Programming Fundamental

Lab Manual - Week 14

Introduction

Welcome Back to your favorite Programming Lab students. In this lab manual, we shall work together to learn and implement new programming concepts.

Skill: Learning the use of Pointers and Reference Variables

Let's do some coding.

We can use the reference and pointer variables to store the addresses of the variables. This help us solve various problems including

- Returning multiple values from a function
- Accessing addresses of the variables for multiple reasons

Consider the following tasks for better understanding.

Task 01(WP): Write a program that creates a pointer to an integer variable number and displays the address of the number variable.

```
#include <iostream>
using namespace std;
main()
{
    int number = 10;
    int *p = &number;
    cout << "The address of variable: " << p << endl;

}
```

```
The address of variable: 0x93c29ff9c4
```

We can also use this pointer varibale to access the value of the variable whose address is stored inside that pointer variable.

Task 02(WP): Write a program that creates a pointer to an integer variable number and displays the address and value of variable.

```cpp
#include <iostream>
using namespace std;
main()
{
    int number = 10;
    int *p = &number;
    cout << "The address of variable: " << p << endl;
    cout << "The Value of variable: " << *p << endl;
}
```

```
The address of variable: 0x4d4b5ffcb4
The Value of variable: 10
```

We can also access and update variables in other function by using the pointer and reference variables.
Consider the following example for better understanding.
We want to create a function that takes a number from user and increments it by one.

```cpp
#include <iostream>
using namespace std;
void changeNumber(int number)
{
    number++;
    cout << "In function: " << number << endl;
}
main()
{
    int number = 10;
    changeNumber(number);
    cout << "After funtion: " << number << endl;
}
```

```
In function: 11
After funtion: 10
```

Skill: Learning the use of Pointers and Reference Variables

We can solve this question by using both pointers and reference variables

Using Pointer Variable	Using Reference Variable

`#include <iostream>`
`using namespace std;`
`void changeNumber(int *number)`
`{`
`  (*number)++;`
`  cout << "In function: " << *number << endl;`
`}`
`main()`
`{`
`  int number = 10;`
`  changeNumber(&number);`
`  cout << "After funtion: " << number << endl;`
`}`
	`#include <iostream>`
	`using namespace std;`
	`void changeNumber(int &number)`
	`{`
	`  number++;`
	`  cout << "In function: " << number << endl;`
	`}`
	`main()`
	`{`
	`  int number = 10;`
	`  changeNumber(number);`
	`  cout << "After funtion: " << number << endl;`
	`}`

In function: 11 After funtion: 11	In function: 11 After funtion: 11
This is known as Pass by Reference where we are passing a variable to a function however, we are not passing the value of the variable instead we are passing the address of the variable. Therefore, any changes in the variable using its address will be visible when accessing the variable.	

Task 03(CL): Write a function that swaps the values of two passed variables.

```
void swapNumbers(int &number1, int &number2)
{
    int temp = number1;
    number1 = number2;
    number2 = temp;
}
main()
{
    int number1 = 10;
    int number2 = 20;
    swapNumbers(number1, number2);
    cout << "Number1: " << number1 << endl;
    cout << "Number2: " << number2 << endl;
}
```

```
Number1: 20
Number2: 10
```

Skill: Learning the use of Pointers and Reference Variables

Similarly, we can pass an array to a function as well.

However, the arrays are passed as reference by default in c++.

```
void passingToArray(int arr[], int size)
{
    for(int i=0;i<size; i++)
    {
        cout << arr[i] << " ";
    }
}
main()
{
    int size = 3;
    int numbers[size] = {1,2,3};
    passingToArray(numbers, size);
}
```

```
1 2 3
```

Task 04(CP): Write a function that takes an array from the user and returns the average of all the numbers in the array.

```
float averageOfArray(int arr[], int size)
{
    int sum = 0;
    for (int i = 0; i < size; i++)
    {
        sum = sum + arr[i];
    }
    return sum/size;
}
main()
{
    int size = 3;
    int numbers[size] = {1, 2, 3};
    float avg = averageOfArray(numbers, size);
    cout << avg;
}
```

```
2
```

Skill: Learning the use of Pointers and Reference Variables

Programming Fundamental

Lab Manual - Week 14

Task 05(CP): Write a Function that checks whether the 3X3 matrix is sparse matrix or not, the matrix should be passed to the function.
Note: To check whether a matrix is a sparse matrix, we only need to **check the total number of elements that are equal to zero**. If this count is more than **(m * n)/2**, we return true

Task 06(CP): Write a program that passes two arrays into function and merge those two arrays into a third array.
All these arrays should be passed to the function.

Task 07(CP): Write a program that passes an array into function and left rotate the array by n positions.

1	2	3	4	5	6	7	8	9
				3				
4	5	6	7	8	9	1	2	3

Task 08(CP): Aeroplane Seat Reservation System

Write a program that can be used to assign seats for a commercial aeroplane. The aeroplane has 13 rows, with six seats in each row. Rows 1 and 2 are first class, rows 3 through 7 are business class, and rows 8 through 13 are economy class. Your program must prompt the user to enter the following information:

1. Ticket type (first class, business class, or economy class)
2. Desired seat ((1-13) for Rows and (A to F) for Columns)

Output the seating plan in the following form:

	A	**B**	**C**	**D**	**E**	**F**
Row 1	*	*	X	*	X	X
Row 2	*	X	*	X	*	X
Row 3	*	*	X	X	*	X
Row 4	X	*	X	*	X	X
Row 5	*	X	*	X	*	*
Row 6	*	X	*	*	*	X
Row 7	X	*	*	*	X	X
Row 8	*	X	*	X	X	*
Row 9	X	*	X	X	*	X
Row 10	*	X	*	X	X	X

Skill: Learning the use of Pointers and Reference Variables

Row 11	*	*	X	*	X	*
Row 12	*	*	X	X	*	X
Row 13	*	*	*	*	X	*

Here, * indicates that the seat is available; X indicates that the seat is occupied. Make this a menu-driven program; show the user's choices and allow the user to make the appropriate choices. When the program is about to end, store the currently reserved seats arrangement in the file. Next time the program starts, read the file and start from the previously stored reserved seats.

Make the following functions for the menu of the system

1. Load the reserved seats arrangement from the file
2. Ask the user for the ticket type and desired seat
3. Print the reserved seats arrangement
4. Store the reserved seats arrangement in the file
5. New Plane seats arrangement.

Task 09(CP):

Ma Sha Allah!!! You have become an excellent programmer. Now, a mathematician has hired you to develop a software that can apply different operations on Matrices.

Write different functions to perform various Matrix operations on a given 2-D Arrays.

- **Addition of two matrices (A+B)**
- **Subtraction of two matrices (A-B)**
- **Multiplication of two matrices (A*B)**
  - For multiplying two matrices, The number of columns in the first matrix must be equal to the number of rows in the second matrix.
- **Scalar multiplication (value*Matrix)**
  - For scalar multiplication, The matrix can be of any order. You just have to multiply all elements in the matrix by the scalar.
- **Is Identity matrix (isIdentity(Matrix))**
  - In linear algebra, the identity matrix of size n is the n × n square matrix with ones on the main diagonal and zeros elsewhere.
- **Transpose of the Matrix (transpose(Matrix))**
  - In linear algebra, the transpose of a matrix is actually an operator that flips a matrix over its diagonal by switching the row and column indices of matrix B and producing another matrix.
- **Is Diagonal Matrix (isDiagonal(Matrix))**

Skill: Learning the use of Pointers and Reference Variables

- A diagonal matrix is a square matrix where all of its non-diagonal elements are zero

- **Is Symmetric Matrix (isSymmetric(Matrix))**
  - In linear algebra, a symmetric matrix is a square matrix that is equal to its transpose.

There are 3 arrays available with names A, B and C each of 3x3 size. You can initialize any array any time.
The program runs until the user enters “Exit”.
The format to take input is as follows:
A={1,2,3,4,5,6,7,8,9}
C={1,0,0,0,1,0,0,0,1}
This will initialize a 3x3 array.
You will take input in a string format and then extract the information of the array elements and store them in the array accordingly.
The user will enter C=A+B for matrix addition, C=A-B for matrix subtraction, C=A*B for matrix multiplication and so on and store the result in C array. Results will always be stored in the C array.
You have to take input in a string and then separate out what is the first matrix, what is the operation and what is the second matrix.
print(array) will print the contents of the array on the console.
E.g, print(A), print(B), print(C)
For checking different properties of the matrices, the user will also write the function call on the console i.e., isIdentity(A), isDiagonal(B), isSquare(A).
You have to take that input in the string format and separate out the information accordingly.
Sample Output:

```
>>A={1,0,0,0,1,0,0,0,1}
>>print(A)

1       0       0
0       1       0
0       0       1

>>B={1,1,1,1,1,1,1,1,1}
>>print(B)

1       1       1
1       1       1
1       1       1

>>C=A+B
>>print(C)

2       1       1
1       2       1
1       1       2
```

```
>>C=A*B
>>print(C)

17      17      17
17      17      17
17      17      17

>>isDiagonal(B)

0
>>isDiagonal(A)

1
>>isIdentity(A)

1
>>transpose(A)
>>print(C)

1       0       0
0       1       0
0       0       1
```

```
>>Exit
PS C:\C++\Week11>
```

Good Luck and Best Wishes !!

Happy Coding ahead :)

Skill: Learning the use of Pointers and Reference Variables

Muhammad Irzam & Maida Shahid, Department of Computer Science, UET Lahore