Lab 13

Introduction to Pointers in C++

https://github.com/mmujtaba25/cs-110

Muhammad Mujtaba

CMD ID: 540040

mmujtaba.bese25seecs@seecs.edu.pk

Class: BESE 16B

Batch: 2k25

Table of Contents

Task 1:

GIVEN CODE:

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

int main()
{
    int x = 10;
    int *p = &x;

    cout << "Value of x: " << x << endl;
    cout << "Address of x: " << &x << endl;
    cout << "Value stored in pointer p: " << p << endl;
    cout << "Value using dereferencing *p: " << *p << endl;

    return 0;
}
```

OUTPUT:

```
obscure@Obscures-MacBook-Air output % ./"task1"
Value of x: 10
Address of x: 0x16fd3a988
Value stored in pointer p: 0x16fd3a988
Value using dereferencing *p: 10
obscure@Obscures-MacBook-Air output %
```

QUESTION:

What have you learned?

The “*” symbol in C++ represents pointer of a variable, i.e. the address of that variable in memory.

We can use the “**&**” symbol to get the address of any variable.

To get the value from a location we dereference it using the operator “*” before the variable name.

Task 2:

CODE:

```cpp
#include <iostream>

int main()
{
    float var;
    float *ptr_var = &var;

    *ptr_var = 3.14;

    std::cout << "Value using variable: " << var << "\n";
    std::cout << "Value using pointer:  " << *ptr_var << "\n";
    return 0;
}
```

OUTPUT:

```
obscure@Obscures-MacBook-Air output % ./"task2"
Value using variable: 3.14
Value using pointer:  3.14
obscure@Obscures-MacBook-Air output %
```

Task 3:

GIVEN CODE:

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

int main()
{
    int arr[5] = {1, 2, 3, 4, 5};
    int *p = arr; // Points to first element

    cout << "Array using pointer: ";
    for (int i = 0; i < 5; i++)
    {
        cout << *(p + i) << " ";
    }

    return 0;
}
```

OUTPUT:

```
obscure@Obscures-MacBook-Air output % ./"task3"
Array using pointer: 1 2 3 4 5 %
obscure@Obscures-MacBook-Air output %
```

QUESTION:

What have you learned?

In C++, an array is just a pointer to the first element in the array. Therefore we can treat and array as a pointer.

When we add an integer to a pointer, we don't move 1 byte in memory but, N bytes in memory, where N is the number of bytes of the datatype, for "**int**" it is 4 bytes.

The above process is called **Pointer Arithmetic** and is used to access elements of an array, when a clear size is given.

Task 4:

CODE:

```cpp
#include <iostream>

int main()
{
    int arr[5] = {10, 20, 30, 40, 50};

    for (int i = 0; i < 5; i++)
    {
        std::cout << *(arr + i) << " ";
        *(arr + i) += 5;
    }

    std::cout << "\n += 5 for all\n";

    for (int i = 0; i < 5; i++)
    {
        std::cout << *(arr + i) << " ";
    }

    std::cout << "\n";
    return 0;
}
```

OUTPUT:

```
obscure@Obscures-MacBook-Air output % ./"task4"
10 20 30 40 50
 += 5 for all
15 25 35 45 55
obscure@Obscures-MacBook-Air output %
```

Task 5:

GIVEN CODE:

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

int findMax(int *arr, int size)
{
    int maxVal = arr[0];

    for (int i = 1; i < size; i++)
    {
        if (*(arr + i) > maxVal)
        {
            maxVal = *(arr + i);
        }
    }
    return maxVal;
}

int main()
{
    int numbers[5] = {9, 4, 11, 3, 2};

    cout << "Maximum value: " << findMax(numbers, 5);

    return 0;
}
```

OUTPUT:

```
obscure@Obscures-MacBook-Air output % ./"task5"
Maximum value: 11%
obscure@Obscures-MacBook-Air output %
```

QUESTION:

What have you learned?

One standard for passing array into a function is to pass the address of first element (only array name, since equivalent) and the size of the array. Using this info we have access to all contents of the array.

Task 6:

CODE:

```cpp
#include <iostream>

int *getSmallest(int *arr, int size);

int main()
{
    constexpr int SIZE = 10;
    int arr[SIZE] = {-5, 3, 0, -2, 8, 7, -1, 4, 6, 2};

    int *minPtr = getSmallest(arr, SIZE);

    std::cout << "Smallest value: " << *minPtr << std::endl;
    std::cout << "Address of smallest value: " << minPtr << std::endl;
    return 0;
}

int *getSmallest(int *arr, int size)
{
    int *minPtr = &arr[0];
    for (int i = 1; i < size; i++)
    {
        if (*(arr + i) < *minPtr)
        {
            minPtr = &arr[i];
        }
    }
    return minPtr;
}
```

OUTPUT:

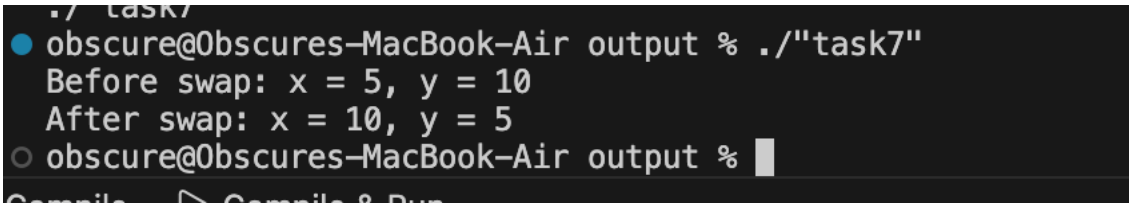
```
obscure@Obscures-MacBook-Air output % ./"task6"
Smallest value: -5
Address of smallest value: 0x16f61e950
obscure@Obscures-MacBook-Air output %
```

Task 7:

GIVEN CODE:

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

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main()
{
    int x = 5, y = 10;

    cout << "Before swap: x = " << x << ", y = " << y << endl;
    swap(&x, &y); // Passing addresses to swap
    cout << "After swap: x = " << x << ", y = " << y << endl;

    return 0;
}
```

OUTPUT:



QUESTION:

What have you learned?

We can pass pointers to variables into a function, and modify the values of these variables by dereferencing the pointers.

Task 8:

CODE:

```cpp
#include <iostream>

void swapNumbers(int *a, int *b);

int main()
{
    int x = 15;
    int y = 30;

    std::cout << "Before: x = " << x << ", y = " << y << "\n";

    swapNumbers(&x, &y);

    std::cout << "After: x = " << x << ", y = " << y << "\n";

    return 0;
}

void swapNumbers(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

OUTPUT:

```
obscure@Obscures-MacBook-Air output % ./"task8"
Before: x = 15, y = 30
After: x = 30, y = 15
obscure@Obscures-MacBook-Air output %
```