GECG10069 (561085) F25: Introduction to Programming (C++)

Lab 12: Pointer

What you will learn from Lab 12

In this laboratory, you will understand how to use C++ pointers.

TASK 11-1 : POINTER

✔ A pointer is the memory address of a variable.

✔ For a variable ***n***, there are two parts in this variable: the ***value*** and the ***address***. Program `lab11-1` explains the concept of pointers.

```
//File: lab11-1.cpp
#include <iostream>
using namespace std;

int main()
{
    int n = 10;
    cout << " The value of n is    "  << n  << endl;
    cout << " The address of n is " << &n << endl;
    return 0;
}
```

➢ ***&n*** takes the *address* of variable ***n***.

✔ The address can be stored in a variable, called ***pointer variable***. Program `lab11-2` introduces the usage of the pointer variables.

```
//File: lab11-2.cpp
#include <iostream>
using namespace std;

int main()
{
    int n = 10;
    int *p = &n;  // p is a pointer variable

    cout << " The address of n is "  << &n << endl;
    cout << " The value of p is "     << p << endl;
    cout << " The address of p is "  << &p << endl;
    cout << " The variable pointed by p is " << *p << endl;
    return 0;
}
```

➢ `int *p = &n;` means that the point variable ***p*** stores the address of variable ***n***. Then, you can observe that "the value of ***p*** = the address of ***n***"

➢ ***&p*** means that the *address* of pointer variable ***p***.

➢ ****p*** means the variable *pointed* by ***p***.

✔ Program `lab11-3` shows the example of pointer manipulations.

```
//File: lab11-3.cpp
#include <iostream>
using namespace std;

int main()
{
    int n = 40;
    int *p1, *p2;

    cout << " n = " << n << endl;

    p1 = &n;
    *p1 = 60;
    cout << " n = " << n << endl;

    p2 = p1;
    *p2 = 50;
    cout << " n = " << n << endl;


    int arr[3] = {2, 3, 5};
    p1 = arr;
    cout << " p1 = " << *p1 << endl;
    cout << " p1+1 = " << *(p1+1) << endl;
    p1 = arr+2;
    cout << " p1 = " << *p1 << endl;

    return 0;
}
```

TASK 11-2 : NEW OPERATOR

✔ ***new*** operator (`new T ()`) is used to allocate a block of raw storage of *`sizeof(T)`*.

✔ Program `lab11-4` shows an example of using the ***new*** operator.

```
//File: lab11-4.cpp
#include <iostream>
using namespace std;

int main()
{
    int *p1 = new int;
    int *p2 = new int(1024);

    cout << "*p1 = " << *p1 << endl;
    cout << "*p2 = " << *p2 << endl;

    delete p1;
    delete p2;
    return 0;
}
```

➢ `int *p = new int;` is used to allocate a memory space with size *`sizeof(int)`* .

- `int *p = new int(1024);` is used to allocate a memory space with size *`sizeof(int)`* and assign the initial value to the memory space.
- If you allocate a memory and do not delete it correctly, you'll have a memory leakage problem.

EXERCISE 12-1 : FIND A VALUE USING A POINTER

Description -

Write a C++ program that:

1. Reads an integer `N`, followed by `N` integers stored in an array.
2. Reads an integer `target`
3. Implements a function:

```
int* findPtr(int* arr, int N, int target)
```

This function should:

- Use **pointer arithmetic** (like `p++` or `*(arr+i)`) to search the array
- Return a pointer to the element if it is found
- Return `nullptr` if it is not found

In the main program:

- If the pointer is not `nullptr`, print: **`Found at index X`**
- Otherwise print: **`Not found`**

Sample Test Cases -

Sample Input - 1
5 3 7 2 5 8 5
Sample Output - 1
Found at index 3

Sample Input - 2
6 6 5 4 3 2 1 7
Sample Output - 2
Not found

Sample Input - 3
6 6 5 4 3 2 1 4
Sample Output - 3
Found at index 2

EXERCISE 12-2 : REVERSE AN ARRAY USING POINTERS

Description -

Write a C++ function:

```
void reverseArray(int* left, int* right)
```

The function reverses the array in-place using only pointer operations:

- Swap `*left` and `*right`
- Move pointers inward (`left++`, `right--`)
- Stop when the pointers meet or cross

In main(), read N and an array of N integers,
call this function, and print the reversed array.

Sample Testcases -

Sample Input - 1
5 1 3 5 7 9
Sample Output - 1
9 7 5 3 1

Sample Input - 2
6 3 5 7 6 5 4
Sample Output - 2
4 5 6 7 5 3