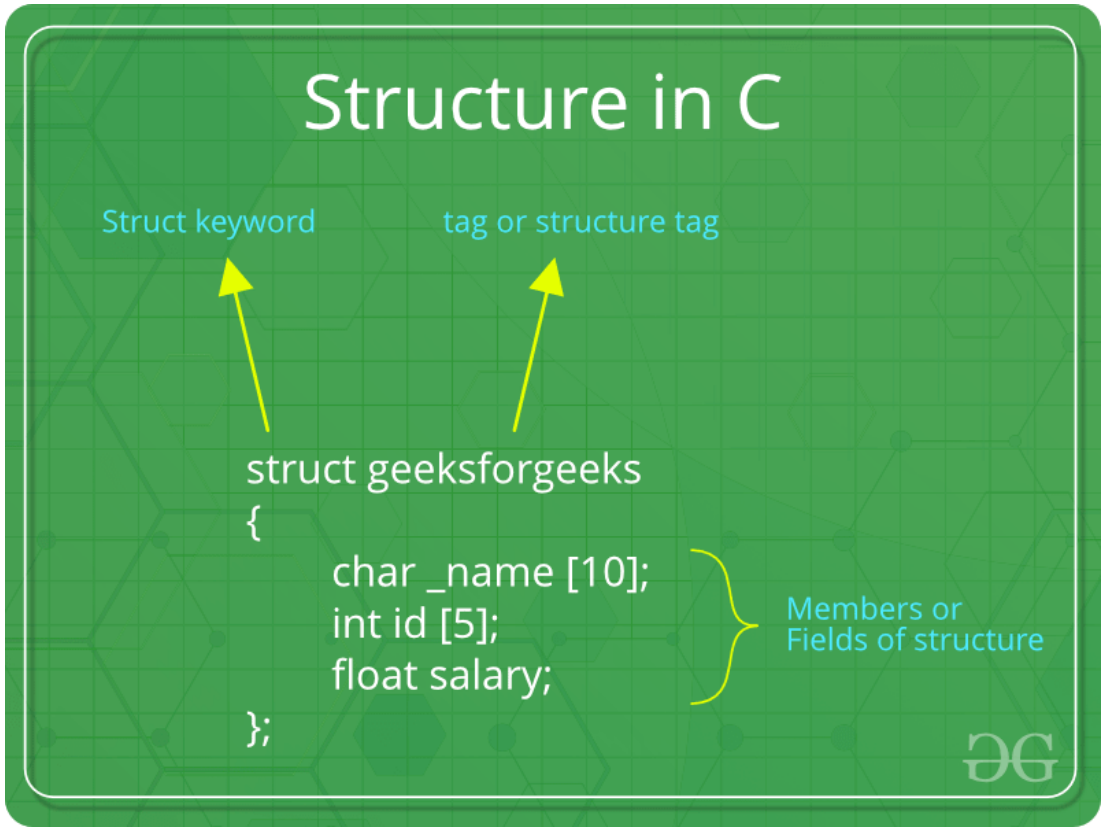
Session 12

Mostafa Akram

Structures



- Structures (also called structs) are a way to group several related variables into one place. Each variable in the structure is known as a member of the structure.

- Unlike an array, a structure can contain many different data types (int, float, char, etc.).

Create a Structure

```
struct MyStructure {    // Structure declaration
  int myNum;            // Member (int variable)
  char myLetter;        // Member (char variable)
}; // End the structure with a semicolon
```

Create a Structure

- Use the struct keyword inside the main() method, followed by the name of the structure and then the name of the structure variable

```
struct myStructure {
  int myNum;
  char myLetter;
};

int main() {
  struct myStructure s1;
  return 0;
}
```

Access Structure Members

- To access members of a structure, use the dot syntax (.)

```c
// Create a structure called myStructure
struct myStructure {
  int myNum;
  char myLetter;
};

int main() {
  // Create a structure variable of myStructure called s1
  struct myStructure s1;

  // Assign values to members of s1
  s1.myNum = 13;
  s1.myLetter = 'B';

  // Print values
  printf("My number: %d\n", s1.myNum);
  printf("My letter: %c\n", s1.myLetter);

  return 0;
}
```

Access Structure Members

- assign values to members of a structure variable at declaration time

```c
// Create a structure
struct myStructure {
  int myNum;
  char myLetter;
  char myString[30];
};

int main() {
  // Create a structure variable and assign values to it
  struct myStructure s1 = {13, 'B', "Some text"};

  // Print values
  printf("%d %c %s", s1.myNum, s1.myLetter, s1.myString);

  return 0;
}
```

create multiple structure variables with different values

```c
// Create different struct variables
struct myStructure s1;
struct myStructure s2;

// Assign values to different struct variables
s1.myNum = 13;
s1.myLetter = 'B';

s2.myNum = 20;
s2.myLetter = 'C';
```

Copy Structures

```
struct myStructure s1 = {13, 'B', "Some text"};
struct myStructure s2;

s2 = s1;
```

Modify Values

```c
struct myStructure {
  int myNum;
  char myLetter;
  char myString[30];
};

int main() {
  // Create a structure variable and assign values to it
  struct myStructure s1 = {13, 'B', "Some text"};

  // Modify values
  s1.myNum = 30;
  s1.myLetter = 'C';
  strcpy(s1.myString, "Something else");

  // Print values
  printf("%d %c %s", s1.myNum, s1.myLetter, s1.myString);

  return 0;
}
```

Initialize Structure Members

- Structure members **cannot be** initialized with the declaration.

```
struct Point
{
   int x = 0;  // COMPILER ERROR:  cannot initialize members here
   int y = 0;  // COMPILER ERROR:  cannot initialize members here
};
```

Default Initialization

```
struct Point
{
      int x;
      int y;
};

struct Point p = {0}; // Both x and y are initialized to 0
```

1. Initialization using Assignment Operator

```
struct structure_name str;
str.member1 = value1;
str.member2 = value2;
str.member3 = value3;
.
.
.
```

2. Initialization using Initializer List

```
struct structure_name str = { value1, value2, value3 };
```

3. Initialization using Designated Initializer List

```
struct structure_name str = { .member1 = value1, .member2 = value2, .member3 = value3 };
```

3. Initialization using Designated Initializer List

```
struct structure_name str = { .member1 = value1, .member2 = value2, .member3 = value3 };
```

typedef for Structures

```c
// C Program to illustrate the use of typedef with
// structures
#include <stdio.h>

// defining structure
typedef struct {
    int a;
} str1;

// another way of using typedef with structures
typedef struct {
    int x;
} str2;

int main()
{
    // creating structure variables using new names
    str1 var1 = { 20 };
    str2 var2 = { 314 };

    printf("var1.a = %d\n", var1.a);
    printf("var2.x = %d\n", var2.x);

    return 0;
}
```

Nested Structures

1. Embedded Structure Nesting

```c
// Embedded nested struct

struct member {
  int category;
  struct personal {
    int age;
    char name[30];

  } memberdata;
};

int main() {

struct member Drmostafa;

  printf(" member data 1- category 2- age 3-name");

  scanf("%d", &Drmostafa.category);
  scanf("%d", &Drmostafa.memberdata.age);
  scanf("%s", &Drmostafa.memberdata.name);

  printf("name: %s \nage: %d \ncategory: %d \n", Drmostafa.memberdata.name,
         Drmostafa.memberdata.age, Drmostafa.category);

  return 0;
}
```

Nested Structures

- 2. Separate Structure Nesting

```c
// Seprated Nested struct
struct personal {
  int age;
  char name[30];
};

struct member {
  int category;
  struct personal memberdata;
};

int main() {

  struct member Drmostafa;

  printf(" member data 1- category 2- age 3-name");

  scanf("%d", &Drmostafa.category);
  scanf("%d", &Drmostafa.memberdata.age);
  scanf("%s", &Drmostafa.memberdata.name);

  printf("name: %s \nage: %d \ncategory: %d \n", Drmostafa.memberdata.name,
         Drmostafa.memberdata.age, Drmostafa.category);

  return 0;
}
```

Task

- Repeat edX final project using struct

Links