C - Variable Arguments

Sometimes, you may come across a situation, when you want to have a function, which can take variable number of arguments, i.e., parameters, instead of predefined number of parameters. The C programming language provides a solution for this situation and you are allowed to define a function which can accept variable number of parameters based on your requirement. The following example shows the definition of such a function.

```
int func(int, ... ) {
   .
   .
   .
}

int main() {
   func(1, 2, 3);
   func(1, 2, 3, 4);
}
```

It should be noted that the function **func()** has its last argument as ellipses, i.e. three dotes (**...**) and the one just before the ellipses is always an **int** which will represent the total number variable arguments passed. To use such functionality, you need to make use of **stdarg.h** header file which provides the functions and macros to implement the functionality of variable arguments and follow the given steps −

- Define a function with its last parameter as ellipses and the one just before the ellipses is always an **int** which will represent the number of arguments.
- Create a **va_list** type variable in the function definition. This type is defined in stdarg.h header file.
- Use **int** parameter and **va_start** macro to initialize the **va_list** variable to an argument list. The macro va_start is defined in stdarg.h header file.
- Use **va_arg** macro and **va_list** variable to access each item in argument list.
- Use a macro **va_end** to clean up the memory assigned to **va_list** variable.

Now let us follow the above steps and write down a simple function which can take the variable number of parameters and return their average −

Live Demo

```
#include <stdio.h>
#include <stdarg.h>

double average(int num,...) {

   va_list valist;
   double sum = 0.0;
   int i;

   /* initialize valist for num number of arguments */
   va_start(valist, num);

   /* access all the arguments assigned to valist */
   for (i = 0; i < num; i++) {
      sum += va_arg(valist, int);
   }

   /* clean memory reserved for valist */
   va_end(valist);

   return sum/num;
}

int main() {
   printf("Average of 2, 3, 4, 5 = %f\n", average(4, 2,3,4,5));
   printf("Average of 5, 10, 15 = %f\n", average(3, 5,10,15));
}
```

When the above code is compiled and executed, it produces the following result. It should be noted that the function **average()** has been called twice and each time the first argument represents the total number of variable arguments being passed. Only ellipses will be used to pass variable number of arguments.

```
Average of 2, 3, 4, 5 = 3.500000
Average of 5, 10, 15 = 10.000000
```