Programming Fundamentals

STRUCTURES

Why structure?

- Suppose, you want to store information about a person: his/her name, citizenship number, and salary. You can create different variables name, citNo and salary to store this information.
- What if you need to store information of more than one person? Now, you need to create different variables for each information per person: name1, citNo1, salary1, name2, citNo2, salary2, etc.
- A better approach would be to have a collection of all related information under a single name Person structure and use it for every person.
- Methods can also be written in structure.

What is structure?

- In C programming, a struct (or structure) is a collection of variables (can be of different types) under a single name. The data elements in structure are called structure elements, members or fields.

```
struct structureName
{ dataType member1;
dataType member2; ...
};
```

How to define structures?

- To define a struct, the struct keyword is used.

```
struct Person
{
    char name[50];
    int citNo;
    float salary;
};
```

Creating structure variables

```
struct Person
{
char name[50];
int citNo;
float salary;
};
int main(){
Person person1, person2, p[20];
return 0;}
```

Creating structure variables (cont..)

- Another way of creating a struct variable is

```
struct Person
{
char name[50];
int citNo;
float salary;}
person1, person2, p[20];
```

Access members of a structure

- There are two types of operators used for accessing members of a structure.
- . - Member operator
- -> - Structure pointer operator (used with pointers)
- Suppose, you want to access the salary of person2. Here's how you can do it.

- `person2.salary;`

Nested structure I

```
struct complex
{
int imag;
float real;
};
struct number
{
complex comp;
int integers;
}
num1, num2;
```

Nested structure II

- Suppose, you want to set imag of num2 variable to 11. Here's how you can do it:
- `num2.comp.imag = 11;`

Structure and pointer

```
#include<iostream>
using namespace std;
struct person
{
   int age;
   float weight;
};
```

```
int main()
{
    person *personPtr, person1;
    personPtr = &person1;

    cout<<"Enter age: "<<endl;
    cin>>personPtr->age;

    cout<<"Enter weight: "<<endl;
    cin>>personPtr->weight;

    cout<<"Displaying:\n";
    cout<<personPtr->age;
    cout<<personPtr->weight;
    return 0;
}
```

Passing struct to function

```
struct student
{
   char name[50];
   int age;
};
void display(student );

int main()
{
    student s1;
   cout<<"Enter name: "<<endl;
   cin>>s1.name;
```

Passing struct to function (cont..)

```
cout<<"Enter age: "<<endl;
cin>>s1.age;
display(s1);   // passing struct as an argument
  return 0;
}
void display(student s)
{
  cout<<“Displaying information”<<endl;
  cout<<s.name;
  cout<<s.age;
}
```

References

- C++ How to Program
  By Deitel & Deitel
- The C++ Programming Language
  By Bjarne Stroustrup
- Object oriented programming using C++ by Tasleem Mustafa, Imran Saeed, Tariq Mehmood, Ahsan Raza
- https://www.tutorialspoint.com/cplusplus
- http://ecomputernotes.com/cpp/introduction-to-oop
- http://www.cplusplus.com/doc/tutorial
- http://www.c4learn.com/c-programming/c-dangling-pointer-causes/
- https://www.guru99.com/c-loop-statement.html
- www.w3schools.com
- https://www.geeksforgeeks.org/