Fundamentals of Programming: Structure in C++

Abdul Haseeb

Agenda

- Need of Structure
- How to Create a Structure in C++
- Members of a Structure
- Dot . Operator
- Arrays inside Structure
- Functions inside Structure
- Array vs Structure
- Unnamed Structure

Concept of Structure

The Problem:

You want to store some information about a person: his/her name, citizenship number and gender:

Create different variables name, citNo, gender.

If, in the future, you would want to store information about multiple persons, what you will do ?

Concept of structure

you'd need to create different variables for each information per person: name1, citNo1, salary1, name2, citNo2, salary2

You can easily visualize how big and messy the code would look. Also, since no relation between the variables (information) would exist, it's going to be a scary task.

Concept of structure

- A better approach will be to have a collection of all related information under a single name Person, and use it for every person. Now, the code looks much cleaner, readable and efficient as well.
- This collection of all related information under a single name Person is a structure.

What is a structure?

- Collection of variables of different data types under a single name.
- It is a **user defined data type** which groups together items of possibly **different data types** under a **single type**.

How to create a structure in C++

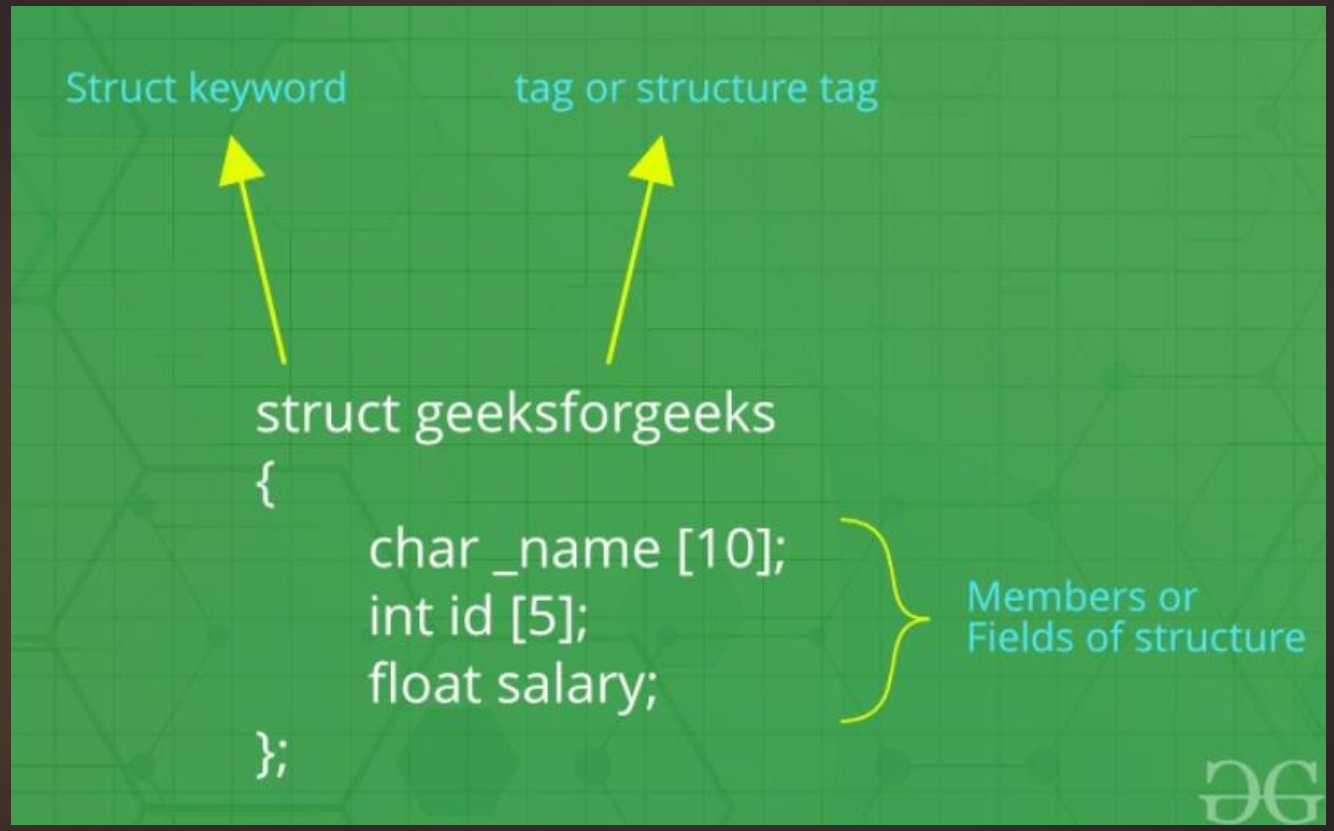
- struct keyword is used to create a structure in C++

```
struct structureName{
    member1;
    member2;
    member3;
    .
    .
    .
    memberN;
};
```

Types of structure members

- **Data Members:**
  - Normal variables/Arrays in a structure
- **Member Functions:**
  - Normal C++ functions defined inside a structure

Example struct



```
//Creating a structure named person, with data members called age,name and gender
struct Person{
    float age;
    string name;
    char gender;
};
```

Declaration of a struct and its variables

Creating a structure variable


```
int main(){

//Creates two variables, p1 and p2 of Person data Type
Person p1;
Person p2;


return 0;
}
```

Dot (.) operator/ Member access operator

- It is used to access the members of structure:
  - For initializing/modification of structure member variables
  - For accessing the values of structure member variables

```
int main(){

Person p1;
p1.age=35;
p1.name="Ayan";
p1.gender='M';

return 0;
}
```

Initializing the variables of structure

```
int main(){

Person p1;
p1.age=35;
p1.name="Ayan";
p1.gender='M';

cout<<"Name of person 1: "<<p1.name<<endl;
cout<<"Age of person 1: "<<p1.age;

return 0;
}
```

```
C:\Users\92306\Deskt

Name of person 1: Ayan
Age of person 1: 35
--------------------------------
Process exited after 0.0511 seconds with ret
urn value 0
Press any key to continue . . .
```

Accessing the variables of structure

Task

- Create a structure called employee, an employee has name, emp_id and emp_salary.
- Inside main function assign structure members a value and print them using Dot Operator.

```
struct Dept{
    string dept_name;
    string emp_names[20];
    int building_number;
};
```

Using Array in a structure

```
Dept d1;
d1.name="Computer Science";
d1.building_number=4;
d1.emp_names[0]="Ahmed"


for(int i=1;i<20;i++){
    cin>>d1.emp_names[i];
}
```

Using array in a structure

Memory allocation in structure

- Once you declare a structure person as above. You can define a structure variable as:
- *Person ahmed;*
- Here, a structure variable **ahmed** is defined which is of type structure Person.
- When structure variable is defined, only then the required memory is allocated by the compiler.
- Memory of int is 4 bytes and memory of string is 8 byte. Hence, 12 bytes of memory is allocated for structure variable ahmed.

Scope of structure

- So far we have used global scope
- If we define a structure inside main function, then we can only use it inside the main function, not outside the main function

Creating a function inside structure

```
struct Person{
    string name;
    int age;
    char gender;

    void display_person_info(){
        cout<<"Person name: "<<name<<endl;
        cout<<"Person Age:"<<age<<endl;
        cout<<"Person Gender:"<<gender<<endl;
    }
};
```

Accessing the function of a structure

```
int main(){
Person p1;

p1.age=43;
p1.name="Bilal";
p1.gender='M';

p1.display_person_info();

return 0;
}
```

Array of Structures

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

```
int main(){

person p[3];


for(int i=0; i<3; i++){
    cout<<"Enter name of person "<<i+1;
    cin>>p[i].name;

    cout<<"Enter Age of person "<<i+1;
    cin>>p[i].age;

}
```

Array Vs Structure

Array

- Collection of homogeneous data.
- Data is accessed using Index.
- Allocates static memory.
- Access takes less time than structure.

Structure

- Collection of heterogeneous data.
- Data is accessed using dot (.).
- Allocates dynamic memory.
- Access takes more time than array.