**Programming Fundamentals**

|  |  |
| --- | --- |
| Lab 08 Revision | |
| **Topic** | POINTERS, POINTERS & FUNCTIONS, POINTERS & FUNCTIONS, POINTERS & 1D ARRAYS, POINTERS, 1D ARRAYS & FUNCTIONS |
| **Objectives** | Practical implementation of the following topics   * POINTERS * POINTERS & FUNCTIONS * POINTERS & 1D ARRAYS * POINTERS, 1D ARRAYS & FUNCTIONS * DYNAMIC MEMEORY ALLOCATION |

**Lab Description:**

This lab is basically design to revise all the previous concepts which we have used in our last labs which were about pointers, pointers and its functions, pointers and 1D array, pointers 1D array and Dynamic Memory Allocation.

**POINTERS:**

A pointer however, is a variable that stores the memory address as its value. A pointer variable points to a data type (like int or string) of the same type, and is created with the \* operator. The address of the variable you're working with is assigned to the pointer.

**POINTERS Declaration:**

Create a pointer variable with the name ptr, that points to a string variable, by using the asterisk sign \* (string\* ptr). Note that the type of the pointer has to match the type of the variable you're working with. Use the & operator to store the memory address of the variable called food, and assign it to the pointer. Now, ptr holds the value of declared array in memory address.

There are three ways to declare pointer variables, but the first way is preferred:

string\* mystring; // Preferred  
string \*mystring;  
string \* mystring;

**1D-Array:**

An array is a collection of a fixed number of components (also called elements) all of the same data type and in contiguous (that is, adjacent) memory space. An array whose components are of type char.

**Declaration statement of an array:**

char list[10];

Above statement is used for creating an array. But as we discussed above when a memory location is reserved for this array it holds some values which are not assigned by user. So those values are considered as garbage value. In order to avoid garbage value, it is a good practice to assign values at the time of creation of array.

**Initialization statement of an array:**

Assigning value at the time of declaring of array or variable is called initialization statement. There are multiple ways of initialize an array.

**Initialize an array with null character:**

char list[5]={}; OR char arr[5]{};

**Full array initialization with different elements:**

char list[5]={‘a’,’b’,’c’,’d’,’e’};

**Partial array initialization with different elements:**

char list[5]={‘a’,’b’};

on remaining indexes null will be assigned as initial value in case of partial array initialization.

**What is Memory Allocation?**

As we discussed earlier when we need a memory for any purpose we need to allocate it before using it. There are two ways via which memories can be allocated. The two ways are:

**Compile time allocation or static allocation of memory:**

Where the memory for named variables is allocated by the compiler. Exact size and storage must be known at compile time and for array declaration, the size has to be constant.

**How is it different from memory allocated to normal variables?**

For normal variables like “int a”, “char str[10]”, etc, memory is automatically allocated and deallocated. For dynamically allocated memory like “int \*p = new int[10]”, it is the programmer’s responsibility to deallocate memory when no longer needed. If the programmer doesn’t deallocate memory, it causes a [memory leak](https://www.geeksforgeeks.org/what-is-memory-leak-how-can-we-avoid/) (memory is not deallocated until the program terminates).

**Runtime allocation or dynamic allocation of memory:**

Where the memory is allocated at runtime and the allocation of memory space is done dynamically within the program run, the exact space or number of the item does not have to be known by the compiler in advance. Pointers play a major role in this case.

The key word **new** is used to allocate dynamic or run time memory.

**Example:**

int \* p=nullptr, size=5;  //declares a pointer p

p = new int;   //dynamically allocate an int and loading the base address in pointer p.

p = new int[size]; //dynamically allocate an int array of size 5 and loading the base address in pointer p.

**Delete Operator:**

Memory de-allocation is also a part of this concept where the “clean-up” of space is done for variables or other data storage. It is the job of the programmer to de-allocate dynamically created space. For de-allocating dynamic memory, we use the delete operator. In other words, dynamic memory Allocation refers to performing memory management for dynamic memory allocation manually.

**Example:**

int \* p=nullptr, size=5;  //declares a pointer p

p = new int;   //dynamically allocate an int and loading the base address in pointer p.

delete p; // de allocate or free the memory.

p = new int[size]; //dynamically allocate an int array of size 5 and loading the base address in pointer p.

delete [] p; // de allocate or free the memory.

**Lab Tasks**

**Task 1**

Write a C++ program in which you are required to take N from user which is size of the data. Allocate memory for N integers at runtime. Calculate its average and deallocate the unnecessary memory. At the end your program should display the average, size of dynamic memory which has been deleted now and size of memory which is in use now.

**Task 2**

Write a C++ program to demonstrate dynamic allocation of two variables p and r in memory and then its deallocation from the previous slot of memory by using the keywords new and delete where the value of two different variables should be

Value of p= 25

Value of r=25.75

**Task 3**

Write a C++ program to find sum, average of three numbers by taking input from file into the variables and allocate the memory dynamically to those variables and send the two pointers to the function where sum will be calculated. You can use required functions.

**Task 4**

Write a C++ program to define a pointer variable. Then assign the address of a variable ‘num’ to the pointer and then you will have two ways to access the value of num; one way is direct using num variable and other one is by using pointer. Use the pointer variable to manipulate the value of num and then store the answers in file of name ‘results.txt’. User can add, multiply, divide and subtract any number from ‘num’.

**Task 5**

Create three parallel arrays dynamically of same size in order to store ids, working\_years and salaries of employees working in an XYZ organization. Take employee’s data from user and store it in arrays. Now your task is to remove those employees which are present on indices that are multiple of 3. Display the remaining data of employees at the end of your program.

**Task 6**

An organization want to filter all their employees on seniority basis how will they use the pointers to direct the rank of each employee through their ID. Use arrays which you have created in Task 5.

**Task 7**

Write a C++ program which will take address of two person and then compare them whether they are living at same place or different.

**Task 8**

Suppose University want to keep record of students who donate some money for flood effected people. They asked you to write program in C++ to solve this problem. You program should ask the user to enter donor detail, see donors details, see overall donation, etc. Data should be permanently stored on hard disk.

**Task 9**

Use description from above mentioned question. Now University wants to give 100% scholarship to that student who donated maximum amount, 75% scholarship to second maximum doner and 50% to 3rd maximum doner. Write a C++ code to help the University management to figure out top three students who donated maximum amount.