Data Structures and Object Oriented Programming

Lecture 2

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

Last Class

- Any questions regarding:
  - **Type Casting**
  - **Dynamic Memory Allocation**

Administrivia

Lectures available here:
naveedanwarbhatti.github.io/DS&OOP.html

For any query use:
naveed.bhatti@mail.au.edu.pk

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.

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

```
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;
}
```

Structures (Recap)

- Exercise: Find the output of the following program

```
struct MyBox
{
    int length, breadth, height;
};

void dimension(MyBox M)
{
    cout << M.length << "x" << M.breadth << "x";
    cout << M.height << endl;
}
```

```
int main()
{
    MyBox B1 = { 10, 15, 5 }, B2, B3;
    ++B1.height;
    dimension(B1);
    B3 = B1;
    ++B3.length;
    B3.breadth++;
    dimension(B3);
    B2 = B3;
    B2.height += 5;
    B2.length--;
    dimension(B2);

    return 0;
}
```

Output:

```
10x15x6
11x16x6
10x16x11
```

Nested Structures (Recap)

- Example

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

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

struct Employee
{
    int Id;
    char Name[30];
    char Job[30];
    struct Address Add;
};

int main()
{
    Employee E;

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

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

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

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

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

    cout << "\n Enter Employee Pin code : ";
    cin >> E.Add.PinCode;

    cout << "\n Details of Employees";
    cout << "\n Employee Id : " << E.Id;
    cout << "\n Employee Name : " << E.Name;
    cout << "\n Employee Job : " << E.Job;
    cout << "\n Employee House No : " << E.Add.HouseNo;
    cout << "\n Employee City : " << E.Add.City;
    cout << "\n Employee House No : " << E.Add.PinCode;

    return(0);

}
```

Structures (Recap)

- Example (continued…)

```
Enter Employee Id : 1212

Enter Employee Name : Naveed

Enter Employee Job : Professor

Enter Employee House No : 1

Enter Employee City : Newyork

Enter Employee Pin code : 1

Details of Employees
Employee Id : 1212
Employee Name : Naveed
Employee Job : Professor
Employee House No : 1
Employee City : Newyork
Employee House No : 1
```

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.