**Object Oriented Programming Lab 02**

|  |  |
| --- | --- |
| **Course**: Object Oriented Programming (CL1004) | **Semester**: Spring 2025 |
| **Instructor**: Muhammad Monis |  |
| Note:   * Maintain discipline during the lab. * Listen and follow the instructions as they are given. * Just raise hand if you have any problem. * Completing all tasks of each lab is compulsory. * Get your lab checked at the end of the session. |  |

**Pointers in C++**

A Pointer is a variable whose content is a memory address. In C++, a pointer is a variable that stores the memory address of another variable. Pointers are one of the fundamental concepts in C++ that allow for dynamic memory allocation, passing variables by reference.

**Single Pointer**

A single pointer refers to a pointer variable that holds the address of a single variable of a specific type. It points to a single memory location. To declare a single pointer variable, you need to specify the data type, an asterisk symbol (\*) and the name of the pointer variable.

**dataType \*ptrName;**

Following is an example of declaration of a Pointer variable:

**int \*ptr;**

Pointer variable holds the memory address of the variable which is of same data type (integer in this case). To assign the memory address of any variable to the pointer variable we use Address of Operator (&):

**int intVar = 5;**

**int \*ptr = &intVar;**

In this statement ptr now holds the memory address of an integer variable **‘intVar’**. To access the value at the memory address (currently stored) in the variable we use Dereferencing Operator (\*). Do not confuse this with the symbol used for the declaration of a pointer.

**int intVar2 = \*ptr;**

In this statement another integer variable ‘intVar2’ is now initialized with the value at the memory address which is stored in ptr (that is the value of intVar).

**2D Pointer**

In C++, a 2D array can be represented using pointers. A 2D pointer is essentially a pointer to an array of pointers, where each pointer points to a separate array representing a row in the 2D array.

Following is an example of declaration of a 2D pointer:

**int\*\* matrix;**

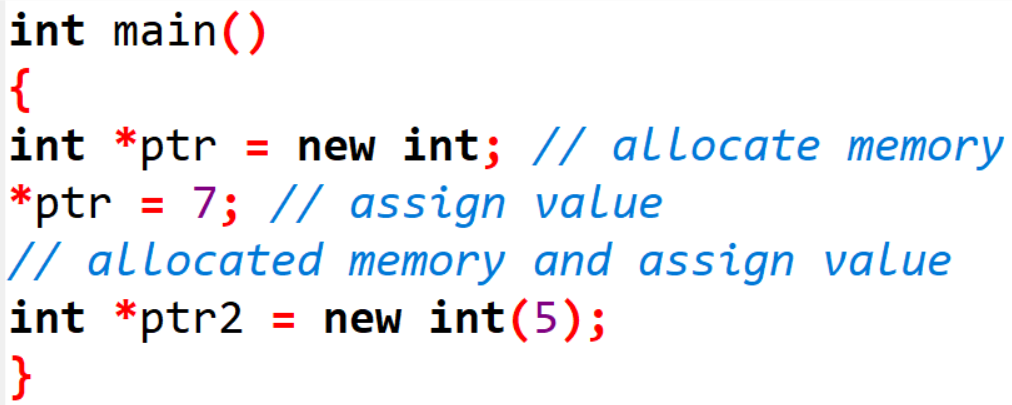
**Dynamic Memory Allocation**

Variables created during the program execution are called dynamic variables. To create a dynamic variable, we use new operator. C++ supports three types of memory allocation:

1. Static memory allocation happens for static and global variables. Memory for these types of variables is allocated once when your program is run and persists throughout the life of your program.
2. Automatic memory allocation happens for function parameters and local variables. Memory for these types of variables is allocated when the relevant block is entered, and freed when the block is exited, as many times as necessary.
3. Dynamic memory allocation is a way for running programs to request memory from the operating system when needed.

**new Operator**

This operator is used to allocate a memory of a particular type. This creates an object using the memory and returns a pointer containing the memory address. The return value is mostly stored in a pointer variable.

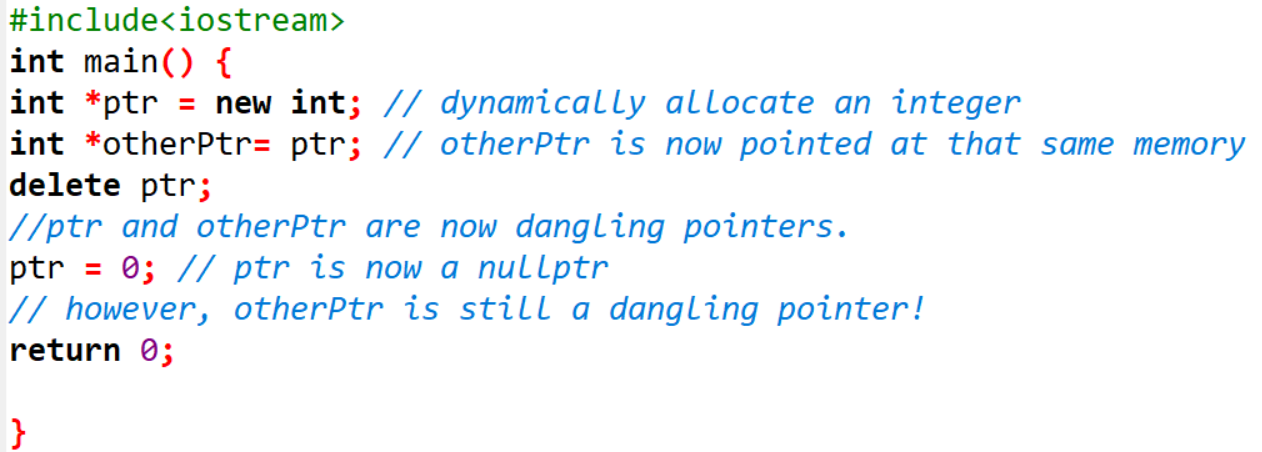


Consider another example below:

**new dataType [ size]; // to allocate an array of variables.**

**delete Operator**

When we allocate memory dynamically, we need to explicitly tell C++ to deallocate this memory. delete Operator is used to release / deallocate the memory. Consider the below example:

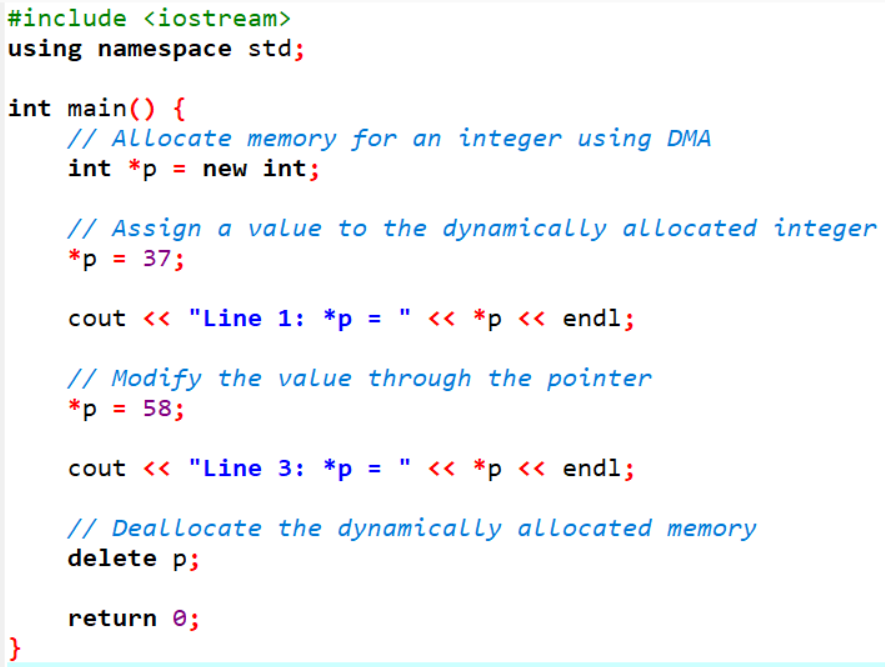


To delete the dynamically allocated memory we use delete operator. delete operator is used to free the memory which is dynamically allocated using new operator.

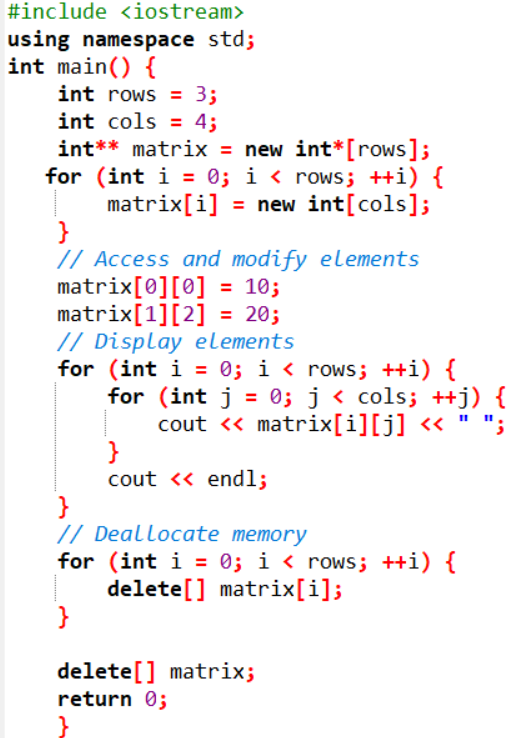
**delete ptrVar; //to deallocate single dynamic variable**

**delete [] ptrArray; //to deallocate dynamically created array.**

**Example: Code for Single Pointer using Dynamic Memory Allocation**



**Example: Code for 2D Pointer using Dynamic Memory Allocation**



**Struct in C++**

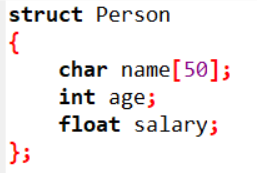
Structure is a collection of variables of different data types under a single name. It is similar to a class in that, both holds a collection of data of different data types. For example: You want to store some information about a person: his/her name, citizenship number and salary. You can easily create different variables name, citNo, salary to store this information separately.

However, in the future, you would want to store information about multiple people. Now, you'd need to create different variables for each information per person: name1, citNo1, salary1, name2, citNo2, salary2.

You can easily visualize how big and messy the code would look. Also, since no relation between the variables (information) would exist, it's going to be a daunting task. A better approach will be to have a collection of all related information under a single name Person, and use it for every person. Now, the code looks much cleaner, readable and efficient as well. This collection of all related information under a single name Person is a structure.

**Declaring a struct in C++**

The struct keyword defines a structure type followed by an identifier (name of the structure). Then inside the curly braces, you can declare one or more members (declare variables inside curly braces) of that structure. For example:

****

Here a structure person is defined which has three members: name, age and salary.

When a structure is created, no memory is allocated. The structure definition is only the blueprint for the creating of variables. You can imagine it as a datatype like when you define an integer. The int specifies that a variable can hold integer element only. Similarly, structure definition only specifies that, what property a structure variable holds when it is defined.

**Note: Remember to end the declaration with a semicolon (;).**

**Defining a struct variable in C++**

Once you declare a structure person as above. You can define a structure variable as:

**Person Ali;**

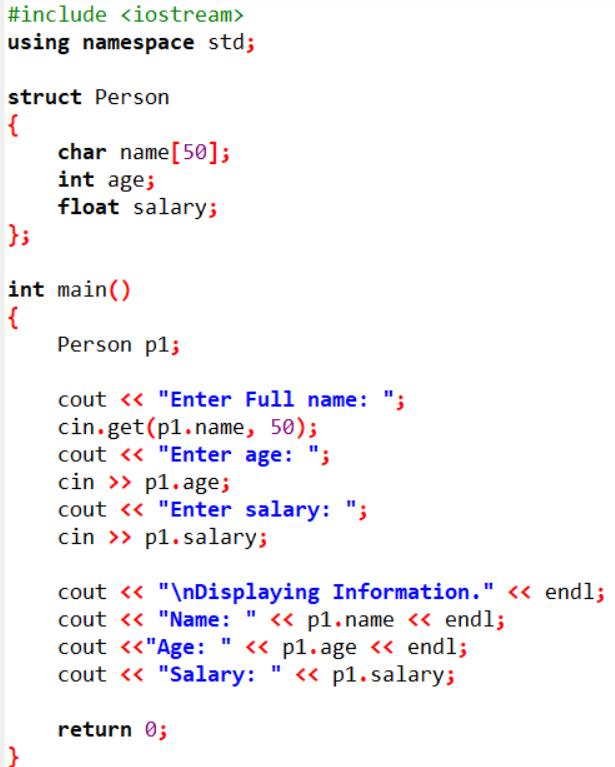
Here, a structure variable bill is defined which is of type structure Person. When structure variable is defined, only then the required memory is allocated by the compiler. Considering you have either 32-bit or 64-bit system, the memory of float is 4 bytes, memory of int is 4 bytes and memory of char is 1 byte. Hence, 58 bytes of memory is allocated for structure variable Ali.

**Accessing members of struct in C++**

The members of structure variables are accessed using a dot (.) operator. Suppose, you want to access age of structure variable Ali and assign it 50 to it. You can perform this task by using following code below:

**Ali.age = 50;**

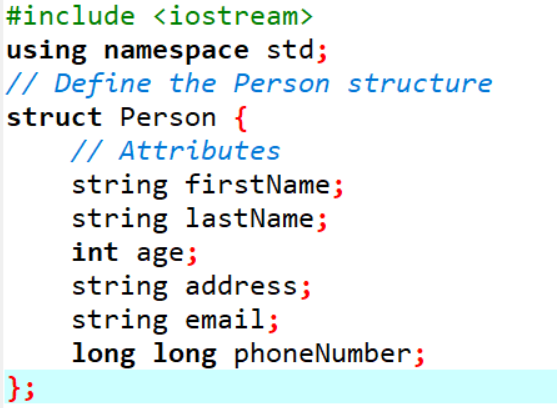
**Example: Code for struct in C++**



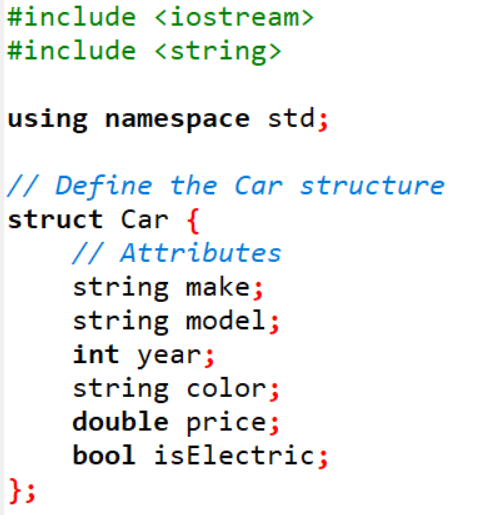
Here a structure Person is declared which has three members name, age and salary. Inside main() function, a structure variable p1 is defined. Then, the user is asked to enter information and data entered by user is displayed.

**Real World Use Case of struct in C++**

Consider a real-world scenario of a "Person" structure in C++ with attributes that might be used in a personal information management system.



Consider a different example with a "Car" structure in C++. This structure represents real-world attributes of a car.



**Tasks:**Task 1: Write a program that calculates the sum of all the elements in array using pointers

(Note: Generate the array via user in the main function use argv)

Task 2: You are required to pass a single dimension array to a function (type of the array can be your choice) and perform add operations to each element. Return the array to the main function after completion. One small catch is that the arguments and the return type should be void\*. (Create an array via the main arguments)

Task 3:

Write a program that can resize a single dimension int array if the elements exceed the total size. Consider an int array of size 5 (uninitialized), you start adding items to the array and if the quantity of the elements exceeds double the array size. Once finished resizing the array again to the max quantity of the elements present in the array.

Task 4:

Consider there are two structures Employee (depended structure) and another structure called Organisation(Outer structure). The structure Organization has the data members like organisation\_name,organisation\_number. The Employee structure is nested inside the structure Organization and it has the data members like

employee\_id, name, salary.

org.emp.employee\_id;

org.emp.name;

org.emp.salary;

org.organisation\_name;

org.organisation\_number;

Here, org is the structure variable of the outer structure Organisation and emp is the structure variable of the inner structure Employee.

Output the following data using above structure

The size of structure organisation : 123

Organisation Name : NU-Fast

Organisation Number : NUFAST123ABC

Employee id : 127

Employee name : Linus Sebastian

Employee Salary : 400000

Task 5:

Write a program that creates N structures. Inside each struct are two entities, id and name. You are required to first sort this structure first via ID then via the first character of the name in ascending order. Print all the struct data in both formations.

Task 6:

Write a program that creates a dynamic array of not values but structures. Each struct would contain another struct and in that struct a variable of subject should exist. The outer structure should be an ID and an array (Fixed) for subjects. (Note you can utilize vectors for this question)