ program to sort a given unsorted array of integers, in wave form. Note: An array is in wave form when array[0] >= array[1] <= array[2] >= array[3] <= array[4] >= . . . .

Prototype: class Container

```
{
Private:
        T *arr;
        Int size;
public:

T* sortInWaveForm()

};
```