Fundamental Coding with C

Functions 2:Call Methods

Fundamental
Coding with C

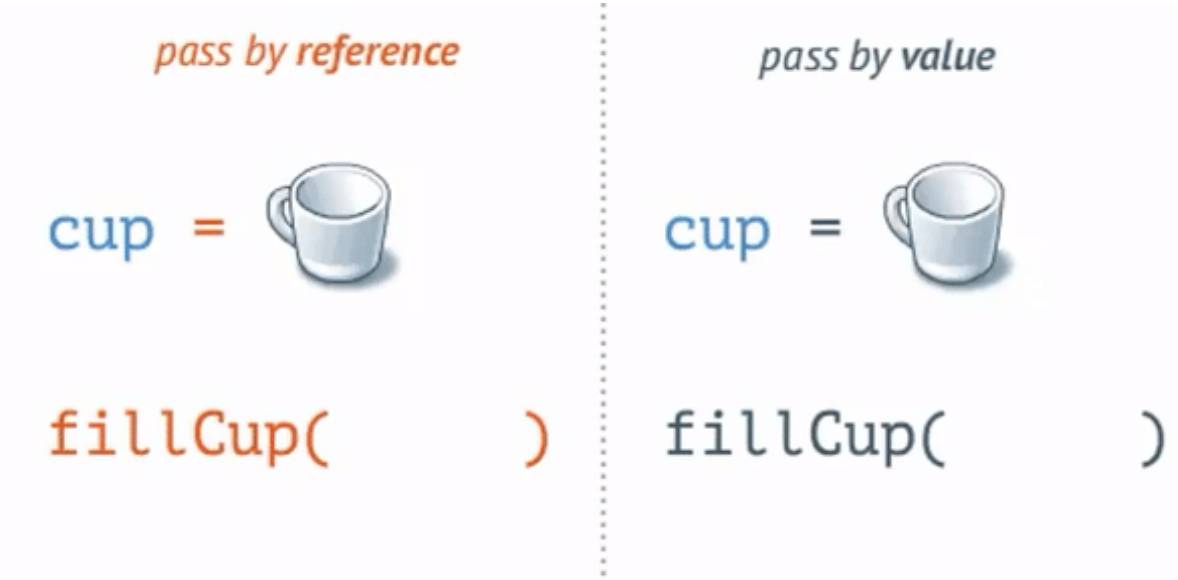
Functions 2:Call Methods

- When you define a function, you give a specification of what the function has to do. To use a function, you must call it up to perform the defined task.
- When a program calls a function, the control is transferred to the function. The function performs the defined task and terminates the program control back to the main program.
- To call a function, you need to pass the required parameters along with the function name, and if the function returns a value, then you can store the returned value.

Fundamental
Coding with C

Functions 2:Call Methods

- There are basically 2 methods to call a function:
- Call by value: The parameters get copied. Any changes in the values of the parameters only affect in the function.
- Call by reference: The changes in the parameters inside the functions, it would be reflected in the rest of the program.



Fundamental
Coding with C

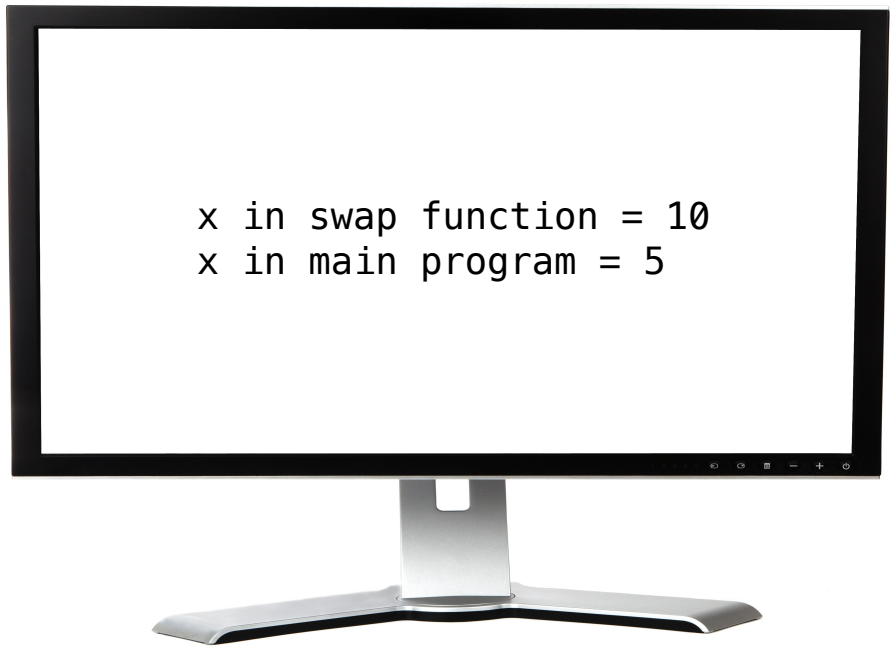
Functions 2:Call Methods

- Example call by value in C

```c
#include <stdio.h>

void swap(int x, int y) {

    int temp;
    temp = x;
    x = y;
    y = temp;
    printf("x in swap function = %d \n",x);
    return;
}

int main(void) {

    int x = 5;
    int y = 10;
    swap(x,y);
    printf("x in main program = %d \n",x);
    return 0;
}
```

Changes in x only take effect within the function.

Shows the original value of x.



It’s time to try

https://repl.it/languages/c

Fundamental
Coding with C

```c
#include <stdio.h>

void swap(int x, int y) {

    int temp;
    temp = x;
    x = y;
    y = temp;
    printf("x in swap function = %d \n",x);
    return;
}

int main(void) {

    int x = 5;
    int y = 10;
    swap(x,y);
    printf("x in main program = %d \n",x);
    return 0;
}
```

Fundamental
Coding with C

Functions 2:Call Methods

- The call by reference method of passing arguments to a function copies the address of an argument into the parameter.
- Inside the function, the address is used to access the actual argument used in the call.
- It means the changes made to the parameter affect the passed argument.

C Declaration function by reference:

```
void swap(int *x, int *y)
{
  …
}
```

* means that the parameters are by reference

C call by reference:

```
int x = 5;
int y = 10;

swap(&x,&y);
```

& indicates call to function is by reference

Fundamental
Coding with C

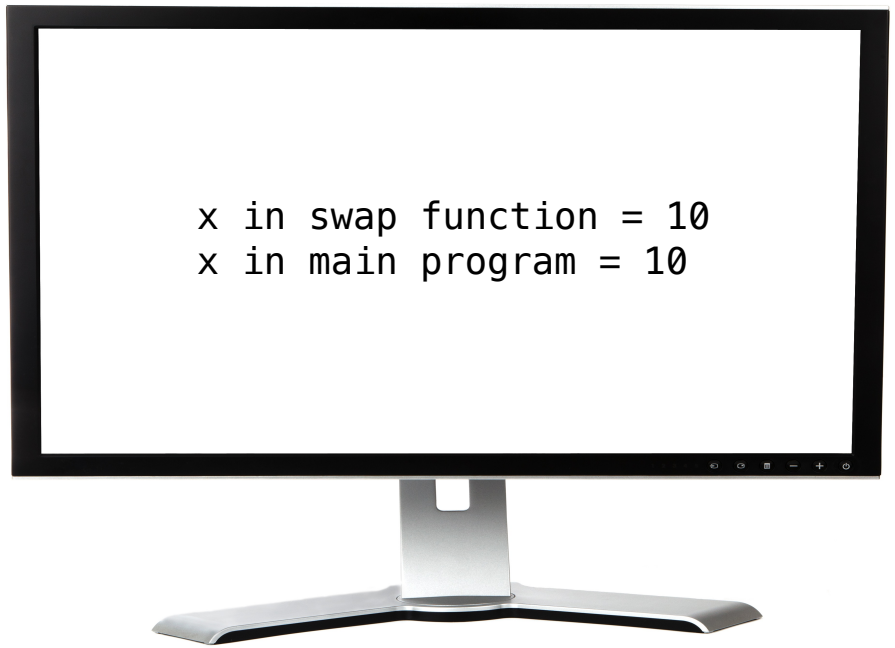
Functions 2:Call Methods

- Example call by value in C

```c
#include <stdio.h>

void swap(int *x, int *y) {
     int temp;
     temp = *x;
     *x = *y;
     *y = temp;
     printf("x in swap function = %d \n",*x);
     return;
}

int main(void) {
     int x = 5;
     int y = 10;
     swap(&x,&y);
     printf("x in main program = %d \n",x);
     return 0;
}
```

Changes in x take effect in all the program. Variables by reference with * in front

Shows the value of x changed by the swap function



It’s time to try

https://repl.it/languages/c

Fundamental
Coding with C

```c
#include <stdio.h>

void swap(int *x, int *y) {

     int temp;
     temp = *x;
     *x = *y;
     *y = temp;
     printf("x in swap function = %d \n",*x);
     return;
}

int main(void) {
     int x = 5;
     int y = 10;
     swap(&x,&y);
     printf("x in main program = %d \n",x);
     return 0;
}
```