va_arg() – va_copy() – va_end() – va_start() — Handle Variable Argument List

Last Updated: 2025-09-29

Format

```
#include <stdarg.h>
var_type va_arg(va_list arg_ptr, var_type);
void va_copy(va_list dest, va_list src);
void va_end(va_list arg_ptr);
void va_start(va_list arg_ptr, variable_name);
```

Language Level

ANSI

Threadsafe

Yes

Description

The `va_arg()`, `va_copy()`, `va_end()`, and `va_start()` functions access the arguments to a function when it takes a fixed number of required arguments and a variable number of optional arguments. You declare required arguments as ordinary parameters to the function and access the arguments through the parameter names.

`va_start()` initializes the *arg_ptr* pointer for subsequent calls to `va_arg()`, `va_copy()` and `va_end()`.

The argument *variable_name* is the identifier of the rightmost named parameter in the parameter list (preceding , ...). Use `va_start()` before `va_arg()`. Corresponding `va_start()` and `va_end()` macros must be in the same function.

`va_copy()` initializes *dest* as a copy of *src* , as if `va_start()` had been applied to *dest* followed by the same sequence of uses of `va_arg()` as had previously been used to reach the present state of *src*.

Neither `va_copy()` nor `va_start()` shall be called to reinitialize *dest* without an intervening call to `va_end()` for the same *dest*.

The `va_arg()` function retrieves a value of the given *var_type* from the location given by *arg_ptr*, and increases *arg_ptr* to point to the next argument in the list. The `va_arg()` function can retrieve arguments from the list any number of times within the function. The *var_type* argument must be one of int, long, decimal, double, struct, union, or pointer, or a typedef of one of these types.

The `va_end()` function is needed to indicate the end of parameter scanning. Each call of `va_start()` and `va_copy()` must be matched by a corresponding call to `va_end()` in the same function.

Because it is not always possible for the called function to determine how many arguments there are, the calling function should communicate the number of arguments to the called function. To determine the number of arguments, a function can use a null pointer to signal the end of the list or pass the count of the optional arguments as one of the required arguments. The `printf()` function, for instance, can tell how many arguments there are through the *format-string* argument.

Return Value

The `va_arg()` function returns the current argument. The `va_copy()`, `va_end()` and `va_start()` functions do not return a value.

Example

This example passes a variable number of arguments to a function which prints each argument twice.

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

int vout(int max, ...);

int main(void)
{
   vout(2, "Sat", "Sun");
   printf("\n");
   vout(3, "Mon", "Tues", "Wed");
}

int vout(int max, ...)
{
   va_list arg_ptr;
   va_list args_copy;
   int args;
   char *day;
   va_start(arg_ptr, max);
   va_copy(args_copy, arg_ptr);
   args = 0;
   while(args < max)
   {
      day = va_arg(arg_ptr, char *);
      printf("Day: %s\n", day);
      args++;
   }
   va_end(arg_ptr);

   args = 0;
   while(args < max)
   {
      day = va_arg(args_copy, char *);
      printf("Day: %s\n", day);
      args++;
   }
   va_end(args_copy);
}

/******************  Output should be similar to:  ****************
Day: Sat
Day: Sun
Day: Sat
Day: Sun

Day: Mon
Day: Tues
Day: Wed
Day: Mon
Day: Tues
Day: Wed
*/
```

Related Information

- vfprintf() — Print Argument Data to Stream
- vprintf() — Print Argument Data
- vfwprintf() — Format Argument Data as Wide Characters and Write to a Stream
- vsprintf() — Print Argument Data to Buffer
- <stdarg.h>

Parent topic:

→ Library Functions