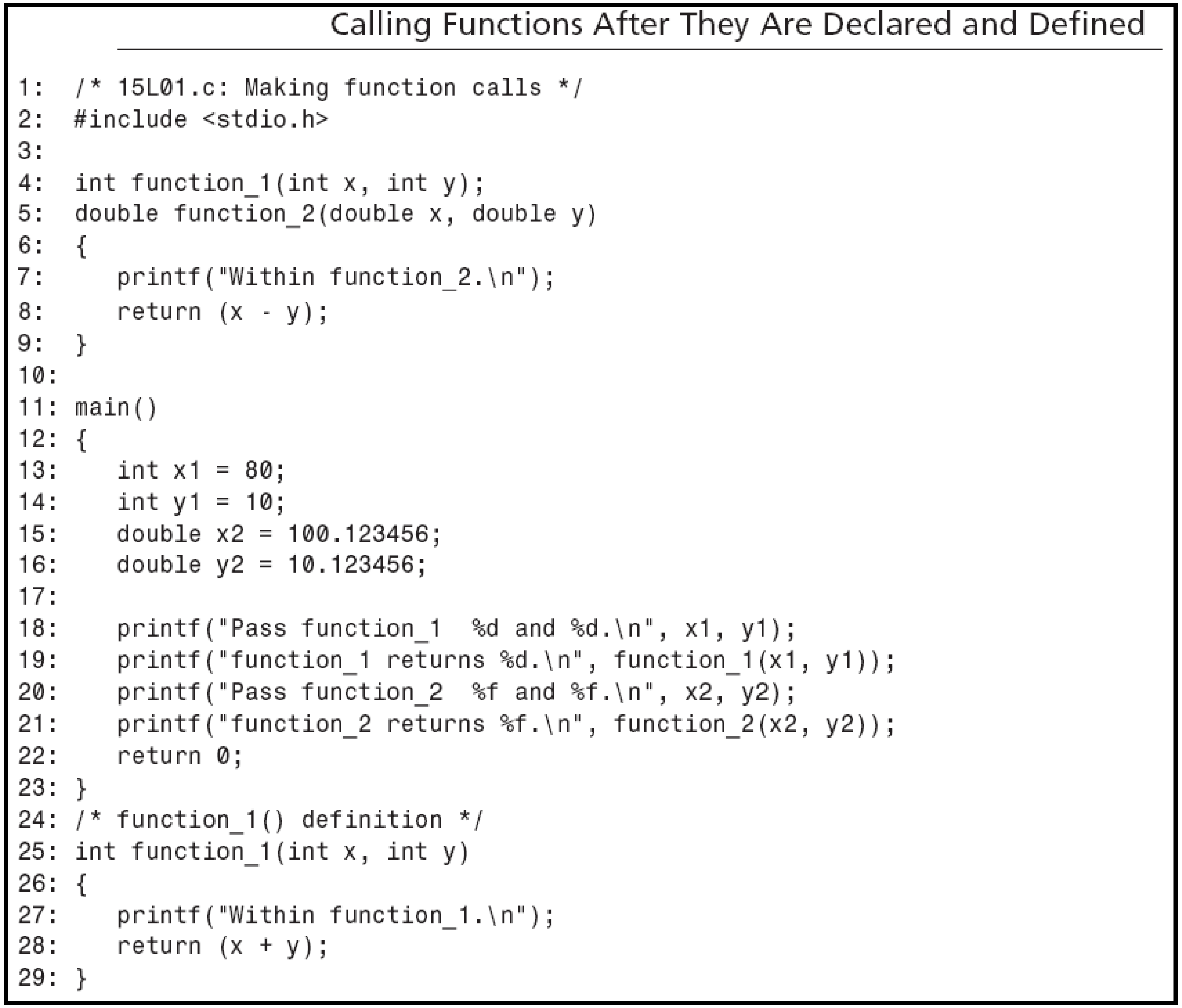
Introduction to Scientific and Engineering Computation (BIL 104E)

Lab 7

Functions : Making Function Calls

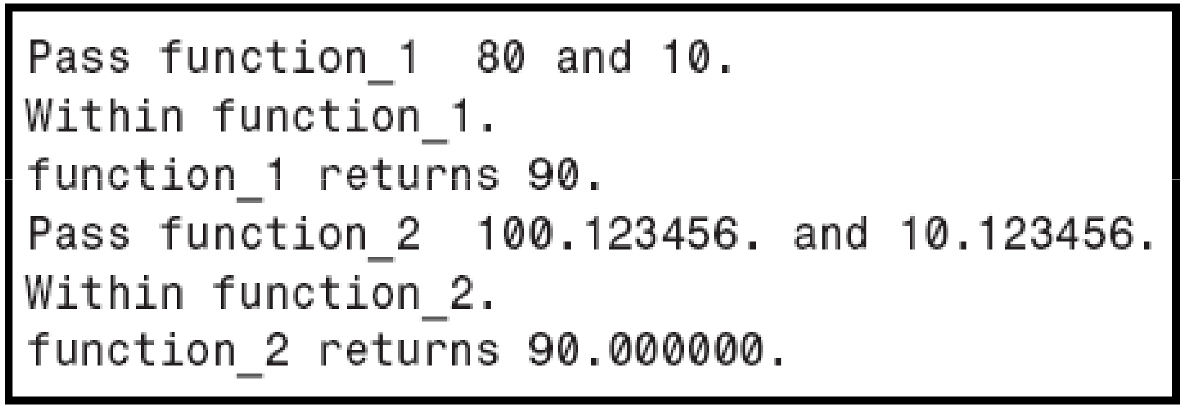


Calling Functions After They Are Declared and Defined

```c
/* 15L01.c: Making function calls */
#include <stdio.h>

int function_1(int x, int y);
double function_2(double x, double y)
{
   printf("Within function_2.\n");
   return (x - y);
}

main()
{
   int x1 = 80;
   int y1 = 10;
   double x2 = 100.123456;
   double y2 = 10.123456;

   printf("Pass function_1  %d and %d.\n", x1, y1);
   printf("function_1 returns %d.\n", function_1(x1, y1));
   printf("Pass function_2  %f and %f.\n", x2, y2);
   printf("function_2 returns %f.\n", function_2(x2, y2));
   return 0;
}
/* function_1() definition */
int function_1(int x, int y)
{
   printf("Within function_1.\n");
   return (x + y);
}
```

Functions : Making Function Calls

Computer Screen



```
Pass function_1  80 and 10.
Within function_1.
function_1 returns 90.
Pass function_2  100.123456. and 10.123456.
Within function_2.
function_2 returns 90.000000.
```

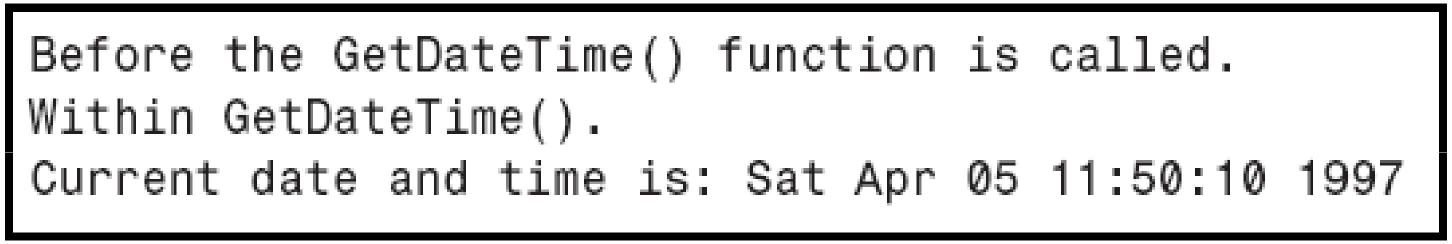
Functions with No Arguments

Using void in Function Declarations

```c
/* 15L02.c: Functions with no arguments */
#include <stdio.h>
#include <time.h>

void GetDateTime(void);

main()
{
   printf("Before the GetDateTime() function is called.\n");
   GetDateTime();
   printf("After the GetDateTime() function is called.\n");
   return 0;
}
/* GetDateTime() definition */
void GetDateTime(void)
{
   time_t now;

   printf("Within GetDateTime().\n");
   time(&now);
   printf("Current date and time is: %s\n",
      asctime(localtime(&now)));
}
```

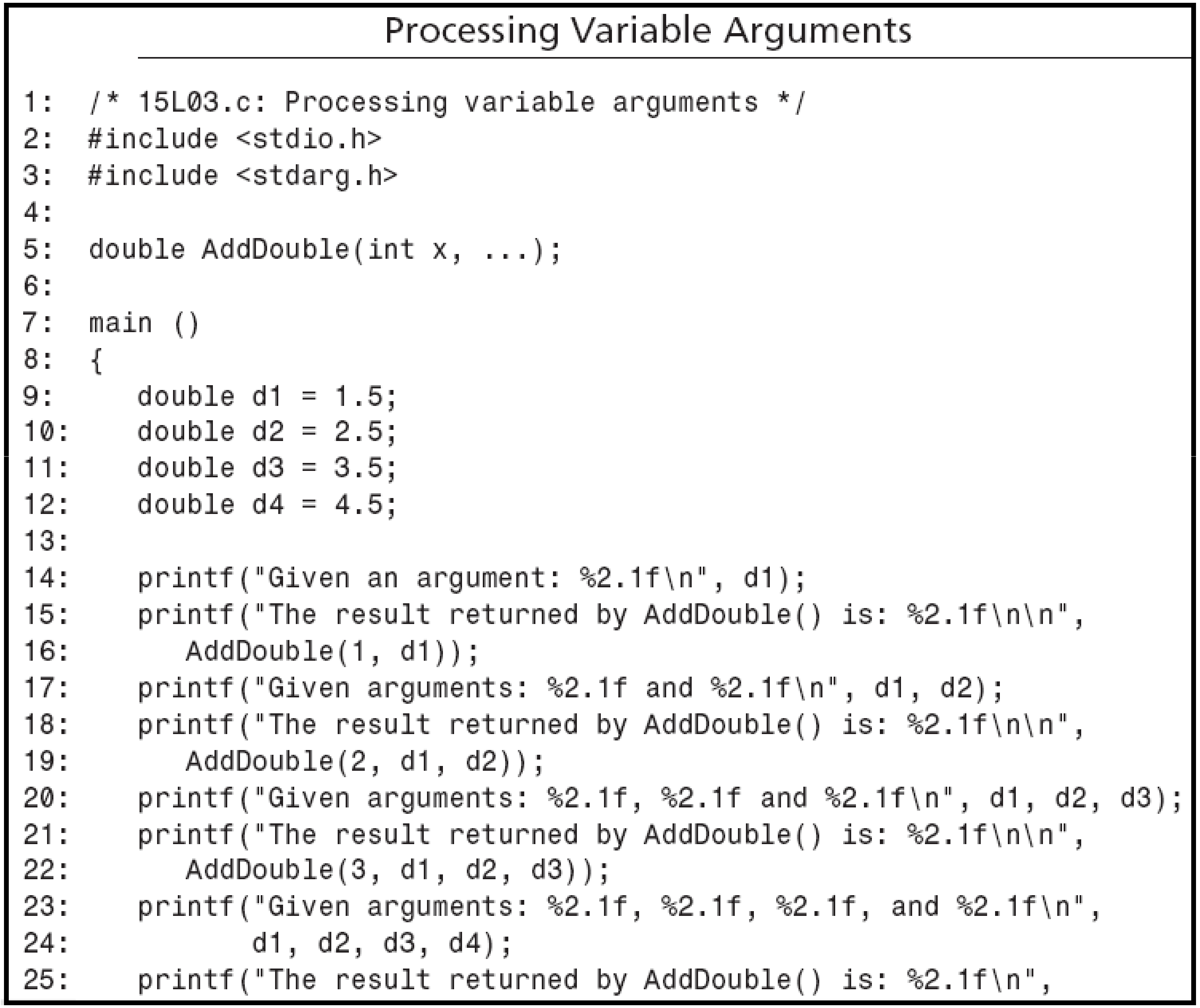
Functions with No Arguments

Computer Screen



```
Before the GetDateTime() function is called.
Within GetDateTime().
Current date and time is: Sat Apr 05 11:50:10 1997
```

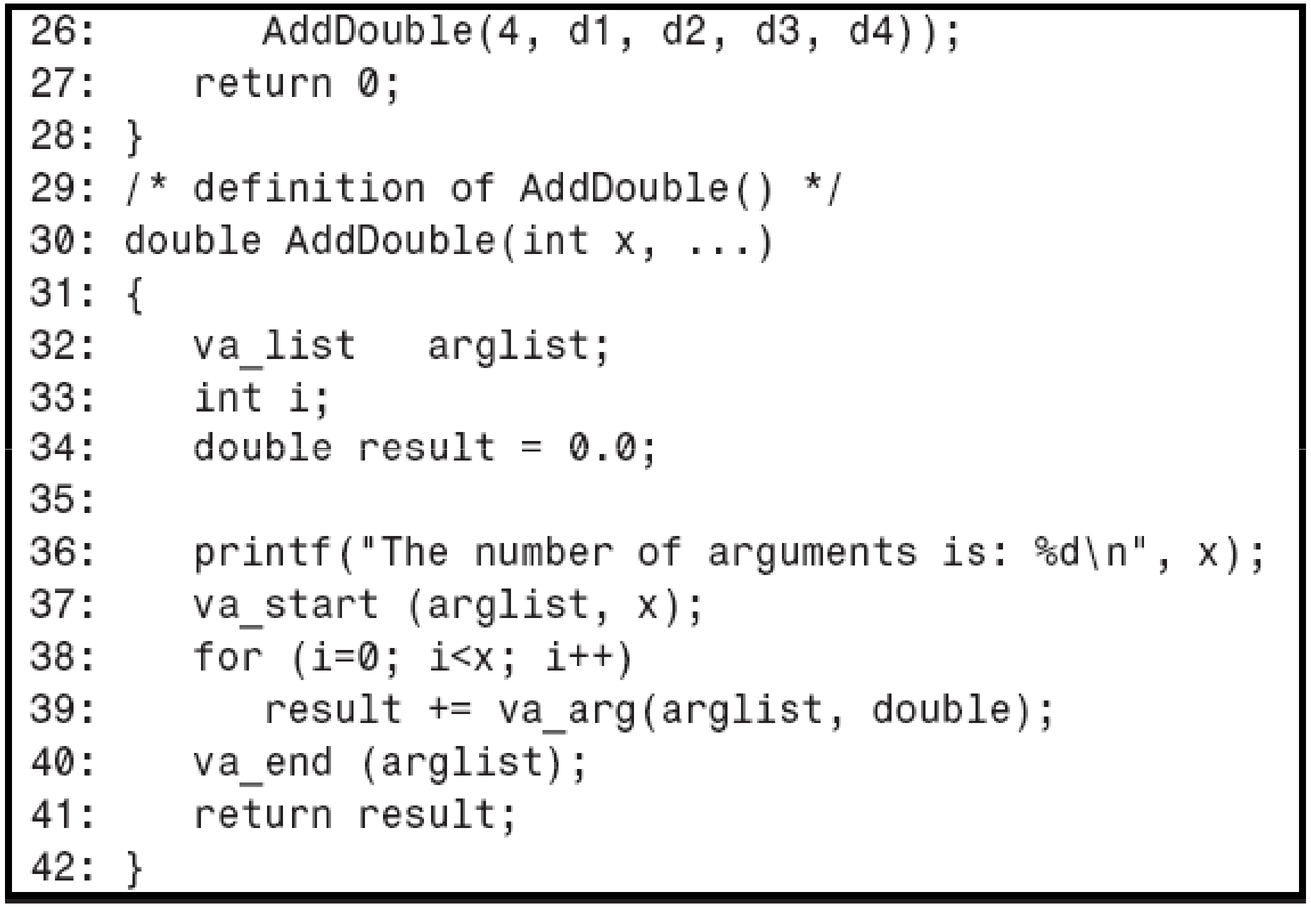
Processing Variable Arguments



Processing Variable Arguments

```c
/* 15L03.c: Processing variable arguments */
#include <stdio.h>
#include <stdarg.h>

double AddDouble(int x, ...);

main ()
{
   double d1 = 1.5;
   double d2 = 2.5;
   double d3 = 3.5;
   double d4 = 4.5;

   printf("Given an argument: %2.1f\n", d1);
   printf("The result returned by AddDouble() is: %2.1f\n\n",
      AddDouble(1, d1));
   printf("Given arguments: %2.1f and %2.1f\n", d1, d2);
   printf("The result returned by AddDouble() is: %2.1f\n\n",
      AddDouble(2, d1, d2));
   printf("Given arguments: %2.1f, %2.1f and %2.1f\n", d1, d2, d3);
   printf("The result returned by AddDouble() is: %2.1f\n\n",
      AddDouble(3, d1, d2, d3));
   printf("Given arguments: %2.1f, %2.1f, %2.1f, and %2.1f\n",
          d1, d2, d3, d4);
   printf("The result returned by AddDouble() is: %2.1f\n",
```

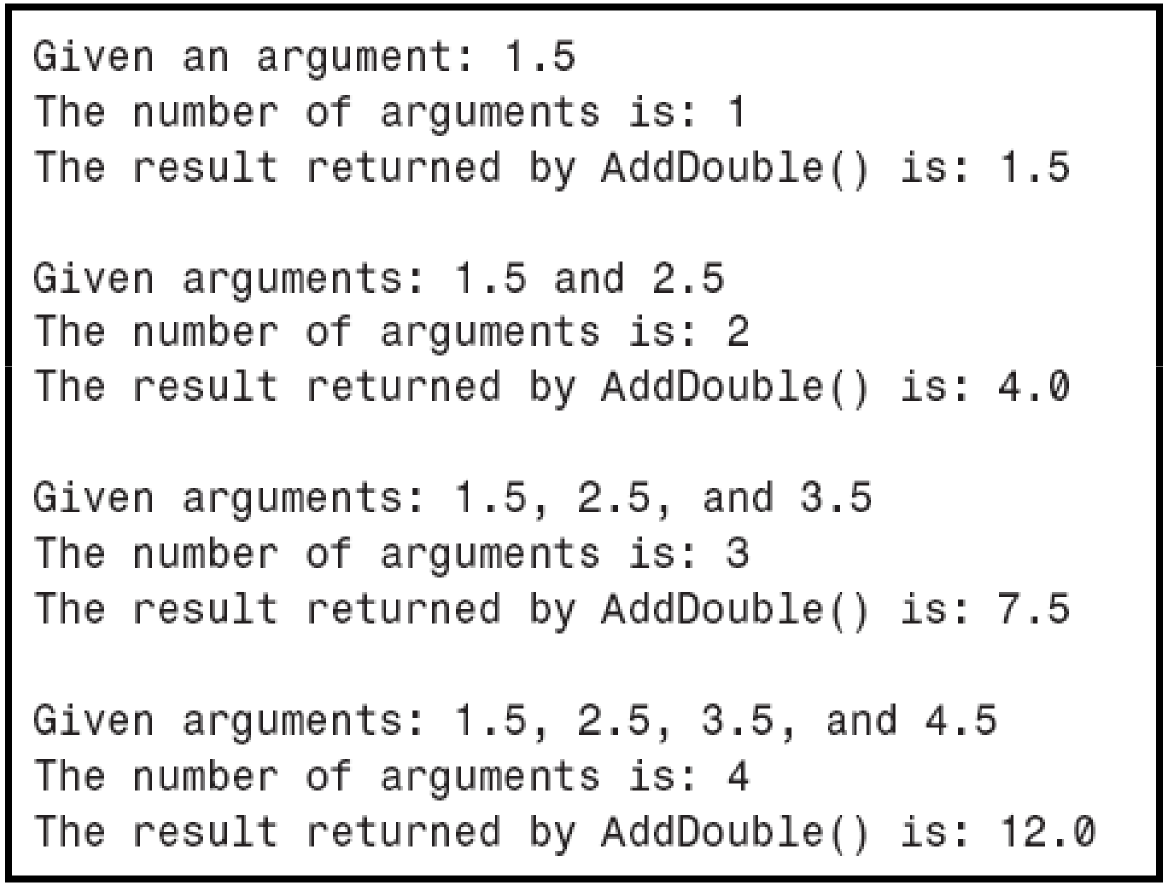
Processing Variable Arguments



```
      AddDouble(4, d1, d2, d3, d4));
   return 0;
}
/* definition of AddDouble() */
double AddDouble(int x, ...)
{
   va_list   arglist;
   int i;
   double result = 0.0;

   printf("The number of arguments is: %d\n", x);
   va_start (arglist, x);
   for (i=0; i<x; i++)
      result += va_arg(arglist, double);
   va_end (arglist);
   return result;
}
```

Processing Variable Arguments

Computer Screen



```
Given an argument: 1.5
The number of arguments is: 1
The result returned by AddDouble() is: 1.5

Given arguments: 1.5 and 2.5
The number of arguments is: 2
The result returned by AddDouble() is: 4.0

Given arguments: 1.5, 2.5, and 3.5
The number of arguments is: 3
The result returned by AddDouble() is: 7.5

Given arguments: 1.5, 2.5, 3.5, and 4.5
The number of arguments is: 4
The result returned by AddDouble() is: 12.0
```