Experiment No. 04

Structure

**OBJECTIVE**:

Things that will be covered in today’s lab:

* struct
* Pointer to struct
* nested struct

**THEORY:**

**Structs:**

We have already discussed array which is a structured data type having all elements of the same type. Another structured data type is a *struct* (or *record*) which allows you to group related values that are of different types.

A *struct* is a collection of fixed number of components, members, in which the members are accessed by name. The members may be different data types.

**Defining a Structure:**

A *struct* is a user defined data type that requires the keyword *struct* followed by the name chosen for the *struct*. The data members that define *struct* are contained between a set of curly brackets. *Struct* definition must be ended with a semicolon.

Struct struct\_name

{

//

// Member variables

//

};

**Accessing Struct Members:**

To access any member of a structure, we use the *member access operator (.).* The member access operator is coded as a period between the structure variable name and the structure member that we wish to access. You would use the *struct* keyword to define variables of structure type.

You can return struct and pass it as parameter to functions as well. You can also declare a pointer to struct datatype the way you would declare it for primitive datatypes.

**Pointer to structure:**You can define pointers to structures in a very similar way as you define a pointer to any other variable. Now, you can store the address of a structure variable in the above defined pointer variable. To find the address of a structure variable, place “&” operator before the name of the structure.

Here variable is an object of structure type *struct\_name* and *p\_var* is a pointer to point to objects of structure type variable. Therefore, the following code would also be valid:

struct\_name \* p\_var;

struct\_name variable;

p\_var = & variable;

p\_var->struct\_members;

The value of the pointer *p\_var*would be assigned the address of the object variable. The arrow operator (->) is a **dereference** operator that is used exclusively with pointers to objects that have members. This operator serves to access the member of an object directly from its address as can be seen from Table 5.1.

|  |  |  |
| --- | --- | --- |
| **Expression** | **What is evaluated** | **Equivalent** |
| a.b | Member b of object a |  |
| a->b | Member b of object pointed to by a | (\*a).b |

**Table 5.1**

**Example:** What should be the output of this program?

|  |
| --- |
| struct movies\_t  {  char \* title ;  int year;  };  int main ()  {  movies\_t amovie;  movies\_t \* pmovie;  pmovie = &amovie;  pmovie->title= "MATRIX";  pmovie->year = 1999;  cout<<pmovie->title;  cout<<" ("<<pmovie->year <<")\n";  } |

**Example:** Please study and write the output of this program.

|  |
| --- |
| #include<iostream>  #include<string>  using namespace std;  struct uni {  string title;  int year;  };  uni ReturnUni()  {  uni runi;  cout << "Enter university name and then year" << endl;  cin >> runi.title >> runi.year;  return runi;  }  void PrintUni(uni p)  {  cout << p.title << " ";  cout << p.year << endl;  }  int main() {  uni funi, puni;  int num = 0;  puni.title = "FAST University";  funi.year = 1992;  puni = ReturnUni();  PrintUni(funi);  PrintUni(puni);  uni\* uniPtr = &funi;  cout << uniPtr->title << endl;  cout << (\*uniPtr).year << endl;  cout << "Enter number of universities" << endl;  cin >> num;  uniPtr = new uni[num];  for (int i = 0; i < num; i++)  uniPtr[i] = ReturnUni();  for (int i = 0; i < num; i++)  PrintUni(uniPtr[i]);  delete[]uniPtr;  return 0;  } |

**For all of the exercises below save your code on the learning management system (LMS) and give the screen shot of the output you get on the console in the space provided after every exercise.**

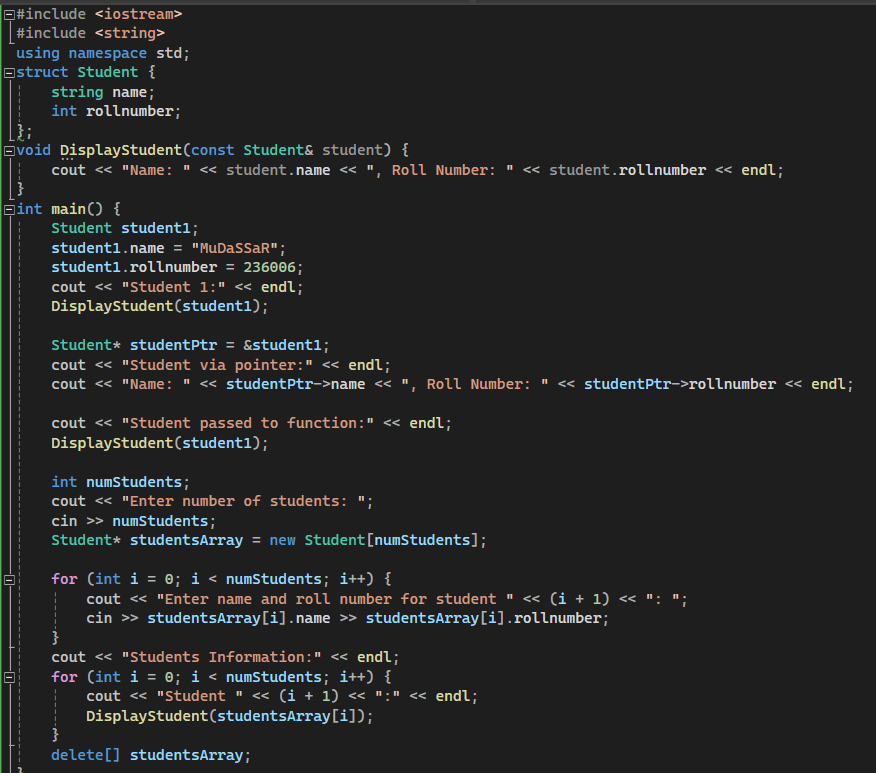
**Exercise 1:** (**10 points)**

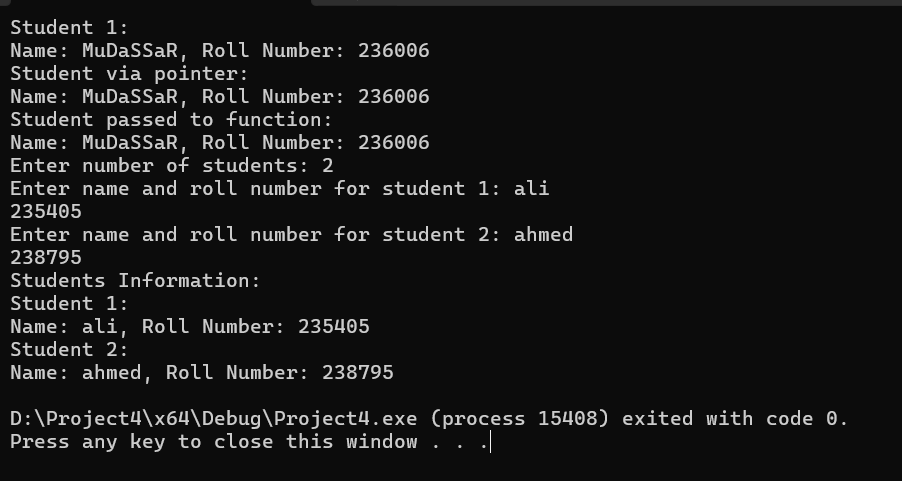
Using the above example as guideline please define a struct named **Student**, with two data members: *name* and *rollnumber* .

Write a program that demonstrates the following:

1. Create a Student variable and access its data members. Please use your own name and rollnumber.
2. Create a Student pointer that carries the address of the variable declared in step a and access its data members via the pointer.
3. Write a function that takes as parameter the Student variable declared in step a and displays its data members
4. Create a dynamic array of Students after asking the user about the number of Students they would like to enter and then uses the function written above in a loop to get data for the students. Display the data in another loop.

Please give the screenshot of the output that you get after running your program:

****

****

**Nested Struct:**

A nested structure in C++ is nothing but a structure within a structure. One structure can be declared inside the other structure as we declare structure members inside a structure. A structure variable can be a normal structure variable or a pointer variable to access the data.

**Declare two separate struct Declare a nested struct:**

|  |  |
| --- | --- |
| struct xyz{  // xyz member variables  };  Struct abc  {  xyz var;  // abc member variables  }a;  // Access member variables  a.abc\_member  a.var.xyz\_member | struct abc{  // abc member variables  struct xyz  {  // xyz member variables  }var;  }a;  // Access member variables  a.abc\_member  a.var.xyz\_member |

**Access nested members:**

* You can access the members of “*abc”*struct using dot operator.
* A*“xyz”* structure is nested within the “*abc”*struct so the members of *“xyz”* can be accessed using “*abc”* struct object.

**For all of the exercises below save your code on the learning management system (LMS) and give the screen shot of the output you get on the console in the space provided after every exercise.**

**Exercise 2:** (**10 points)**

1. **Please declare a struct called Date which should carry 3 integer data members called day, month and year.**
2. **Now for the Student struct you declared in exercise1, add date\_of\_birth data member whose datatype would be Date.**
3. **Please demonstrate setting and then displaying values for the date of birth of a Student variable. You may reuse the code you wrote for exercise1.**

