# Function Pointers in C/C++



Posted by Alejandro G. Carlstein Ramos Mejia on October 20, 2010 February 9, 2012 About Programming / ANSI/POSIX C / C++

In the previous post, we cover how to work pointers with arrays. In this post we will see how to use function pointers, and how useful they can be:

I am assuming that you read the previous postings about pointers and pointers with arrays.

As we may recall, when we declare a pointer, the pointer will store an address of a position in memory.
Normally, we wish to create pointers of the same kind as the object we want to point at. For example, if the variable is an integer then we want the pointer to be an integer:

```
int i_variable = 22;
int* pi_variable = &i_variable;
```

If you create a different variable of the same kind, you could change where the pointer is pointing at:

```
int i_variable = 22;
int i_variable_2 = 44;
int* pi_variable = &i_variable;
printf('Value pointed at: %d \n', *pi_variable);     /* This line print 22 */
pi_variable = &i_variable_2;                         /* pi_variable points at
i_variable_2 */
printf('New Value pointed at: %d \n', *pi_variable); /* This line prints 44 */
```

Now the question comes, can we use this with functions? Yes, we can!

Here is an example of how this works:

```
/* This return the addition of value_a with value_b */
int add(int value_a, int value_b){
 return value_a + value_b;
}

/* This function return the subtraction of value_b from value_a */
int sub(int value_a, int value_b){
  return value_a - value_b;
}

int main(int argc, char* argv[]){
  int val_a = 4;
  int val_b = 5;

  /* Function pointer  must have the same return type and parameter type */
  int (*p_function)(int, int);

  p_function = add;
  printf('ADD A: %d with B:%d to obtain %d \n',
         val_a,
         val_b,
         (*p_function)(val_a, val_b));

  p_function = sub;
  printf('SUBTRACT B: %d OF A:%d to obtain %d \n',
         val_b,
         val_a,
         (*p_function)(val_a, val_b));

return 0;
}
```

This will print:

```
ADD A: 4 with B:5 to obtain 9
SUBTRACT B: 5 OF A:4 to obtain -1
```

As you can see this can be a very powerful feature. The function pointer will point to the address of any function we want to point at while the function have the same return type (int in this case), the same number of parameters (in this case, we have two parameters), and the same type of parameters (both parameters are int).