Static Variables in C

Difficulty Level : Easy • Last Updated : 06 Aug, 2019

Static variables have a property of preserving their value even after they are out of their scope!Hence, static variables preserve their previous value in their previous scope and are not initialized again in the new scope.
Syntax:

```
static data_type var_name = var_value;
```

Following are some interesting facts about static variables in C.

1) A static int variable remains in memory while the program is running. A normal or auto variable is destroyed when a function call where the variable was declared is over.

For example, we can use static int to count a number of times a function is called, but an auto variable can't be used for this purpose.

For example below program prints "1 2"

```

#include<stdio.h>
int fun()
{
int count = 0;
count++;
return count;
}

int main()
{
printf("%d ", fun());
printf("%d ", fun());
return 0;
}

```

Output:

```
1 2
```

But below program prints 1 1

```
#include<stdio.h>
int fun()
{
  int count = 0;
  count++;
  return count;
}

int main()
{
  printf("%d ", fun());
  printf("%d ", fun());
  return 0;
}
```

Output:

```
1 1
```

2) Static variables are allocated memory in data segment, not stack segment. See memory layout of C programs for details.

3) Static variables (like global variables) are initialized as 0 if not initialized explicitly. For example in the below program, value of x is printed as 0, while value of y is something garbage. See this for more details.

```
#include <stdio.h>
int main()
{
    static int x;
    int y;
    printf("%d \n %d", x, y);
}
```

Output:

```
 0
 [some_garbage_value]
```

4) In C, static variables can only be initialized using constant literals. For example, following program fails in compilation. See this for more details.

```
#include<stdio.h>
int initializer(void)
{
    return 50;
}

int main()
{
    static int i = initializer();
    printf(" value of i = %d", i);
    getchar();
    return 0;
}
```

Output

```
  In function 'main':
 9:5: error: initializer element is not constant
      static int i = initializer();
      ^
```

Please note that this condition doesn't hold in C++. So if you save the program as a C++ program, it would compile \and run fine.

5) Static global variables and functions are also possible in C/C++. The purpose of these is to limit scope of a variable or function to a file. Please refer Static functions in C for more details.

6) Static variables should not be declared inside structure. The reason is C compiler requires the entire structure elements to be placed together (i.e.) memory allocation for structure members should be contiguous. It is possible to declare structure inside the function (stack segment) or allocate memory dynamically(heap segment) or it can be even global (BSS or data segment). Whatever might be the case, all structure members should reside in the same memory segment because the value for the structure element is fetched by counting the offset of the element from the beginning address of the structure. Separating out one member alone to data segment defeats the purpose of static variable and it is possible to have an entire structure as static.

Related Articles:

- Static Keyword in C++
- Quiz on Static Keyword
- Static data members in C++
- When are static objects destroyed?
- Interesting facts about static member functions
- Can static functions be virtual?
- Comparison of static keyword in C++ and Java
- Static functions in C