Computer Programing [Lab]

Lab#02: Structs

Agenda

- Structs
- Initialization and Accessing Struct Members
- Input and Output in Structs
- Passing Struct to Functions
- Struct in Arrays
- Nested Structs

Struct

Structure is the collection of variables of different types under a single name.

Arrays is also collection of data but arrays can hold data of only one type whereas **structure** can hold data of one or more types.

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

The general syntax of a struct in C++ is:

```
struct structName
{
    dataType1 identifier1;
    dataType2 identifier2;
        .
        .
        .
    dataTypen identifiern;
};
```

The **struct** keyword defines a structure type followed by an identifier (name of the structure). Then inside the curly braces, you can declare one or more members (declare variables inside curly braces) of that structure. For example:

```
struct person {
      string name;
      int age;
      float salary;
};
```

Here a structure person is defined which has three members: name, age and salary.

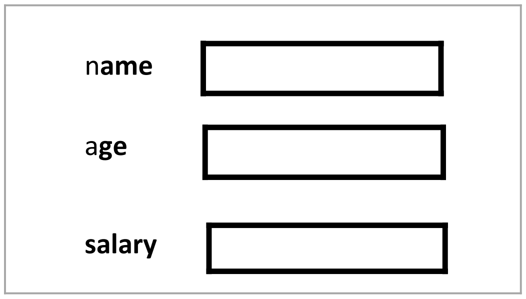
When a structure is created, no memory is allocated. The structure definition is only the blueprint for the creating of variables. You can imagine it as a data type. When you define an integer as below:

```
int foo;
```

The int specifies that, variable foo can hold integer element only. Similarly, structure definition only specifies that, what property a structure variable holds when it is defined.

Once you declare a structure person as above. You can define a structure variable as:

```
person bill;
```



Here, a structure variable bill is defined which is of type structure person. When structure variable is defined, then only the required memory is allocated by the compiler. Considering you have either 32-bit or 64-bit system, the memory of float is 4 bytes, memory of int is 4 bytes and memory of char is 1 byte. Hence, 9 bytes of memory is allocated for structure variable bill.

Accessing Structure Members:

In C++, the dot (.) is an operator called the **member access operator**.

To access any member of a structure, we use the **member access operator (.)**.

bill.age = 50;

bill.salary = 25000.00

cin >> bill.age;

infile >> bill.salary

Example:

```
// This program demonstrates the use of a record (C++
struct) struct PersonRec
{
        string name;
        int age;
        float salary;
};
void main()
{
        PersonRec thePerson;
        cout << "Enter your name:
        "; cin >> thePerson.name;
        cout << "Enter age: ";
        cin >> thePerson.age; cout
        << "Enter salary: "; cin
        >> thePerson.salary;
        cout << "\n\nHello " << thePerson.name << ". How are you? \n";
        cout << "\nCongratulations on reaching the age of " << thePerson.age
        << " having salary of " <<thePerson.salary<<" .\n";
}
```

Structures as Function Arguments:

You can pass a structure as a function argument in very similar way as you pass any other variable or pointer. You would access structure variables in the similar way as you have accessed in the above example:

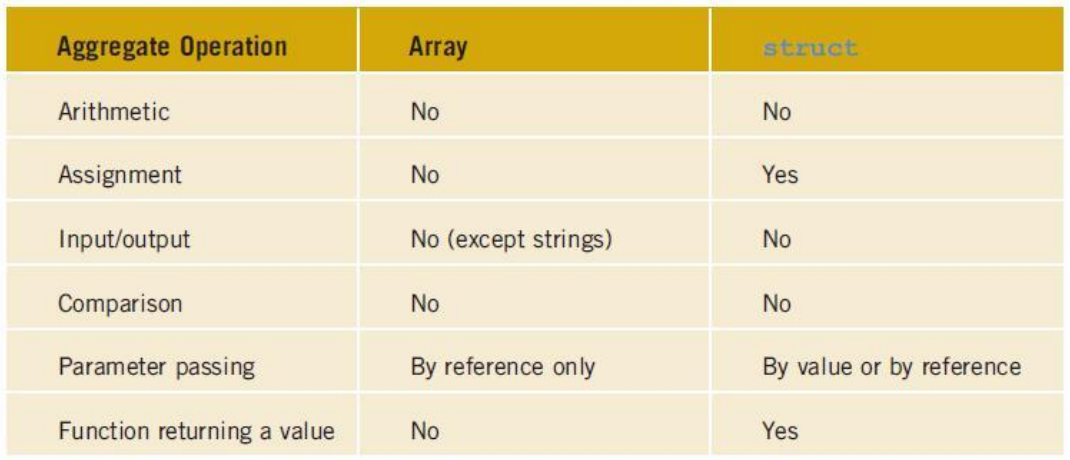
```
struct movies_t {
  string title;
  int year;
} mine, yours;
```

```
void printmovie (movies_t movie)
{
  cout << movie.title;
  cout << " (" << movie.year << ")\n";
}
```

```
void printmovie (movies_t movie);

int main ()
{
  string mystr;

  mine.title = "2001 A Space Odyssey";
  mine.year = 1968;

  cout << "Enter title: ";
  getline (cin,yours.title);
  cout << "Enter year: ";
  getline (cin,mystr);
  stringstream(mystr) >> yours.year;

  cout << "My favorite movie is:\n ";
  printmovie (mine);
  cout << "And yours is:\n ";
  printmovie (yours);
  return 0;
}
```

```
Enter title: Alien
Enter year: 1979

My favorite movie is:
 2001 A Space Odyssey (1968)
And yours is:
 Alien (1979)
```

Arrays vs. Structs



Aggregate Operation	Array	struct
Arithmetic	No	No
Assignment	No	Yes
Input/output	No (except strings)	No
Comparison	No	No
Parameter passing	By reference only	By value or by reference
Function returning a value	No	Yes

Structs in Arrays Example:

Suppose a company has 50 full-time employees. We need to print their monthly paychecks and keep track of how much money has been paid to each employee in the year-to-date. First, let's define an employee's record:

```
// This program demonstrates the use of an array of
structs struct PersonRec
{
string lastName;
string firstName;
int age;
};

void main(void)
{
       PersonRec people[10]; //a variable of the array
       type for (int i = 0; i < 10; i++) {

              cout << "Enter first name: ";
              cin >> people[i].firstName;
              cout << "Enter last name: ";
              cin >> people[i].lastName;
              cout << "Enter age: ";
              cin >> people[i].age;
       }
       for (int i = 0; i < 10; i++)
       {
              cout << people[i].firstName;
              cout << "  " << people[i].lastName << setw(10)<< people[i].age;
       }
}
```

Nested Structs (Struct within a Struct) Example:

```
// this program demonstrates the use of a nested struct

struct GradeRec
{
	float percent;
	char grade;
};
struct StudentRec
{
	string lastName;
	string firstName;
	int age;
	GradeRec courseGrade;
};
void main()
{
StudentRec student;
cout << "Enter first name: ";
cin >> student.firstName;
cout << "Enter last name: ";
cin >> student.lastName;
cout << "Enter age: ";
cin >> student.age;
cout << "Enter overall percent: ";
cin >> student.courseGrade.percent;
if(student.courseGrade.percent >= 90)
{
	student.courseGrade.grade = 'A';
}
else if(student.courseGrade.percent >= 75)
{
	student.courseGrade.grade = 'B';
}
else
{
	student.courseGrade.grade = 'F';
}
cout << "\n\nHello " << student.firstName << student.lastName
<< ". How are you?\n";
cout << "\nCongratulations on reaching the age of " << student.age << ".\n";
cout << "Your overall percent score is "<<student.courseGrade.percent
<< " for a grade of "<<student.courseGrade.grade;
}
```

```
// Output
Enter first name: Sally
Enter last name: Smart
Enter age: 19
Enter overall percent: 98
Hello Sally Smart. How are you?
Congratulations on reaching the age of 19.
Your overall percent score is 98 for a grade of A
```