Data Structures and Object Oriented Programming

Lecture 2

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

User-defined data types

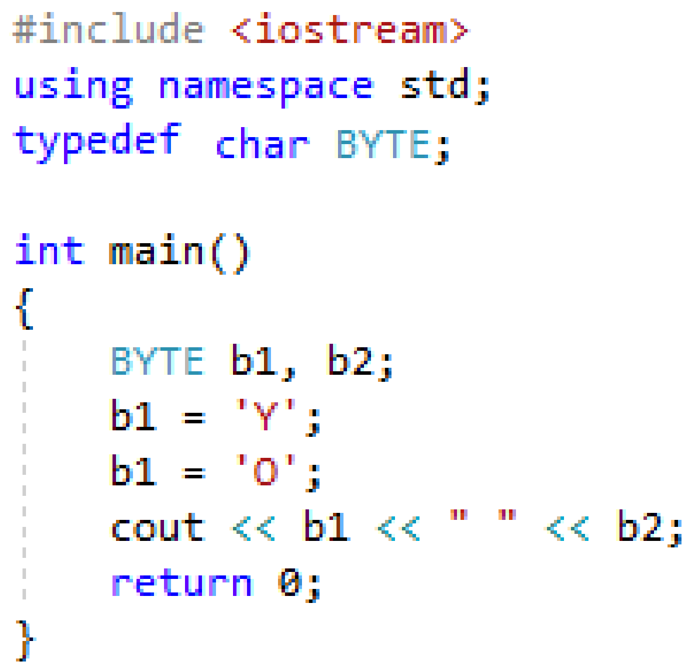
User defined data types

- The data types that are defined by the user are called the ***derived datatype (****or* ***user-defined derived data type / user-defined data type)***

- These types include:
  - ❑ Typedef
  - ❑ Structure
  - ❑ Union
  - ❑ Class

Typedef

- Allows you to define explicitly new data type names by using the keyword typedef
- Does not actually create a new data class, rather it defines a name for an existing type



```
#include <iostream>
using namespace std;
typedef char BYTE;

int main()
{
    BYTE b1, b2;
    b1 = 'Y';
    b1 = 'O';
    cout << b1 << " " << b2;
    return 0;
}
```

Output: Y O

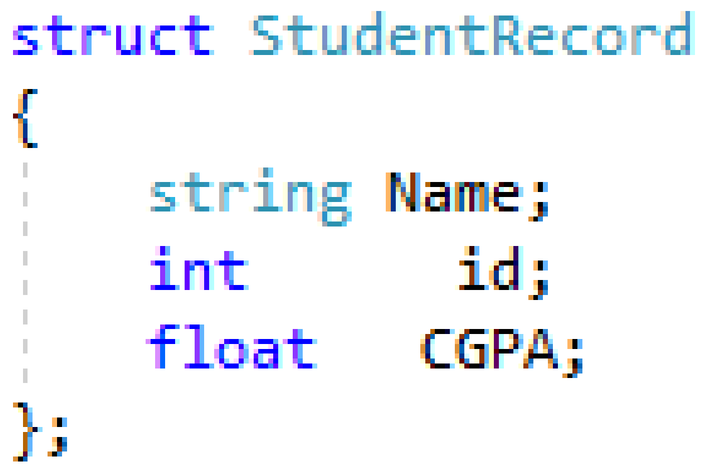
Structures

- A ***struct*** (structure) is a collection of information of different data types (heterogeneous). The fields of a struct are referred to as ***members***.

- Defining a Structure:

```
struct StructName
{
    dataType memberName;
    …
    …
};
```

Example:

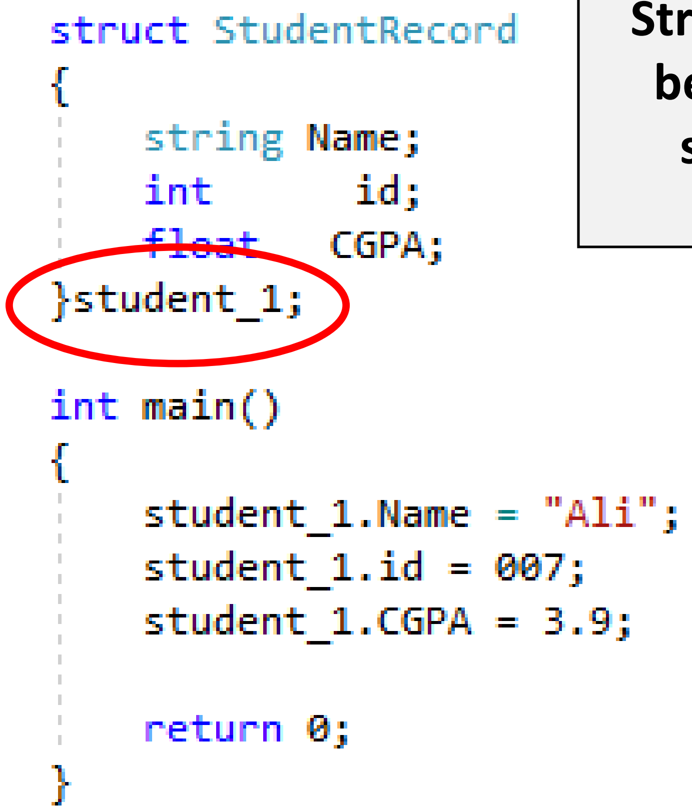


```
struct StudentRecord
{
    string Name;
    int      id;
    float   CGPA;
};
```

Structures

- Two ways to create instance of Structure and accessing the Data Members

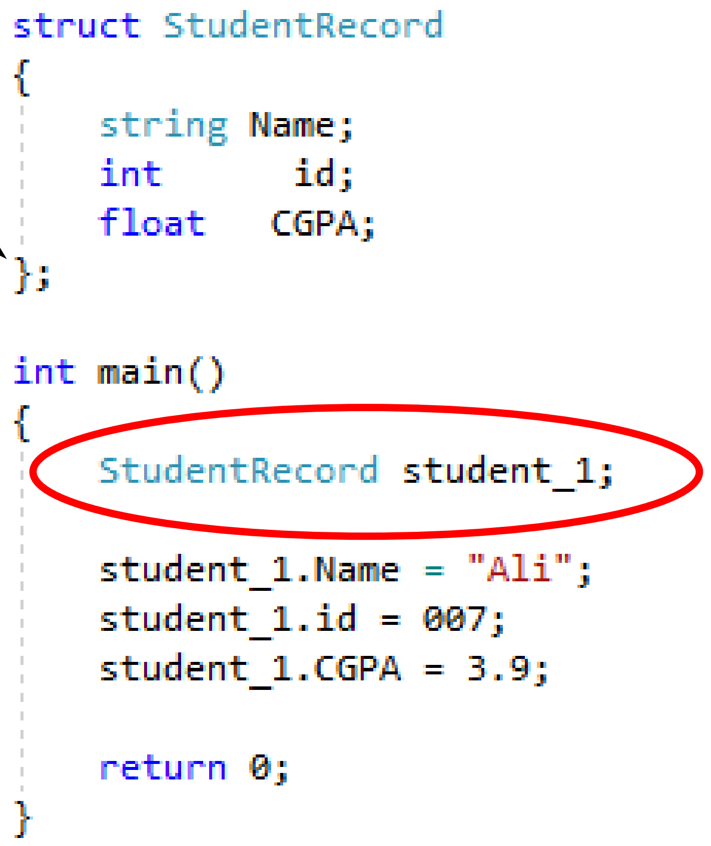
Option 1



```
struct StudentRecord
{
    string Name;
    int       id;
    float    CGPA;
}student_1;

int main()
{
    student_1.Name = "Ali";
    student_1.id = 007;
    student_1.CGPA = 3.9;

    return 0;
}
```

Structure definition must be followed either by a semicolon or a list of declarations

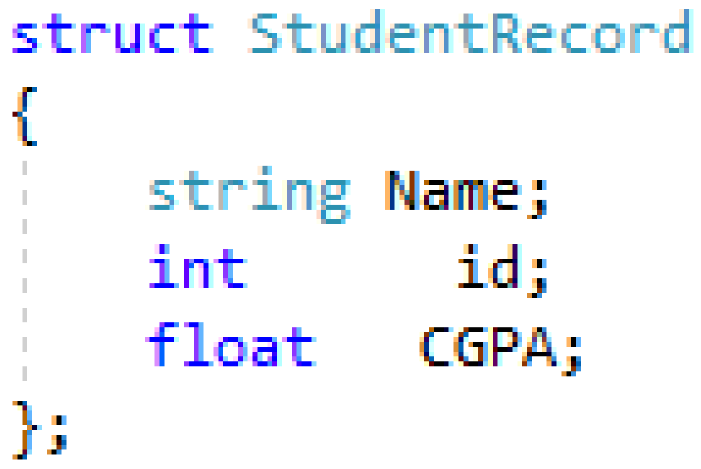
Option 2



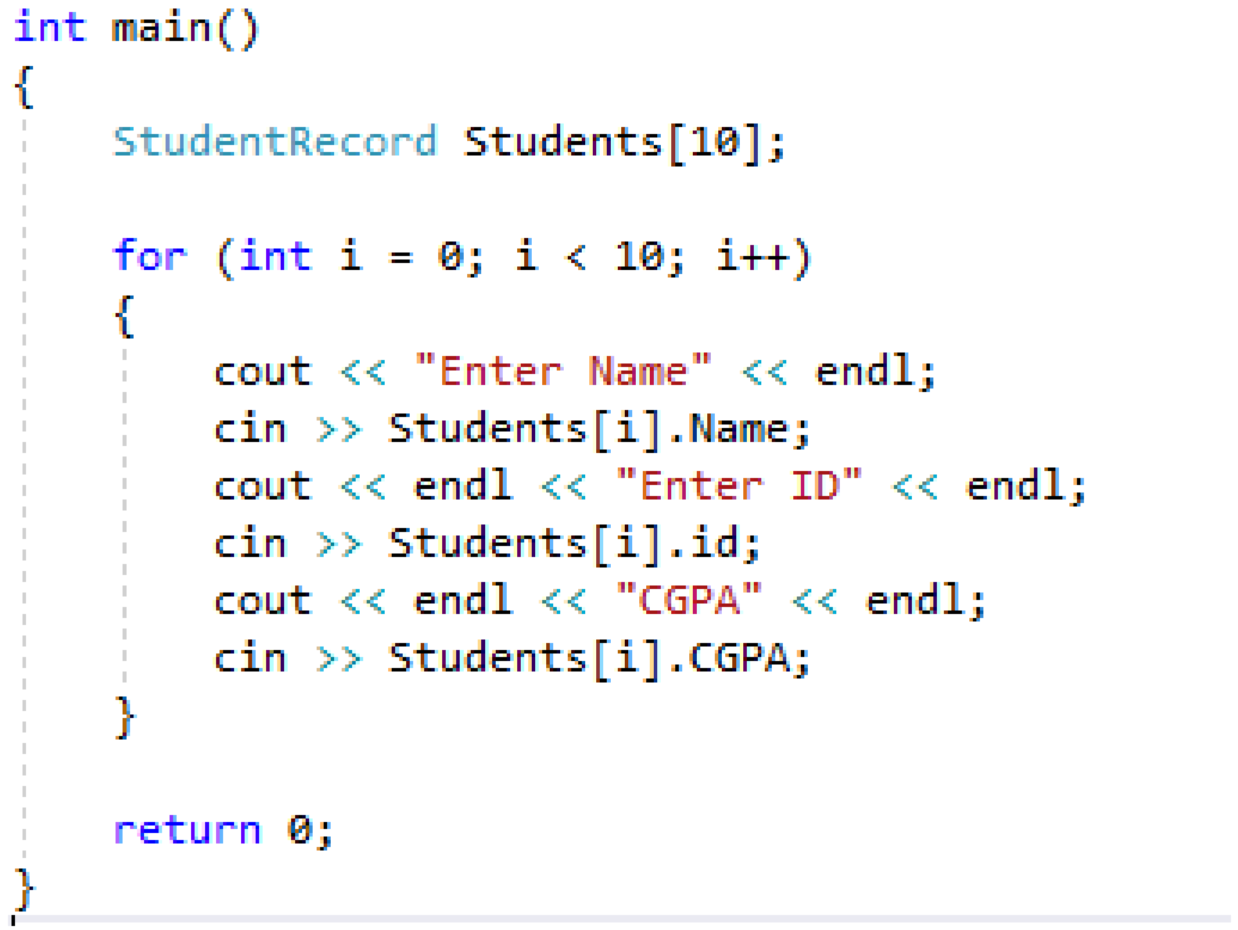
```
struct StudentRecord
{
    string Name;
    int       id;
    float    CGPA;
};

int main()
{
    StudentRecord student_1;

    student_1.Name = "Ali";
    student_1.id = 007;
    student_1.CGPA = 3.9;

    return 0;
}
```

Structures (Recap)

- Exercise: Create an array of “*StudentRecord*” structure and insert data of 10 students in it.



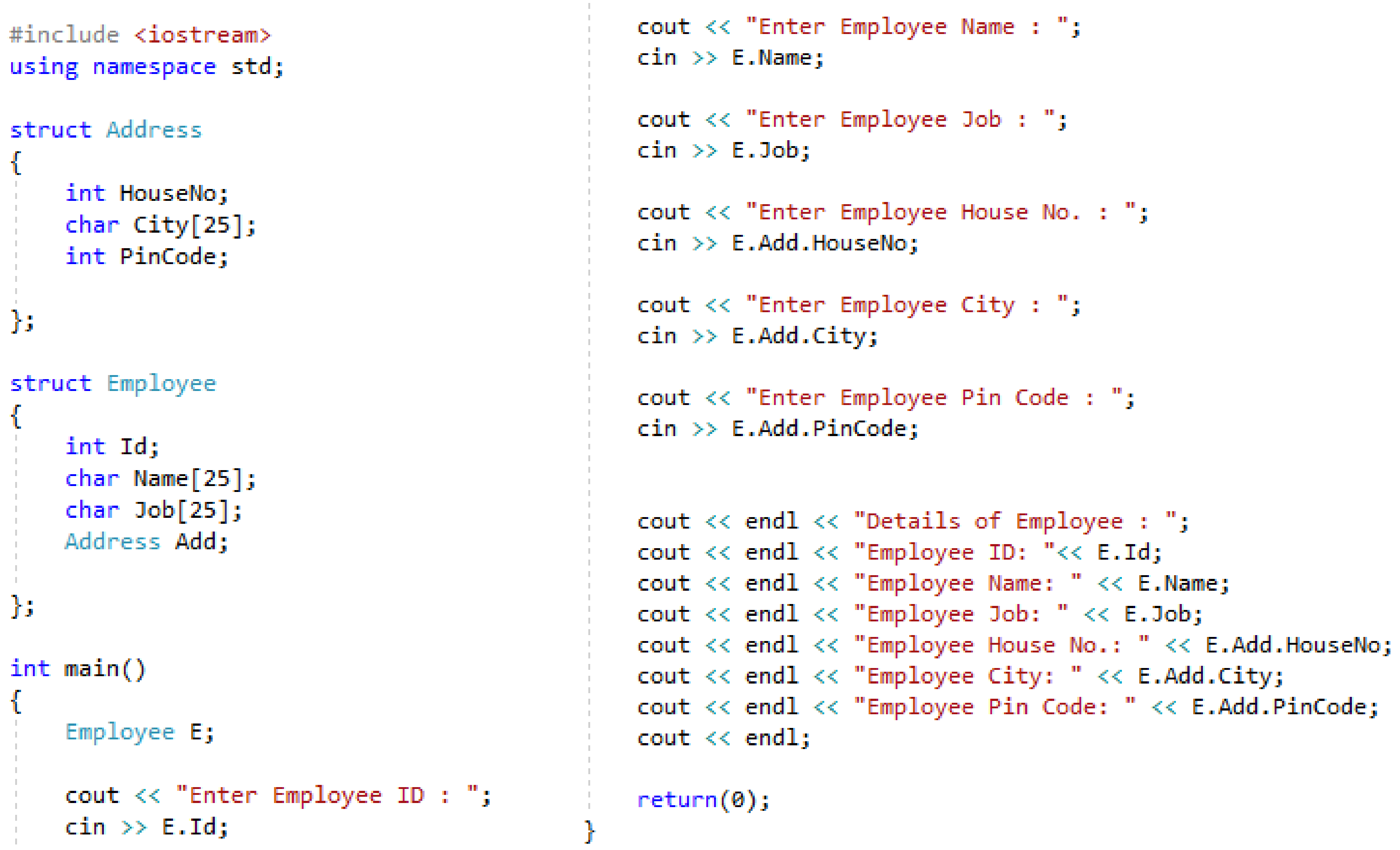
```cpp
struct StudentRecord
{
    string Name;
    int       id;
    float    CGPA;
};
```



```cpp
int main()
{
    StudentRecord Students[10];

    for (int i = 0; i < 10; i++)
    {
        cout << "Enter Name" << endl;
        cin >> Students[i].Name;
        cout << endl << "Enter ID" << endl;
        cin >> Students[i].id;
        cout << endl << "CGPA" << endl;
        cin >> Students[i].CGPA;
    }

    return 0;
}
```

Nested Structures (Recap)

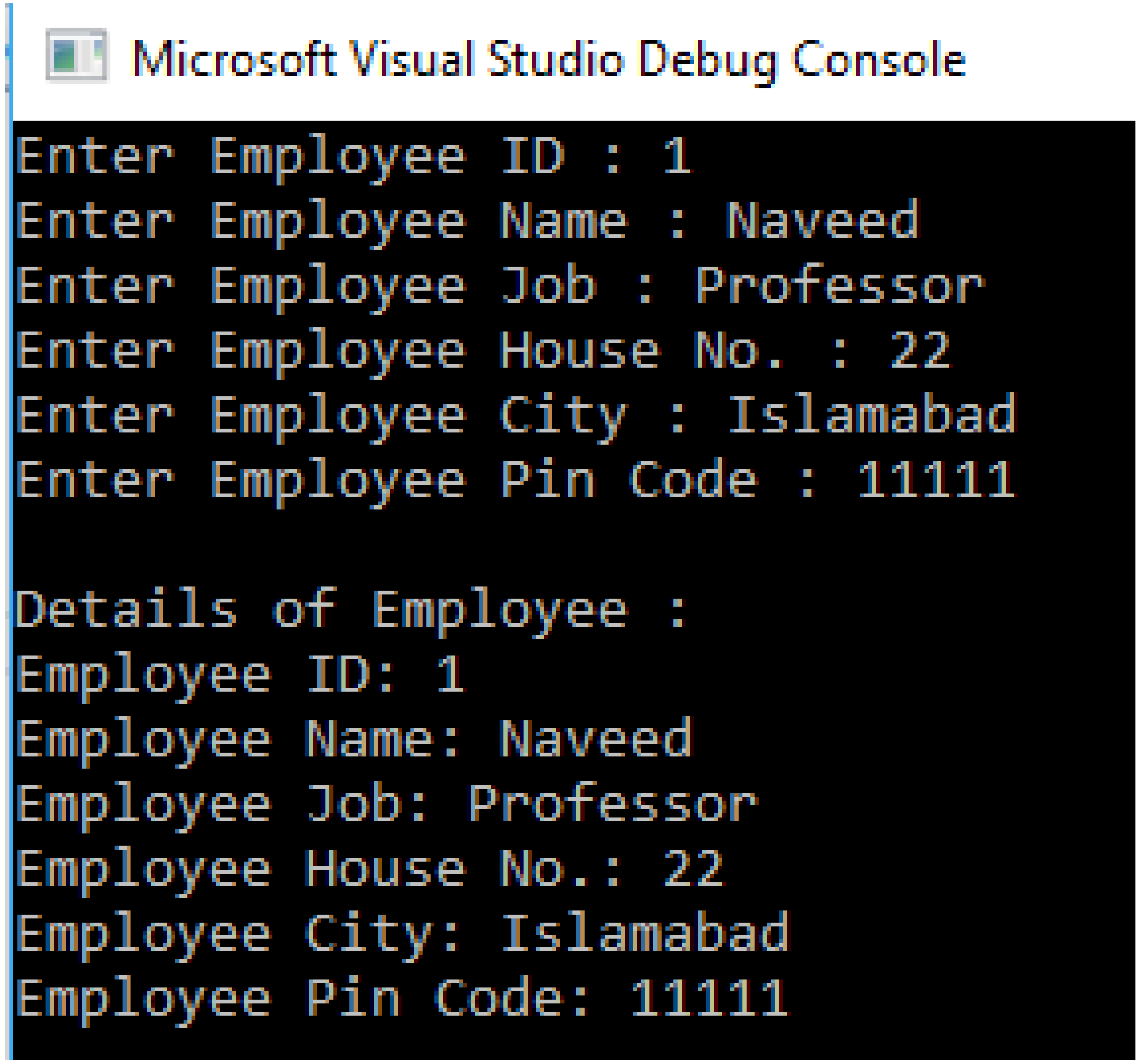
- Example



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

struct Address
{
    int HouseNo;
    char City[25];
    int PinCode;

};

struct Employee
{
    int Id;
    char Name[25];
    char Job[25];
    Address Add;

};

int main()
{
    Employee E;

    cout << "Enter Employee ID : ";
    cin >> E.Id;

    cout << "Enter Employee Name : ";
    cin >> E.Name;

    cout << "Enter Employee Job : ";
    cin >> E.Job;

    cout << "Enter Employee House No. : ";
    cin >> E.Add.HouseNo;

    cout << "Enter Employee City : ";
    cin >> E.Add.City;

    cout << "Enter Employee Pin Code : ";
    cin >> E.Add.PinCode;


    cout << endl << "Details of Employee : ";
    cout << endl << "Employee ID: "<< E.Id;
    cout << endl << "Employee Name: " << E.Name;
    cout << endl << "Employee Job: " << E.Job;
    cout << endl << "Employee House No.: " << E.Add.HouseNo;
    cout << endl << "Employee City: " << E.Add.City;
    cout << endl << "Employee Pin Code: " << E.Add.PinCode;
    cout << endl;

    return(0);
}
```

Structures (Recap)

- Example (continued...)


Microsoft Visual Studio Debug Console

```
Enter Employee ID : 1
Enter Employee Name : Naveed
Enter Employee Job : Professor
Enter Employee House No. : 22
Enter Employee City : Islamabad
Enter Employee Pin Code : 11111

Details of Employee :
Employee ID: 1
Employee Name: Naveed
Employee Job: Professor
Employee House No.: 22
Employee City: Islamabad
Employee Pin Code: 11111
```

Structures (Recap)

- Some important points to remember:

  - ❑ Aggregate I/O is **not allowed**. I/O must be performed on a member by member basis.
  - ❑ Aggregate assignment is allowed. All data members (fields) are copied (**if both structure variables are of same type**)
  - ❑ Aggregate arithmetic is **not allowed**.
  - ❑ Aggregate comparison is **not allowed**. Comparisons must be performed on a member by member basis.
  - ❑ A struct is a valid return type for a value returning function.