Object-Oriented Programming Lab

SPRING 2024

LAB 02

FAST National University of Computer

and Emerging Sciences

Learning Outcomes

In this lab you are expected to learn the following:

- Pointers
- Dynamic Memory Allocation

POINTERS

A pointer is a variable that stores an address of a memory location.

Pointer Declaration and initialization

Pointers is declared as follow.

Type* <variable Name>;

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

int main(){

    int *x=NULL;
    // or
    int* x=NULL;
    //or
    int * x=nullptr;

    return 0;
}
```

Address-of operator (&)

The ampersand, **&**, called the **address of operator**, is a unary operator that returns the address of its operand.

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

int main(){

    //declare and initialize string* variable
    string *str=nullptr;
    //declare and initialize string variable
    string temp="hello";
    //store the address of temp to str, both will
    //start pointing to same memory location
    str=&temp;


    //print the address of temp
    cout<<"temp Address "<<&temp<<endl;

    //print value in str
    cout<<"str Value "<<str<<endl;

    return 0;
}
```

```
temp Address 0x7ffdc2d7c440
str Value 0x7ffdc2d7c440
```

Dereference operator (*)

The unary operator * is the dereferencing operator that returns the value of its operand.

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

int main(){

    //declare and initialize string* variable
    string *str=nullptr;
    //declare and initialize string variable
    string temp="hello";
    //store the address of temp to str, both will
    //start pointing to same memory location
    str=&temp;


    //print the address of temp
    cout<<"temp Value "<< temp<<endl;

    //Dereferencing to print value in str
    cout<<"str Value "<< *str<<endl;

    return 0;
}
```

```
temp Value hello
str Value hello
```

Array manipulation using pointer

The identifier of the array is a constant pointer. The elements of the array can be accessed using dereferencing in the following manner.

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

int main(){

    int arr[3]={2,4,8};
    int *p=arr;

    cout<<arr[1]<<endl;
    cout<< *( p+1 )<<endl;

    return 0;
}
```

```
4
4
```

Note: You are supposed to pass the test cases for all functions.

Problem 1.

Write a program that stores two values in the variables 'a' and 'b' respectively. Then declare integer pointers named ptrA and ptrB. Assign the values of 'a' and 'b' to ptrA and ptrB respectively, and display them.

Function Prototype: string assignAndDisplayValues(int, int)

Problem 2.

Write a function named as calculateGardes which takes an array and its size as function parameters. You are required to apply following grading policy. **Note:** You have to release the memory using delete operator before the end of program.

Function Prototype: char *calculateGrades(int arr[ ],int size)

Score	Grade
91-100	A
76-90	B
60-75	C
51-59	D

Below 50	F

Problem 3.

Write a function named as sortArray which takes an array and its size as function parameters, you are required to sort the array such that odd numbers come first and then even numbers. Odd and even values should be reordered within their interval i-e in the example given below if 6 comes before 4 in original array, it must be sort in the sorted array. Use pointer notation to solve this problem.

Function Prototype: int* sortArray(int *arr, int size);

Example:

Input: a=6, 3, 1, 2, 7, 5, 9, 4; Output: a=1, 3, 5, 7, 9, 2, 4, 6;

Problem 4:

You have a program that uses a dynamic array to store integers. The user is allowed to input integers until they enter a specific sentinel value, say -1. Implement a scenario where the dynamic array resizes itself when it reaches its current capacity. Ensure the program doesn't lose any data during the resizing process.

Function Prototype: int resizeArray(arr, capacity);

Problem 5:

Write a function named as rotateArray that takes an array of type integer, its size, input n which shifts elements of an array to the right n times, and a number m which divide array into m parts as function parameters. You first need to need to divide array into m number of parts and then shifts elements of an array to the right n times. Array under consideration is circular.

Function Prototype: int* rotateArray (int *arr, int sizeofArray, int n, int m)

Example:

Input:

1	2	3	4	5	6

Output: Input n=2, m=2

2	3	1	5	6	4

Submission Instructions:

- Create a new folder with the name ROLLNO_SEC_LAB01 e.g. **i23XXXX_SEC_LAB02**.
- Move your header.h and debugging.cpp file to this newly created directory and compress it into a .zip file.
- Now you have to submit this zipped file on Google Classroom.

- If you don’t follow the above-mentioned submission instructions, you will be marked zero.
- Plagiarism in the Lab Task will result in zero marks in the whole category