In C programming, there are two primary methods for passing arguments to functions: call by value and call by reference. Understanding the distinction between these two methods is crucial for writing efficient and reliable code.

**Call by value:** In call by value, a copy of the actual argument's value is passed to the function's formal parameter. This means that any modifications made to the formal parameter within the function's scope do not affect the original value of the actual argument.

**Example:**

C

void swap(int x, int y) {  
 int temp = x;  
 x = y;  
 y = temp;  
}  
  
int main() {  
 int a = 5, b = 10;  
 swap(a, b); // a remains 5, b remains 10  
 printf("a: %d, b: %d\n", a, b);  
 return 0;  
}

**Call by reference:** In call by reference, the address of the actual argument is passed to the function's formal parameter. This means that any modifications made to the formal parameter within the function's scope directly affect the original value of the actual argument.

**Example:**

C

void swap(int \*x, int \*y) {  
 int temp = \*x;  
 \*x = \*y;  
 \*y = temp;  
}  
  
int main() {  
 int a = 5, b = 10;  
 swap(&a, &b); // a becomes 10, b becomes 5  
 printf("a: %d, b: %d\n", a, b);  
 return 0;  
}

In summary, call by value is used when the function should not modify the original values of the arguments, while call by reference is used when the function needs to modify the original values of the arguments.