forbeginners.c   tutorials.out

CSE1062 | CCS1063 ‘Practicals’ {

[Fundamentals of Computer Programming]

< Tutorial Session 10 - Struct >

}

What is an struct?

A struct (or structure) is a collection of variables (can be of different types) under a single name.

Before you can create structure variables, you need to define its data type. To define a struct, the struct keyword is used.

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

https://www.programiz.com/c-programming/c-structures

Example

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

Here, a derived type struct Person is defined. Now, you can create variables of this type.

Create struct Variables

When a struct type is declared, no storage or memory is allocated. To allocate memory of a given structure type and work with it, we need to create variables.

Ex:1

```c
struct Person {
  // code
};

int main() {
  struct Person person1,
person2, p[20];
  return 0;
 }
```

Ex:2

```c
struct Person {
   // code
 } person1, person2,

 p[20];
```

person1 and person2 are struct Person variables

p[] is a struct Person array of size 20.

Access Members of a Structure

There are two types of operators used for accessing members of a structure.

* . - Member operator
* → - Structure pointer operator (will be discussed in the next tutorial)

Suppose, you want to access the salary of person2. Here's how you can do it.

```
person2.salary
```

Example

Output

```
Name: George Orwell
Citizenship No.: 1984
Salary: 2500.00
```

In this program, we have created a struct named Person. We have also created a variable of Person named person1

https://www.programiz.com/c-programming/c-structures

```c
#include <stdio.h>
#include <string.h>

// create struct with person1 variable
struct Person {
  char name[50];
  int citNo;
  float salary;
} person1;

int main() {

  // assign value to name of person1
  strcpy(person1.name, "George Orwell");

  // assign values to other person1
variables
  person1.citNo = 1984;
  person1. salary = 2500;

  // print struct variables
  printf("Name: %s\n", person1.name);
  printf("Citizenship No.: %d\n",
person1.citNo);
  printf("Salary: %.2f", person1.salary);

  return 0;
  }
```

Keyword typedef

We use the typedef keyword to create an alias name for data types. It is commonly used with structures to simplify the syntax of declaring variables.

Example

```c
struct Distance{
  int feet;
  float inch;
};
int main() {
  struct Distance d1, d2;
  }
```

We can use typedef to write an equivalent code with a simplified syntax:

```c
typedef struct Distance {
  int feet;
  float inch;
} distances;

int main() {
  distances d1, d2;
  }
```

https://www.programiz.com/c-programming/c-structures

Example

Output

```
Name: George Orwell

Citizenship No.: 1984

Salary: 2500.00
```

```c
#include <stdio.h>
#include <string.h>
// struct with typedef person
typedef struct Person {
  char name[50];
  int citNo;
  float salary;
} person;
int main() {

  // create  Person variable
  person p1;

  // assign value to name of p1
  strcpy(p1.name, "George Orwell");

  // assign values to other p1 variables
  p1.citNo = 1984;
  p1. salary = 2500;

  // print struct variables
  printf("Name: %s\n", p1.name);
  printf("Citizenship No.: %d\n", p1.citNo);
  printf("Salary: %.2f", p1.salary);
  return 0;
  }
```

https://www.programiz.com/c-programming/c-structures

Local Variables

Variables that are declared inside a function or block are called local variables.

Local variables are not known to functions outside their own.

```
#include <stdio.h>

int main () {

  /* local variable declaration */
  int a, b;
  int c;

  /* actual initialization */
  a = 10;
  b = 20;
  c = a + b;

  printf ("value of a = %d, b = %d and c = 
%d\n", a, b, c);

  return 0;
}
```

https://www.tutorialspoint.com/cprogramming/c_scope_rules.htm

Global Variables

Global variables are defined outside a function, usually on top of the program. Global variables hold their values throughout the lifetime of your program and they can be accessed inside any of the functions defined for the program.

```c
#include <stdio.h>

/* global variable declaration */
int g;

int main () {

  /* local variable declaration */
  int a, b;

  /* actual initialization */
  a = 10;
  b = 20;
  g = a + b;

  printf ("value of a = %d, b = %d and g = 
%d\n", a, b, g);

  return 0;
}
```

https://www.tutorialspoint.com/cprogramming/c_scope_rules.htm

Example

What will be the output?

```c
#include <stdio.h>

/* global variable declaration */
int g = 20;

int main () {

  /* local variable declaration */
  int g = 10;

  printf ("value of g = %d\n",  g);

  return 0;
}
```

https://www.tutorialspoint.com/cprogramming/c_scope_rules.htm

```
Thanks; {
    'Do you have any questions?'

        < bgamage@sjp.ac.lk >

}
```