En el apunte anterior de funciones hemos visto cómo trabajar con ellas con variables simples. En el siguiente texto veremos algunos ejemplos de cómo trabajar con funciones y vectores.

Veamos dos declaraciones de funciones más:

```
void mostrarVector(float vec[10]);
```

```
void mostrarVector(float vec[], int tamanio);
```

Ambas funciones no devuelven nada. Eso ya lo hemos aprendido en el ejemplo anterior. Los nombres nos indican que mostrarán por pantalla información. Pero a diferencia de las otras, estas funciones reciben parámetros en el que se visualizan corchetes. Esto se debe a que estos parámetros son un vector de 10 elementos para la primera función y un vector de una indeterminada cantidad de elementos y un entero con el tamaño del vector para la segunda función.

La primera función está pensada para mostrar un vector de 10 elementos del tipo de dato float. Solamente puede recibir un vector de ese tipo y ese tamaño, por lo que a fines prácticos, es más difícil de reutilizar. Veamos su definición:

```
void mostrarVector(float vec[10]){
   int i;
   for(i=0; i<10; i++){
      cout << vec[i] << " ";
   }
}
```

La segunda función está pensada para mostrar cualquier vector de cualquier tamaño de tipo de dato float. Para poder saber cuántos elementos tiene el vector se envía un segundo parámetro de tipo de dato entero que indica la cantidad de elementos del vector. Esta función tiene muchas más posibilidades de ser utilizada. Veamos su definición:

```
void mostrarVector(float vec[], int tamanio){
   int i;
   for(i=0; i<tamanio; i++){
      cout << vec[i] << " ";
   }
}
```

Podemos observar que una función no requiere recibir entre los corchetes el tamaño del vector. Por lo que la segunda función *mostrarVector* podría recibir un vector de float de cualquier tamaño. En la función, el vector se llamará genéricamente *vec*. Además, la función recibe el tamaño del vector como parámetro. Esto veremos será de gran utilidad porque si el objetivo es mostrar todos los elementos del vector necesitaremos saber hasta donde mostrarlo.
La función además recibe un entero con la cantidad de filas que ayudará de la misma manera que la variable *tamanio* en la función *mostrarVector*.

Supongamos que queremos llamar a estas funciones en main. Veamos un ejemplo de cómo podría ser la función main para llamar a estas funciones:

```
#include <iostream>
using namespace std;

void mostrarVector(float vec[10]);
void mostrarVector(float vec[], int tamanio);

int main()
{
   float vec1[10] = {1, 2, 3, 4, 5.5, 6.6, 7, 8, 9, 10.111};
   float vec2[4] = {1.11, 2.22, 3.33, 4.44};
   mostrarVector(vec1);
   cout << endl;
   mostrarVector(vec2, 4);
   cout << endl;
   mostrarVector(vec1, 10);
   return 0;
}
// Sigue en la otra página
```

```
void mostrarVector(float vec[10]){
   int i;
   for(i=0; i<10; i++){
      cout << vec[i] << " ";
   }
}

void mostrarVector(float vec[], int tamanio){
   int i;
   for(i=0; i<tamanio; i++){
      cout << vec[i] << " ";
   }
}
```

Podemos notar varias cosas en este ejemplo. Lo primero es que en el llamado a la funciones los parámetros que son vectores se pasan utilizando el nombre del vector. Por ejemplo, hay tres llamados a funciones y se pasan los vectores vec1 y vec2. En cada llamado se utiliza el nombre del vector que queremos enviar a la función para poder enviarlo. En el caso de los llamados `mostrarVector(vec2, 4)` y `mostrarVector(vec1, 10)` se añade un segundo parámetro que representa el tamaño del vector y que hemos explicado anteriormente su utilidad.

Otro aspecto importante que se puede observar y que es inherente a las funciones es que podemos tener varias funciones con exactamente el mismo nombre pero diferentes parámetros. Esta característica se llama Sobrecarga de funciones y se verá en mayor detalle en Programación II.

La función main si es ejecutada mostraría por pantalla la siguiente información:

```
1 2 3 4 5.5 6.6 7 8 9 10.111
1.11 2.22 3.33 4.44
1 2 3 4 5.5 6.6 7 8 9 10.111
```

Veamos dos ejemplos más:

```
void cargarVector(int vec[], int tamanio);
```

```
int buscarMaximoEnVector(int vec[], int tamanio);
```

Como su nombres hacen suponer, la función cargarVector cargará un vector de enteros de una cantidad de elementos que le indicaremos al llamarla y la función buscarMaximoEnVector recibirá un vector de enteros y su tamaño y devolverá el valor más grande dentro del vector.

Veamos sus definiciones:

```
int calcularMaximoEnVector(int vec[], int tamanio){
   int i, maximo;
   maximo = vec[0];
   for(i=1; i<tamanio; i++){
      if (vec[i] > maximo){
      maximo = vec[i];
      }
   }
   return maximo;
}

void cargarVector(int vec[], int tamanio){
   int i;
   for(i=0; i<tamanio; i++){
      cin >> vec[i];
   }
}
```

La función que calcula el máximo recibe un vector de enteros y su tamaño. Luego, define una variable de tipo entero para almacenar el máximo e indica que inicialmente el valor más grande del vector es el primer elemento. Luego, recorre del segundo elemento del vector en adelante y determina si hay un valor más grande; en caso de encontrarlo actualiza el valor del máximo. Por último, devuelve el máximo y es por eso que la función indica que la misma devuelve un int.

Por otro lado, la función cargarVector recibe un vector y su tamaño. Podríamos decir que no se diferencia en nada con respecto a mostrarVector salvo por su leve cambio en el nombre y porque hace un cin en lugar de un cout.

Algo importante a aclarar y que queda evidenciado en esta función es que si **recibimos un vector en una función y modificamos sus valores también serán modificados en la función que envió como parámetro ese vector.**

Veamos un ejemplo del programa main que utiliza ambas funciones:

```
#include <iostream>
using namespace std;

void cargarVector(int vec[], int tamanio);
int calcularMaximoEnVector(int vec[], int tamanio);

int main()
{
   int vec1[10], maximoValor;
   cargarVector(vec1, 10);
   maximoValor = calcularMaximoEnVector(vec1, 10);
   cout << endl << maximoValor;
   return 0;
}

int calcularMaximoEnVector(int vec[], int tamanio){
   int i, maximo;
   maximo = vec[0];
   for(i=1; i<tamanio; i++){
      if (vec[i] > maximo){
      maximo = vec[i];
      }
   }
   return maximo;
}

void cargarVector(int vec[], int tamanio){
   int i;
   for(i=0; i<tamanio; i++){
      cin >> vec[i];
   }
}
```

En relación a lo mencionado anteriormente, cargarVector es llamado desde main y es main quien envía como parámetro vec1 a la función. Luego, cargar vector recibe genéricamente el vector de main en una variable llamada vec y realiza para cada uno de sus valores un cin.
Luego de ejecutar dichos cin, el vector habrá sido modificado tanto en la función cargarVector como en la variable vec1 de main. Esto se debe a que los vectores son recibidos en funciones de una manera particular que se verá con más detalle en Programación II.

Por último, podemos observar que luego de cargar el vector en main se llama a la función calcularMaximoEnVector enviándole el vector previamente cargado y su tamaño. También se puede ver que como la función devuelve el máximo valor dentro del vector y este es un número entero entonces su valor de retorno es almacenado en una variable entera llamada maximoValor.
Si el usuario cargara los números del 1 al 10 dentro del vector, el programa mostraría por pantalla el número 10.