FAST NATIONAL UNIVERSTIY OF COMPUTER AND EMERGING SCIENCES, PESHAWAR

DEPARTMENT OF COMPUTER SCIENCE

CL217 – OBJECT ORIENTED PROGRAMMING LAB

STRUCTURES IN C++

LAB MANUAL # 03

Instructor: Fariba Laiq

SEMESTER SPRING 2023

OBJECT ORIENTED PROGRAMMING LANGUAGE

Table of Contents

Structure

Structure is a collection of variables under a single name. Variables can be of any type: int, float, char etc. The main difference between structure and array is that arrays are collections of the same data type and structure is a collection of variables under a single name.

Why we need Structures?

As we noticed arrays are very powerful device that allow us to group large amounts of data together under a single variable name.

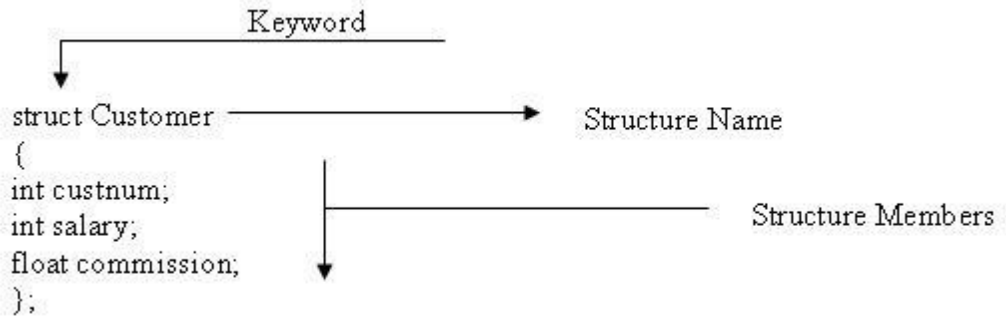
How would you write a program if, instead of being asked for simple list of either integers or characters, you were asked to combine integers, floating point numbers, and string with one variable name? You could not do it by using array.

It is actually quit often we want to group logically connected data that are of different types together. Just think about writing a program to store student record. You would need string for your name, integers to store the ID number, and a floating-point number to store the grades.

Declaring a Structure

Example:

Three variables: custnum of type int, salary of type int, commission of type float are structure members and the structure name is Customer. This structure is declared as follows:



In the above example, it is seen that variables of different types such as int and float are grouped in a single structure name Customer.

Arrays behave in the same way, declaring structures does not mean that memory is allocated. Structure declaration gives a skeleton or template for the structure.

```
struct st_name
{
        type l;
        type m;
        type n;
} ;
```

Where

st_name Represents the structure name. It is also called **Structure tag**. It is used to declare variables of structure type.

type Represent the type of the variable.

l,m,n Represent members of the structure. These may have different or same data type. The members are enclosed in braces.

A semicolon after the closing bracket (**} ;**) indicates the end of the structure.

Each member of a structure must have unique name but different structures may have same names.

A structure is first defined and then variables of structure type are declared. A variable of a structure type is declared to access data in the members of the structure.

Example: Declare a structure with ***address*** as a tag and having two members ***name*** of character type and ***age*** of integer type.

```
struct address
{
        char name [15];
        int age;
} ;
```

The structure member **name** is of character type with 15 characters length (including the null character) and **age** is of integer type.

Structure Variables

After declaring the structure, the next step is to define a structure variable.

A structure is a collection of data items or elements. It is defined to declare its variables. These are called **structure variables**. A variable of structure type represents the members of its structure.

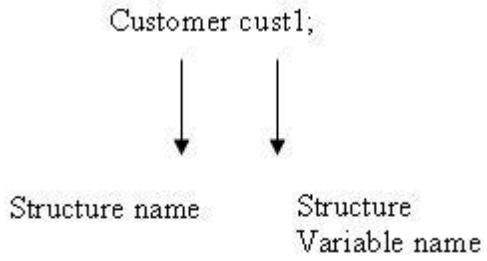
When a structure type variable is declared a space is reserved in the computer memory to hold all members of the structure. The memory occupied by a structure variable is equal to the sum of the memory occupied be each member of the structure.

A structure can be defined prior to or inside the main() function. If it is defined inside the main() function then the structure variable can be declared only inside the main function. If it is declared prior to the main() function, its variables can be defined anywhere in the program.

```
struct address
{
        char city[15];
        int pcode;
} ;
address taq , aye;
```

In the above example, **city** and **pcode** are members of the structure **address**. The variable **taq** and **aye** are declared as structure variables.

The structure **address** has two members **city** and **pcode**. The variable **city** occupies 15 bytes and **pcode** occupies 2 bytes. Thus, each variable of this structure will occupy 17 bytes space in memory, i.e. 15 bytes for **city** and 2 bytes **pcode**.



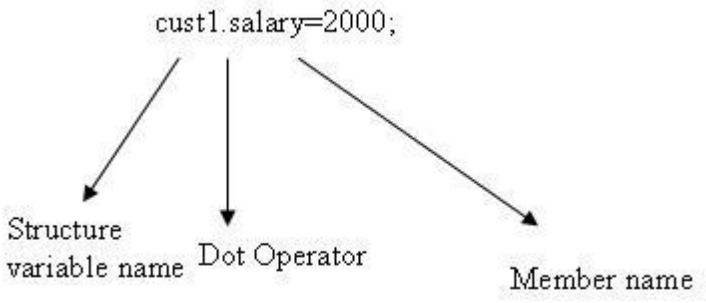
How to access structure members in C++?

To access structure members, the operator used is the dot operator denoted by (.). The dot operator for accessing structure members is used thusly:

```
Structure_Variable_name.member_name;
```

For example:

A programmer wants to assign 2000 for the structure member salary in the above example of structure Customer with structure variable cust1 this is written as:




```
// A Complete example by declaring, using structs as well as data members.
#include <iostream>
#include <conio.h>
#include <string>
using namespace std;
struct employe
{
// members of the structures
string name;
double hours_worked;
double salary;
};
int main()
{
      employe a,b; //structure variables

      //accessing structure variables
      a.name="Asia";
      a.hours_worked=30;
      a.salary=a.hours_worked*50;

      b.name="Naveed";
      b.hours_worked=60;
      b.salary=b.hours_worked*50;
```

```
	//*for 1st employee
	cout<<"Employee Name:\t"<<a.name<<endl;
	cout<<"Hours worked:\t"<<a.hours_worked<<endl;
	cout<<"Salary:\t\t"<<a.salary<<endl;

	//*for second employee
	cout<<"Employee Name:\t"<<b.name<<endl;
	cout<<"Hours worked:\t"<<b.hours_worked<<endl;
	cout<<"Salary:\t\t"<<b.salary<<endl;

}

/*Sample Output

Employee Name:  Asia
Hours worked:   30
Salary:         1500
Employee Name:  Naveed
Hours worked:   60
Salary:         3000
```

Members of Structures

Structures in C++ can contain two types of members:

Data Member: These members are normal C++ variables. We can create a structure with variables of different data types in C++.

Member Functions: These members are normal C++ functions. Along with variables, we can also include functions inside a structure declaration.

Example:

```
// Data Members
int roll;
int age;
int marks;

// Member Functions
void printDetails()
{
	cout<<"Roll = "<<roll<<"\n";
	cout<<"Age = "<<age<<"\n";
	cout<<"Marks = "<<marks;
}
```

In the above structure, the data members are three integer variables to store *roll number, age and marks* of any student and the member function is *printDetails()* which is printing all of the above details of any student.

Structs of Arrays (Array within Structure)

```
#include <iostream>
#include <conio.h>
#include <string.h>
using namespace std;
struct STUDENT
{
//members of structure
int idNum;
int testScores[3];
int finalExam;

//initialize members of structure with 0
int main()
{
STUDENT s1,s2;
// initializing s1 elements one by one without function
s1.idNum=111;
s1.testScores[0]=95;
s1.testScores[1]=80;
s1.testScores[2]=98;
s1.finalExam=100;

}
/*
 ID : 111   Srcor-1 : 95    Srcor-2 : 80    Srcor-3 : 98Finale : 100
*/
```

Arrays of Structs

What is an array of structures?

Like other primitive data types, we can create an array of structures.

Instead of declaring multiple variables we can also declare an array of structure in which each element of the array will represent a structure variable.

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

struct Point {
	int x, y;
};

int main()
{
	// Create an array of structures
	struct Point arr[10];

	// Access array members
	arr[0].x = 10;
	arr[0].y = 20;

	cout << arr[0].x << " " << arr[0].y;

	return 0;
}
```

Output

```
10    20
```

```
#include <iostream>
using namespace std;
struct Student
{
	string name;
	float gpa;
	int marks;
};
int main()
{
	Student s[3];
	for(int i=0; i<3; i++)
	{
		cout<<"Enter name: "<<endl;
		cin>>s[i].name;
		cout<<"Enter marks: "<<endl;
		cin>>s[i].marks;
		cout<<"Enter gpa: "<<endl;
		cin>>s[i].gpa;
		cout<<"\n\n";
	}
	for(int i=0; i<3; i++)
	{
		cout<<"name: "<<endl;
		cout<<s[i].name<<endl;
		cout<<"gpa: "<<endl;
		cout<<s[i].gpa<<endl;;
		cout<<"marks: "<<endl;
		cout<<s[i].marks<<endl;
		cout<<"\n\n";
	}
}
```

Output:

```
Enter name:
fariba
Enter marks:
90
Enter gpa:
3.5

Enter name:
Ali
Enter marks:
70
Enter gpa:
3.2

Enter name:
Fahad
Enter marks:
100
Enter gpa:
3.88

name: fariba
gpa: 3.5
marks: 90

name: Ali
gpa: 3.2
marks: 70

name: Fahad
```

gpa: 3.88
marks: 100

What is a structure pointer?

Like primitive types, we can have pointer to a structure. If we have a pointer to structure, members are accessed using arrow (->) operator instead of the dot (.) operator.

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

struct Point {
    int x, y;
};

int main()
{
    Point p1, *p2;    // structure variables

    p1.x=2;
    p1.y=3;

    // p2 is a pointer to structure p1
    p2 = &p1;

    // Accessing structure members using
    // structure pointer
    cout << p2->x << " " << p2->y;
    return 0;
}

Output
2   3
```

Structure and Function in C++

Using function, we can pass structure as function argument and we can also return structure from function.

Structure can be passed to function through its variable therefore passing structure to function or passing structure object to function is same thing because structure object represents the structure. Like normal variable, structure variable (structure object) can be passed by value or by references / addresses

Passing Structure by Value

In this approach, the structure object is passed as function argument to the definition of function, here object is representing the members of structure with their values.

Example for passing structure object by value

```
#include<iostream>
using namespace std;
struct Employee
{
    int Id;
    char Name[25];
    int Age;
    long Salary;
};

void Display(Employee);   // function declaration
int main()
{
     // Initializing structure variables
     Employee Emp = {1,"Aisha",29,45000};
     Display(Emp);   // calling function
}

void Display(Employee E)
{
    cout << "\n\nEmployee Id : " << E.Id;
    cout << "\nEmployee Name : " << E.Name;
    cout << "\nEmployee Age : " << E.Age;
    cout << "\nEmployee Salary : " << E.Salary;
}
```

Output:

```
Employee Id : 1
Employee Name : Aisha
Employee Age : 29
Employee Salary : 45000
```

Passing Structure by Reference

In this approach, the reference/address structure object is passed as function argument to the definition of function.

Example for passing structure variable by reference

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

struct Employee
{
    int Id;
    char Name[25];
    int Age;
    long Salary;
};

void Display(Employee*);   // function declaration

int main()
{
    Employee Emp = {1,"Kashif",29,45000};

    Display(&Emp);

}

void Display(Employee *E)
{
    cout << "\n\nEmployee Id : " << E->Id;
    cout << "\nEmployee Name : " << E->Name;
    cout << "\nEmployee Age : " << E->Age;
    cout << "\nEmployee Salary : " << E->Salary;
```

```
}
Output:
Employee Id : 1
Employee Name : Kashif
Employee Age : 29
Employee Salary : 45000
```

Function Returning Structure

Structure is user-defined data type, like built-in data types structure can be returned from function.

Example for Function Returning Structure

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

struct Employee
{
         int Id;
         char Name[25];
         int Age;
         long Salary;
};

Employee Input();            //Statement 1
int main()
{
    Employee Emp;

    Emp = Input();  // calling function

    cout << "\n\nEmployee Id : " << Emp.Id;
    cout << "\nEmployee Name : " << Emp.Name;
    cout << "\nEmployee Age : " << Emp.Age;
    cout << "\nEmployee Salary : " << Emp.Salary;

}
Employee Input()
{
```

```
    Employee E;   // declaring structure variable

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

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

    cout << "\nEnter Employee Age : ";
    cin >> E.Age;

    cout << "\nEnter Employee Salary : ";
    cin >> E.Salary;

    return E;             //Statement   2
}
```

Output:

```
Enter Employee Id : 1
Enter Employee Name : Amir
Enter Employee Age : 23
Enter Employee Salary : 234235

Employee Id : 1
Employee Name : Amir
Employee Age : 23
Employee Salary : 234235
```

In the above example, statement 1 is declaring Input() with return type Employee. As we know structure is user-defined data type and structure name acts as our new user-defined data type, therefore we use structure name as function return type.

Input() have local variable E of Employee type. After getting values from user statement 2 returns E to the calling function and display the values.