

Funciones Personalizadas

1. Declaración de una Función.

Una función personalizada tiene el siguiente formato:

```
tipo_de_dato Nombre_Función(parámetros){  
  
    return valor_de_retorno  
}
```

En donde:

- **Tipo_de_Dato:** Corresponde al tipo de dato que va a devolver la función.
- **Nombre_Función:** Es el nombre que tendrá la función, la cual debe respetar las reglas de escritura para un identificador.
- **Parámetros:** Son los valores que se proporcionarán a la función y que serán recibidas en variables. Si la función no requiere de parámetros, se pueden omitir.
- **Return:** Es la sentencia que señala el momento y valor que retornará la función.
- **Valor_de_Returno:** Es el valor que la función devolverá y que debe ser coherente con el tipo de dato definido para la función.

Las funciones pueden estar escritas antes de la función `main()`, pero si son escritas después de esta, deberán anotarse los prototipos de cada una antes de la función `main()`.

El prototipo de una función corresponde al encabezado de la misma. Por ejemplo:

```
01 #include <iostream>  
02 using namespace std;  
03  
04 int NumeroMayor(int, int);           // Prototipo de la Función NumeroMayor  
05  
06 int main(){  
07  
08 }  
09  
10 int NumeroMayor(int Num1, int Num2){  
11     if(Num1 > Num2)  
12         return Num1;  
13     else  
14         return Num2;  
15 }
```

2. Paso de Parámetros.

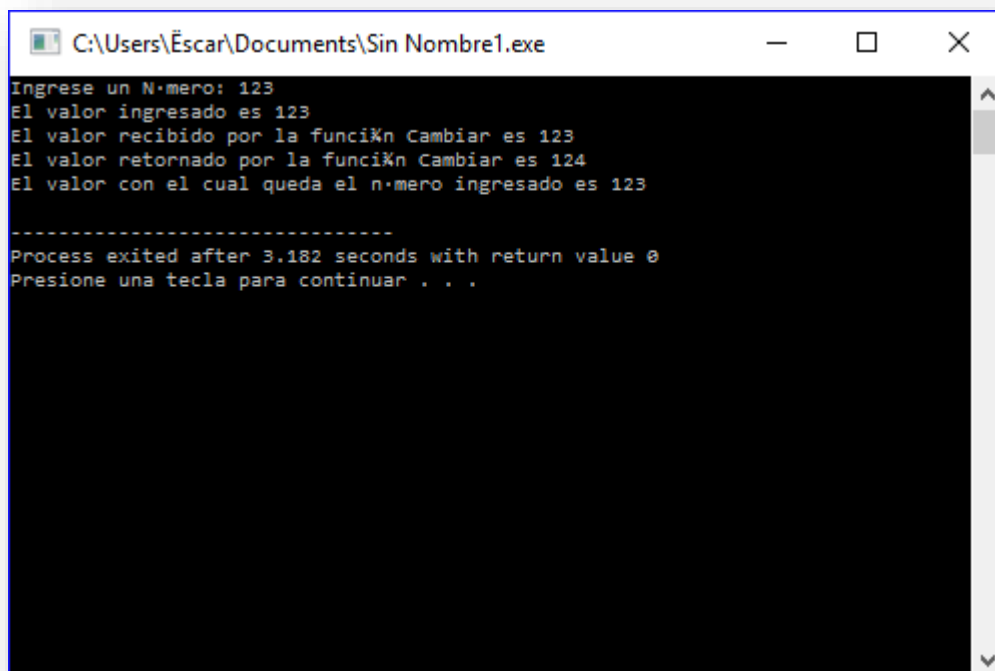
2.1. Parámetros de tipo Valor.

Un parámetro tipo valor es aquel en donde al invocar a la función, se pasa un valor el que es recibido por la función y alojado en una variable. Por ejemplo:

```
01 #include <iostream>
02 using namespace std;
03
04 int Cambiar(int Num);
05
06 int main(){
07     int Numero;
08     cout << "Ingrese un Número: ";
09     cin >> Numero;
10     cout << "El valor ingresado es " << Numero << endl;
11     cout << "El valor retornado por la función Cambiar es " << Cambiar(Numero) << endl;
12     cout << "El valor con el cual queda el número ingresado es " << Numero << endl;
13     return 0;
14 }
15
16 int Cambiar(int Num){
17     cout << "El valor recibido por la función Cambiar es " << Num << endl;
18     Num++;
19     return Num;
20 }
```

En el ejemplo anterior, la alteración del valor contenido en la variable (parámetro) **Num** (línea 18) no se verá reflejado en la variable **Numero** (la modificación queda dentro de la función **Cambiar**).

Si el valor ingresado es 123, el resultado a visualizar es el siguiente:



```
C:\Users\Escar\Documents\Sin Nombre1.exe
Ingrese un Número: 123
El valor ingresado es 123
El valor recibido por la función Cambiar es 123
El valor retornado por la función Cambiar es 124
El valor con el cual queda el número ingresado es 123

-----
Process exited after 3.182 seconds with return value 0
Presione una tecla para continuar . . .
```

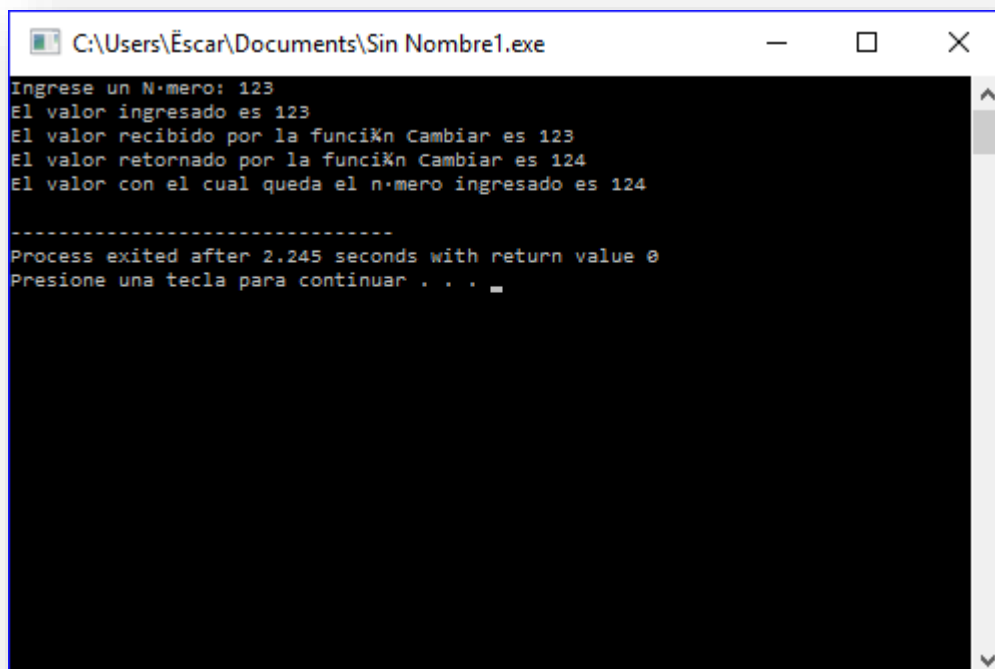
2.2. Parámetros de tipo Referencia.

Una referencia corresponde a una dirección de memoria en donde existe un valor almacenado. Cuando las referencias se ocupan en el paso de parámetros, lo que se entrega no es el valor de la variable como sucede en el paso de parámetros valor, sino que la dirección de memoria de la variable que es pasada como parámetro. Para ello, se debe anteponer el signo ampersand (&) a la variable en la declaración de la función.

```
01 #include <iostream>
02 using namespace std;
03
04 int Cambiar(int &Num);
05
06 int main(){
07     int Numero;
08     cout << "Ingrese un Número: ";
09     cin >> Numero;
10     cout << "El valor ingresado es " << Numero << endl;
11     cout << "El valor retornado por la función Cambiar es " << Cambiar(Numero) << endl;
12     cout << "El valor con el cual queda el número ingresado es " << Numero << endl;
13     return 0;
14 }
15
16 int Cambiar(int &Num){
17     cout << "El valor recibido por la función Cambiar es " << Num << endl;
18     Num++;
19     return Num;
20 }
```

En el programa anterior, en la línea 11, se invoca a la función **Cambiar** pasando como parámetro el valor ingresado por teclado y almacenada en la variable (**Numero**). Sin embargo, en la línea 16, el parámetro es recibido en la variable **Num** la cual almacenará una referencia a la variable **Numero**, debido a que **Num** tiene antepuesto un ampersand.

Si el valor ingresado es 123, el resultado a visualizar es el siguiente:



```
C:\Users\Escar\Documents\Sin Nombre1.exe
Ingrese un N.mero: 123
El valor ingresado es 123
El valor recibido por la funci n Cambiar es 123
El valor retornado por la funci n Cambiar es 124
El valor con el cual queda el n.mero ingresado es 124

-----
Process exited after 2.245 seconds with return value 0
Presione una tecla para continuar . . .
```

Si el parámetro a pasar es un arreglo, se procederá de la siguiente manera:

```
01 #include <iostream>
02 #define TOPE 5
03
04 using namespace std;
05
06 void Ordenar(int Vector[]);
07
08 int main(){
09     int Punt, Arreglo[TOPE];
10     for(Punt = 0; Punt < TOPE; Punt++){
11         cout << "Ingrese un Número: ";
12         cin >> Arreglo[Punt];
13     }
14
15     Ordenar(Arreglo);
16
17     cout << "Los valores ordenados son:" << endl;
18     for(Punt = 0; Punt < TOPE; Punt++)
19         cout << Arreglo[Punt] << endl;
20
21     return 0;
22 }
23
24 void Ordenar(int Vector[]){
25     int Punt1, Punt2, Aux;
26     for(Punt1 = 0; Punt1 < TOPE - 1; Punt1++){
27         for(Punt2 = Punt1 + 1; Punt2 < TOPE; Punt2++){
28             if(Vector[Punt1] > Vector[Punt2]){
29                 Aux = Vector[Punt1];
30                 Vector[Punt1] = Vector[Punt2];
31                 Vector[Punt2] = Aux;
32             }
33 }
```

Observe en el ejercicio anterior la línea 24, no se ha antepuesto el símbolo ampersand a Vector, debido a que el nombre de un arreglo es por sí un puntero.