

Call by Reference

23-11-25

Aim – Write two programs to demonstrate call by value and call by reference respectively

Theory –

Call by value passes copies of variables to functions, so original data remains unchanged.

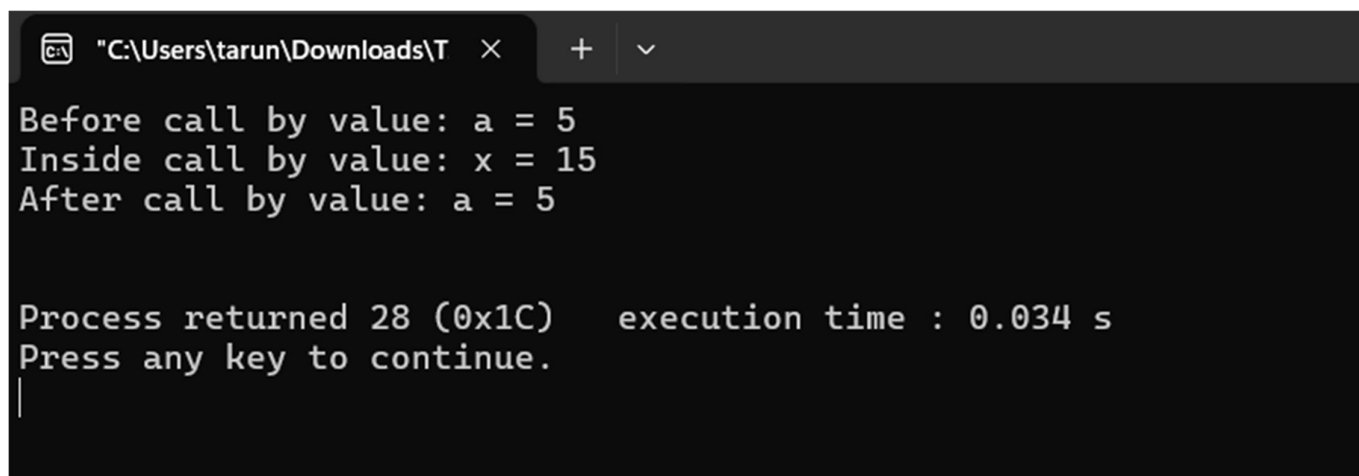
Call by reference passes addresses, allowing functions to modify actual variables directly in memory.

A1.

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

```
void cbv(int x) {  
    x = x + 10;  
    printf("Inside call by value: x = %d\n", x);  
}
```

```
void main() {  
    int a = 5;  
  
    printf("Before call by value: a = %d\n", a);  
    cbv(a);  
    printf("After call by value: a = %d\n\n", a);  
}
```



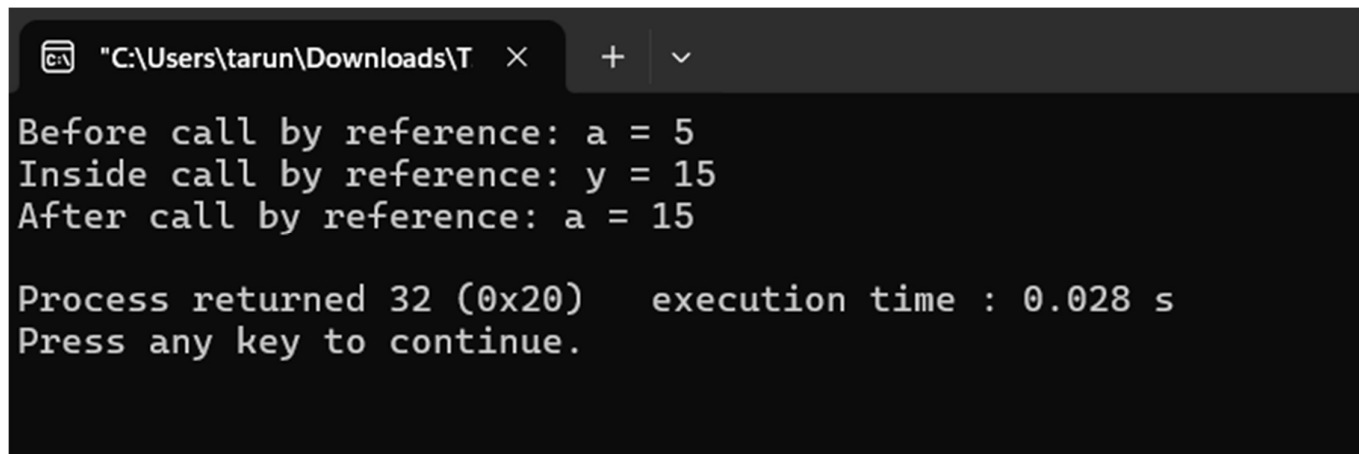
```
"C:\Users\tarun\Downloads\T" × + ∨  
Before call by value: a = 5  
Inside call by value: x = 15  
After call by value: a = 5  
  
Process returned 28 (0x1C)    execution time : 0.034 s  
Press any key to continue.  
|
```

A2.

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

```
void cbr(int *y) {  
    *y = *y + 10;
```

```
    printf("Inside call by reference: y = %d\n", *y);  
}  
  
void main() {  
    int a = 5;  
    printf("Before call by reference: a = %d\n", a);  
    cbr(&a);  
    printf("After call by reference: a = %d\n", a);  
}
```



```
"C:\Users\tarun\Downloads\T" × + ∨  
Before call by reference: a = 5  
Inside call by reference: y = 15  
After call by reference: a = 15  
  
Process returned 32 (0x20)    execution time : 0.028 s  
Press any key to continue.
```

Conclusion

This program uses call by value and call by reference in order to not change and change the values in the original memory location inside the functions.