Lab objective

The goal of this lab is to understand the use of structures in C++. An important use of structures is to create new “aggregate” data types which can later be used in our programs.

**Step 1)** To begin with, let’s create the following C++ program. Then compile and run it. Everything should be compiling and running at this time.

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

void main()
{
    system("pause");
}
```

**Step 2:** Now let’s add the following Employee structure above the main function in your program. Then look at the comments in the function, and below each comment line write the corresponding C++ code:

```cpp
// Creating user defined Data Types
//
struct Employee
{
    string Name;
    double HourlyWage;
    double HoursWorked;
    double GrossPay;
}; // the semicolon is necessary
```

Now compile the program again to make sure it compiles and runs. Make sure to put the “;” at the end of the structure definition.
Step 3: Now let’s add the following function to the bottom of your program. Note that in this function, we will create a new variable called employee1 which is of type Employee. Also, we use the “dot” notation to access the members of the structure.

Now, make sure to add a forward declaration for the function and then call the function from your main program.

```c++
void CreateAnEmployee(void)
{
    cout << "An Employee--------------------------------" << endl;
    Employee employee1;

    employee1.Name = "John Smith";
    employee1.HourlyWage = 8.75;
    employee1.HoursWorked = 38;

    cout << "Employee Name: " << employee1.Name << endl;
    cout << "Hourly Wage : " << employee1.HourlyWage << endl;
    cout << "Hourly Worked: " << employee1.HoursWorked << endl;
}
```

Take screen shot and put it in your Lab 4 submission. The screen should look something like figure below:
Step 4: Now let’s try something new, let’s create a structure and pass it to a function so that the function could populate it member variables. (Note we would have to pass the variable by Reference, since we want the function to change the content of the variable.)

Add the following code to the bottom of your program.

```cpp
void ReadEmployeeInfo(Employee& AnEmployee)
{
    cout << "Enter Employee Name: ";
    cin >> AnEmployee.Name;

    cout << "Enter Hourly Wage: ";
    cin >> AnEmployee.HourlyWage;

    cout << "Enter Hours Worked: ";
    cin >> AnEmployee.HoursWorked;
}

Now, make sure to add a forward declaration for the function and then call the function from your main program using the lines below:

    // Passing and employee structure to a function
    Employee employee2;
    ReadEmployeeInfo(employee2);
```

Take screen shot and put it in your Lab 4 submission.

To be sure that the above code is functioning we need to print the content of the structure (back in the main program) when we return from the function call. Let’s create a function for printing the content of the structure and test our program. (Add the code below to the bottom of your program, then call it from main.) (Note that we don’t have to send the structure variable by reference this time!)

```cpp
void PrintEmployee(Employee AnEmployee)
{
    cout << "Employee Name: " << AnEmployee.Name << endl;
    cout << "Hourly Wage  : " << AnEmployee.HourlyWage << endl;
    cout << "Hourly Worked: " << AnEmployee.HoursWorked << endl;
    cout << "----------------------------" << endl;
}
```

At this point, your main() function should look like:

```cpp
void main()
{
    CreateAnEmployee();
    // Passing and employee structure to a function
    Employee employee2;
    ReadEmployeeInfo(employee2);
    PrintEmployee(employee2);
    system("pause");
}
```
Step 5: Now let’s create a function that does some calculation. Add the following code to the bottom of your program. (Note why we had to send the variable “AnEmployee” by reference)

```cpp
void CalculateGrossPay(Employee& AnEmployee)
{
}
```

Call this function from main(), then call the print statement again to make sure the gross pay is properly calculated.

Take screen shot and put it in your Lab 4 submission. It should look something like the figure below:

![Screen shot of program output](image)

Step 6: Now let’s try creating an employee inside a function, and then return it back to the main program. Add the following function to the bottom of your program.

```cpp
Employee ReturnAnEmployee(void)
{
    cout << "Returning an employee from a function" << endl;
    Employee employee1;
    employee1.Name = "New Employee";
    employee1.HourlyWage = 18.75;
    employee1.HoursWorked = 25;
    return(employee1);
}
```
Now, go back to your main function and add the following lines. Note that in our main() function, we have to have a structure that receives the what is being returned from the function call. Also, note that the “assignment” operator copies the returned value into employee4 structure. Finally, in order to make sure everything is working, we must display the content of employee4.

```c++
Employee employee4;
employee4 = ReturnAnEmployee();
PrintEmployee(employee4);
```

**Take screen shot and put it in your Lab 4 submission.**

**Step 7: Now, the last thing we want to try is to create an array of employees. Copy the code below to your program and call it from main()**

```c++
void CreateArrayOfEmployees(void)
{
    cout << "Array of Employees-----------------------------" << endl;
    Employee WorkGroup[5];

    WorkGroup[2].Name = "John Smith";
    WorkGroup[2].HourlyWage = 8.75;
    WorkGroup[2].HoursWorked = 38;

    for (int i = 0; i < 5; i++) {
        cout << "Employee Name: " << WorkGroup[i].Name << endl;
        cout << "Hourly Wage : " << WorkGroup[i].HourlyWage << endl;
        cout << "Hourly Worked: " << WorkGroup[i].HoursWorked << endl;
        cout << "Gross Pay    : " << WorkGroup[i].GrossPay << endl;
        cout << "---------------------" << endl;
    }
}
```

The output should look something like the figure below. **Note that only one of the array elements is initialized!**

Take screen shot and put it in your Lab 4 submission.