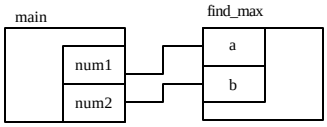
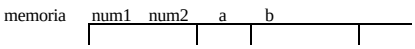
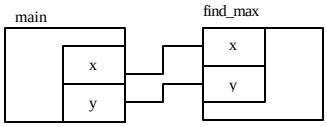
VARIABLES LOCALES

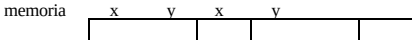
- Sólo se utiliza dentro de la función en la que fue declarada.
- Un nombre de variable puede ser declarado en más de una función (locales).
- Cada función tiene acceso sólo a las variables que declaro.

Gráficamente:





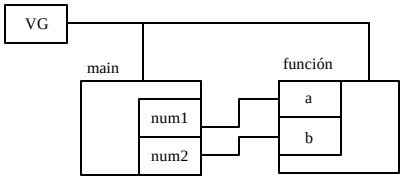




VARIABLES GLOBALES

- Se pueden compartir entre funciones.
- Se declaran fuera del cuerpo de la función.
- El valor de una variable global puede ser accesado o modificado por cualquier función.

Gráficamente:



Nota: El uso indiscriminado de variables globales es mala práctica.

MANEJO DE FUNCIONES: Si queremos que una función devuelva más de un valor:

- La función tiene que tener acceso a ciertas variables de quien la llama.
- El acceso lo obtiene conociendo el "address" en memoria de la variable.

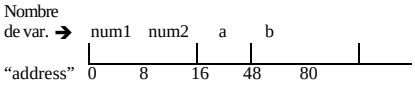
Para el envío, almacenamiento y uso de un "address" se necesitan:

1. **"Address Operator" (&)** : Se usa para conocer el "address" de una variable. Indica la dirección en memoria de una variable.

Ej: cout << "la dirección de la variable num1 es:" << &num1;

- **num1** es una variable en la memoria.
- **&num1** busca la dirección donde comienza la variable.

En memoria:



2. **"Pointers"**: variables para almacenar el "address" (el valor), tienen que ser declarados antes de usarse (tal como otras variables).

Para hacer la declaración de un "pointer" necesitamos:

3. **"Indirection Operator" (*)**: (Asterisco) Identifica un "pointer", indica la variable cuyo "address" es guardado.

Declaración de un "pointer":

- **nombre** de la variable "pointer".
- Puede **inicializarse** a la vez.
- El **tipo** de dato que se especifica es el de la variable cuyo "address" será almacenado en el "pointer".

Ejemplo: tengo una variable char y necesito declarar un pointer para ella.

```
char stop;
char *var_add;
```

Indica que var_add es un pointer.
Indica que var_add apunta a una variable tipo char.

Ej: si se quiere inicializar el "pointer" se puede:

```
char stop;
char *var_add = &stop;
```

o si ya ha sido declarada:

```
var_add = &stop;
```

Cuando a una función se le enviarán como argumentos, los "address" de variables locales, tiene que indicarse que están relacionados a "pointers".

El **Prototipo** de la función es:

```
void nuevo (float *, float *);
```

El **llamado** de la función "nuevo" sería:

```
nuevo (&variable1, &variable2);
```

La función ("**header line**") que recibe los "address" como parámetros, tiene que declarar "pointers" para recibirlos:

```
void nuevo (float *addr1, float *addr2);
```