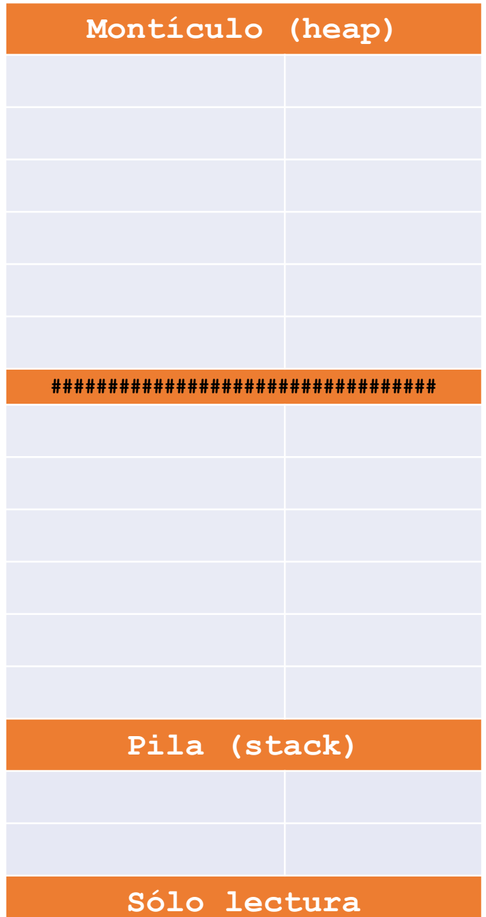
```c
#include <stdio.h>
#include <stdlib.h>

#define T 2

void f(int * input, int ** output) {
    input = (int *) malloc(T * sizeof(int));
    *output = (int *) malloc(T * sizeof(int));
    int * array = *output;

    input[0] = 10;
    input[1] = 11;

    array[0] = 100;
    array[1] = 101;
}

int main() {
    int   a1[T] = {0,1};
    int * a2;

    f(a1, &a2);

    return 0;
}
```

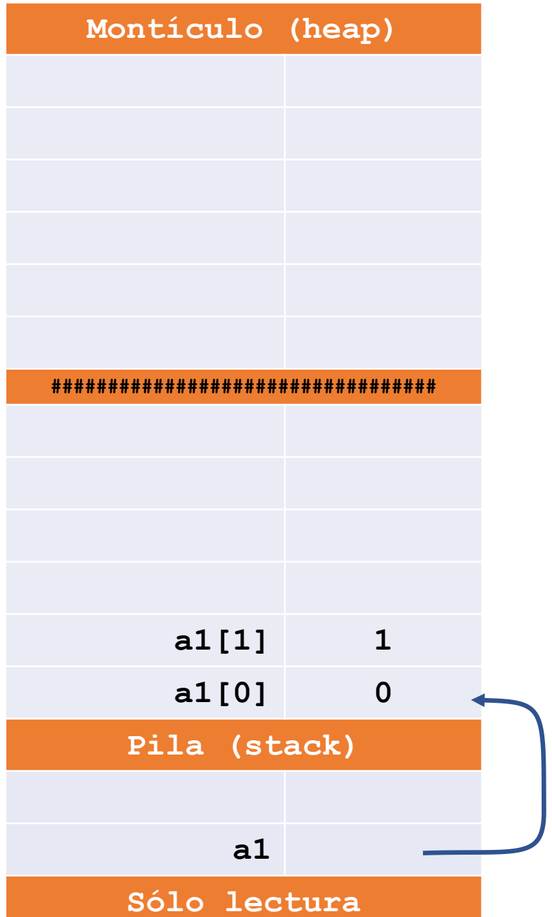
PC



1. El contador de programa (*program counter* o PC) comienza en la primera sentencia del programa. La memoria aún está vacía.

```
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3
 4  #define T 2
 5
 6  void f(int * input, int ** output) {
 7      input = (int *) malloc(T * sizeof(int));
 8      *output = (int *) malloc(T * sizeof(int));
 9      int * array = *output;
10
11      input[0] = 10;
12      input[1] = 11;
13
14      array[0] = 100;
15      array[1] = 101;
16  }
17
18  int main() {
19      int   a1[T] = {0,1};
20      int * a2;
21
22      f(a1, &a2);
23
24      return 0;
25  }
```

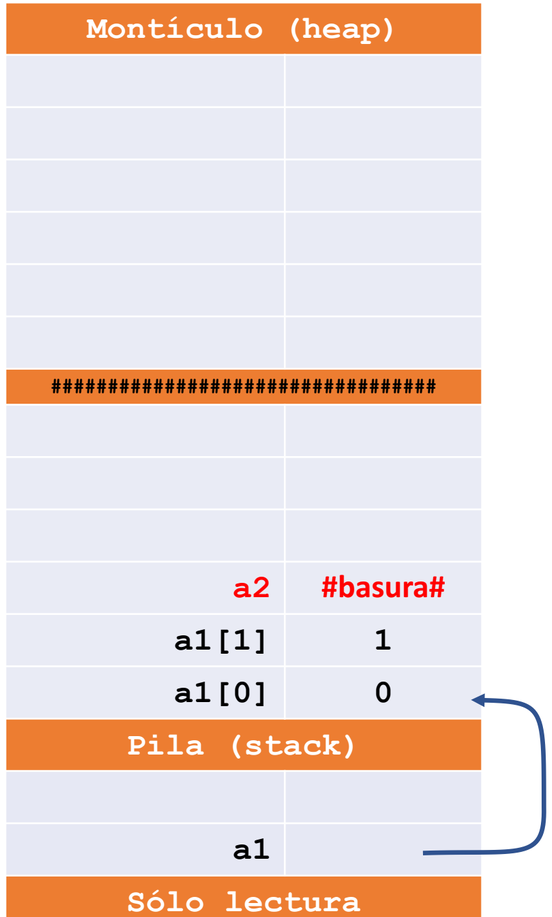
PC



1. El contador de programa (*program counter* o PC) comienza en la primera sentencia del programa. La memoria aún está vacía.
2. Tras ejecutar la línea 19, se reserva memoria: (i) en la pila para dos enteros, y (ii) en unan zona de sólo lectura para un puntero a un entero.

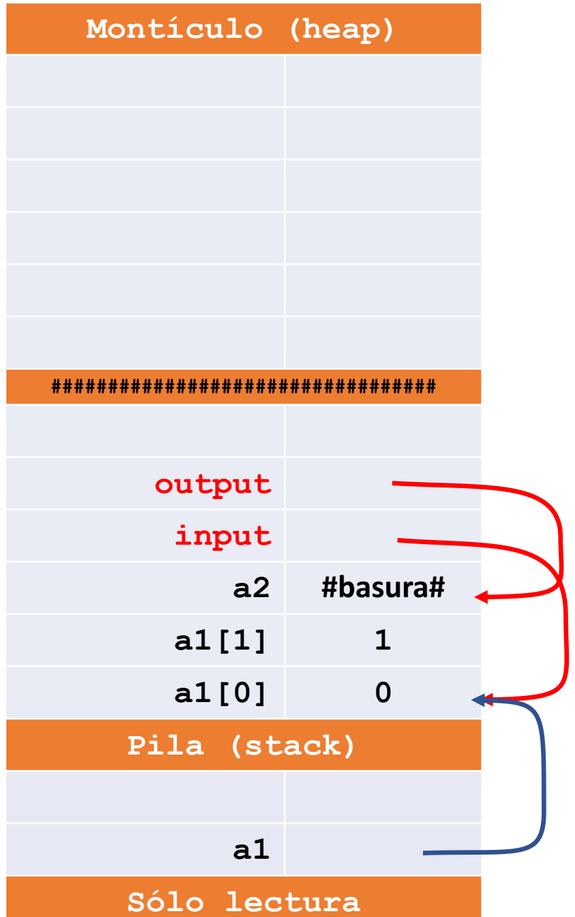
```c
#include <stdio.h>
#include <stdlib.h>

#define T 2

void f(int * input, int ** output) {
    input = (int *) malloc(T * sizeof(int));
    *output = (int *) malloc(T * sizeof(int));
    int * array = *output;

    input[0] = 10;
    input[1] = 11;

    array[0] = 100;
    array[1] = 101;
}

int main() {
    int   a1[T] = {0,1};
    int * a2;

    f(a1, &a2);

    return 0;
}
```

PC



1. El contador de programa (*program counter* o PC) comienza en la primera sentencia del programa. La memoria aún está vacía.
2. Tras ejecutar la **línea 19**, se reserva memoria: (i) en **la pila** para dos enteros, y (ii) en unan zona de **sólo lectura** para un puntero a un entero.
3. La segunda declaración (**línea 20**), reserva memoria **en la pila** para un puntero a un entero.

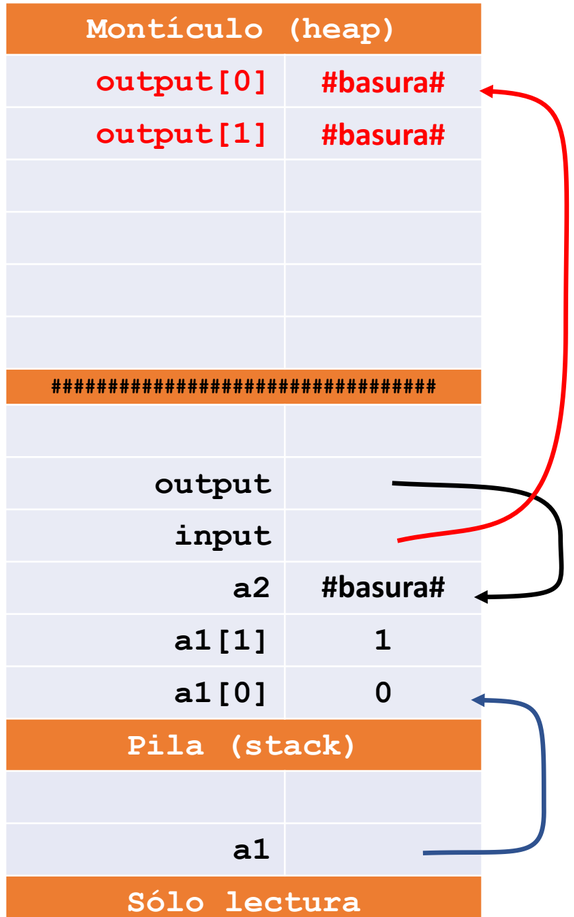
```
PC
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3
 4  #define T 2
 5
 6  void f(int * input, int ** output) {
 7      input = (int *) malloc(T * sizeof(int));
 8      *output = (int *) malloc(T * sizeof(int));
 9      int * array = *output;
10
11      input[0] = 10;
12      input[1] = 11;
13
14      array[0] = 100;
15      array[1] = 101;
16  }
17
18  int main() {
19      int   a1[T] = {0,1};
20      int * a2;
21
22      f(a1, &a2);
23
24      return 0;
25  }
```



1. El contador de programa (*program counter* o PC) comienza en la primera sentencia del programa. La memoria aún está vacía.
2. Tras ejecutar la **línea 19**, se reserva memoria: (i) en **la pila** para dos enteros, y (ii) en unan zona de **sólo lectura** para un puntero a un entero.
3. La segunda declaración (**línea 20**), reserva memoria **en la pila** para un puntero a un entero.
4. Al invocarse **f**, aparecen dos nuevas variables de tipo puntero en la pila: **input** y **output**, cuyo contenido es **a1** y la dirección de **a2**, respectivamente.

```
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3
 4  #define T 2
 5
 6  void f(int * input, int ** output) {
 7      input = (int *) malloc(T * sizeof(int));
 8      *output = (int *) malloc(T * sizeof(int));
 9      int * array = *output;
10
11      input[0] = 10;
12      input[1] = 11;
13
14      array[0] = 100;
15      array[1] = 101;
16  }
17
18  int main() {
19      int   a1[T] = {0,1};
20      int * a2;
21
22      f(a1, &a2);
23
24      return 0;
25  }
```

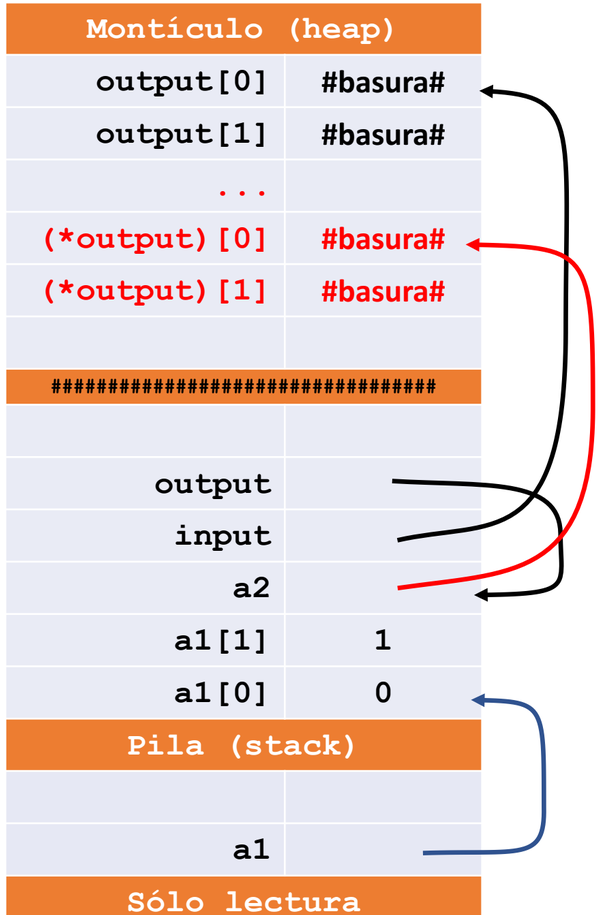
PC



1. El contador de programa (*program counter* o PC) comienza en la primera sentencia del programa. La memoria aún está vacía.
2. Tras ejecutar la **línea 19**, se reserva memoria: (i) en **la pila** para dos enteros, y (ii) en unan zona de **sólo lectura** para un puntero a un entero.
3. La segunda declaración (**línea 20**), reserva memoria **en la pila** para un puntero a un entero.
4. Al invocarse **f**, aparecen dos nuevas variables de tipo puntero en la pila: **input** y **output**, cuyo contenido es **a1** y la dirección de **a2**, respectivamente.
5. Se reserva memoria para dos enteros en el montículo, y se asigna la dirección devuelta por **malloc()** a la variable **input**.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define T 2
5
6  void f(int * input, int ** output) {
7      input = (int *) malloc(T * sizeof(int));
8      *output = (int *) malloc(T * sizeof(int));
9      int * array = *output;
10
11     input[0] = 10;
12     input[1] = 11;
13
14     array[0] = 100;
15     array[1] = 101;
16 }
17
18 int main() {
19     int   a1[T] = {0,1};
20     int * a2;
21
22     f(a1, &a2);
23
24     return 0;
25 }
```

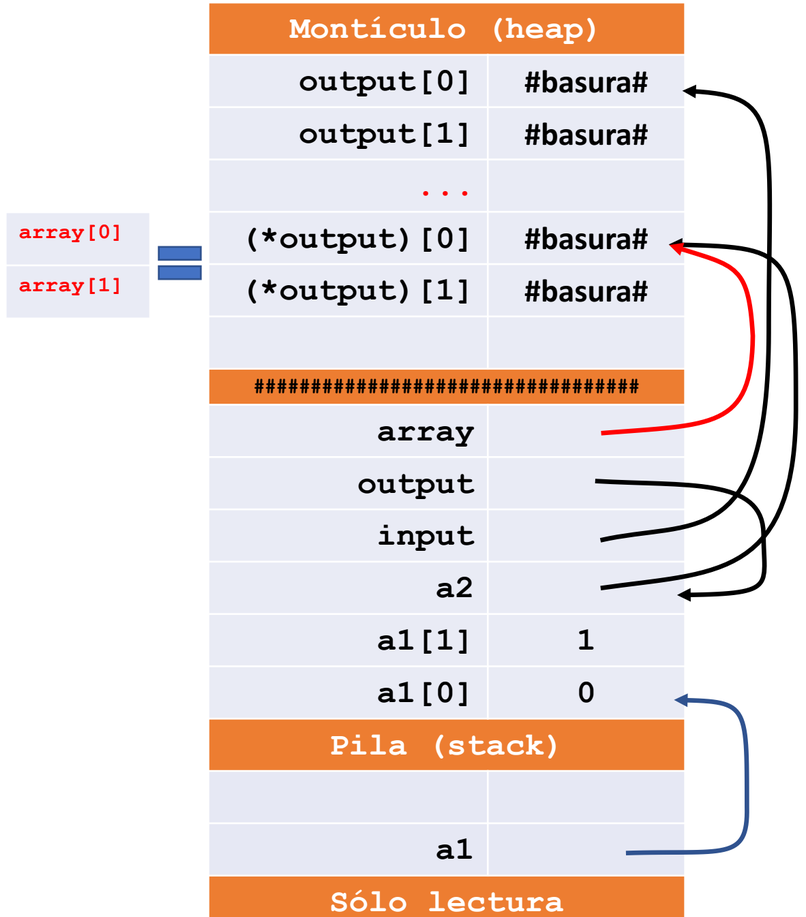
PC



1. El contador de programa (*program counter* o PC) comienza en la primera sentencia del programa. La memoria aún está vacía.
2. Tras ejecutar la **línea 19**, se reserva memoria: (i) en **la pila** para dos enteros, y (ii) en unan zona de **sólo lectura** para un puntero a un entero.
3. La segunda declaración (**línea 20**), reserva memoria **en la pila** para un puntero a un entero.
4. Al invocarse **f**, aparecen dos nuevas variables de tipo puntero en la pila: **input** y **output**, cuyo contenido es **a1** y la dirección de **a2**, respectivamente.
5. Se reserva memoria para dos enteros en el montículo, y se asigna la dirección devuelta por **malloc()** a la variable **input**.
6. Se reserva memoria nuevamente para dos enteros en el montículo, y se asigna la dirección devuelta por **malloc**() a ***output**, es decir, a un **puntero a un entero**.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define T 2
5
6  void f(int * input, int ** output) {
7      input = (int *) malloc(T * sizeof(int));
8      *output = (int *) malloc(T * sizeof(int));
9      int * array = *output;
10
11     input[0] = 10;
12     input[1] = 11;
13
14     array[0] = 100;
15     array[1] = 101;
16 }
17
18 int main() {
19     int   a1[T] = {0,1};
20     int * a2;
21
22     f(a1, &a2);
23
24     return 0;
25 }
```

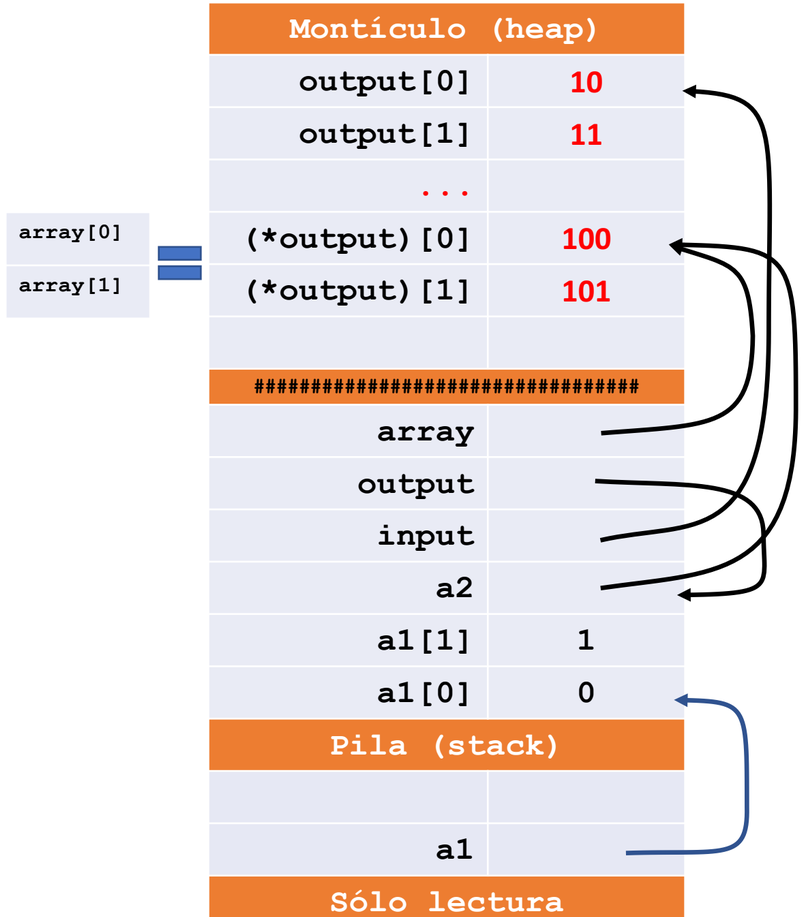
PC



1. El contador de programa (*program counter* o PC) comienza en la primera sentencia del programa. La memoria aún está vacía.
2. Tras ejecutar la **línea 19**, se reserva memoria: (i) en **la pila** para dos enteros, y (ii) en unan zona de **sólo lectura** para un puntero a un entero.
3. La segunda declaración (**línea 20**), reserva memoria **en la pila** para un puntero a un entero.
4. Al invocarse **f**, aparecen dos nuevas variables de tipo puntero en la pila: **input** y **output**, cuyo contenido es **a1** y la dirección de **a2**, respectivamente.
5. Se reserva memoria para dos enteros en el montículo, y se asigna la dirección devuelta por **malloc()** a la variable **input**.
6. Se reserva memoria nuevamente para dos enteros en el montículo, y se asigna la dirección devuelta por **malloc**() a ***output**, es decir, a un **puntero a un entero**.
7. Se crea un nuevo **int *** que apunta a la memoria reservada en la línea 8. Nótese la equivalencia.

```
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3
 4  #define T 2
 5
 6  void f(int * input, int ** output) {
 7      input = (int *) malloc(T * sizeof(int));
 8      *output = (int *) malloc(T * sizeof(int));
 9      int * array = *output;
10
11      input[0] = 10;
12      input[1] = 11;
13
14      array[0] = 100;
15      array[1] = 101;
16  }
17
18  int main() {
19      int   a1[T] = {0,1};
20      int * a2;
21
22      f(a1, &a2);
23
24      return 0;
25  }
```

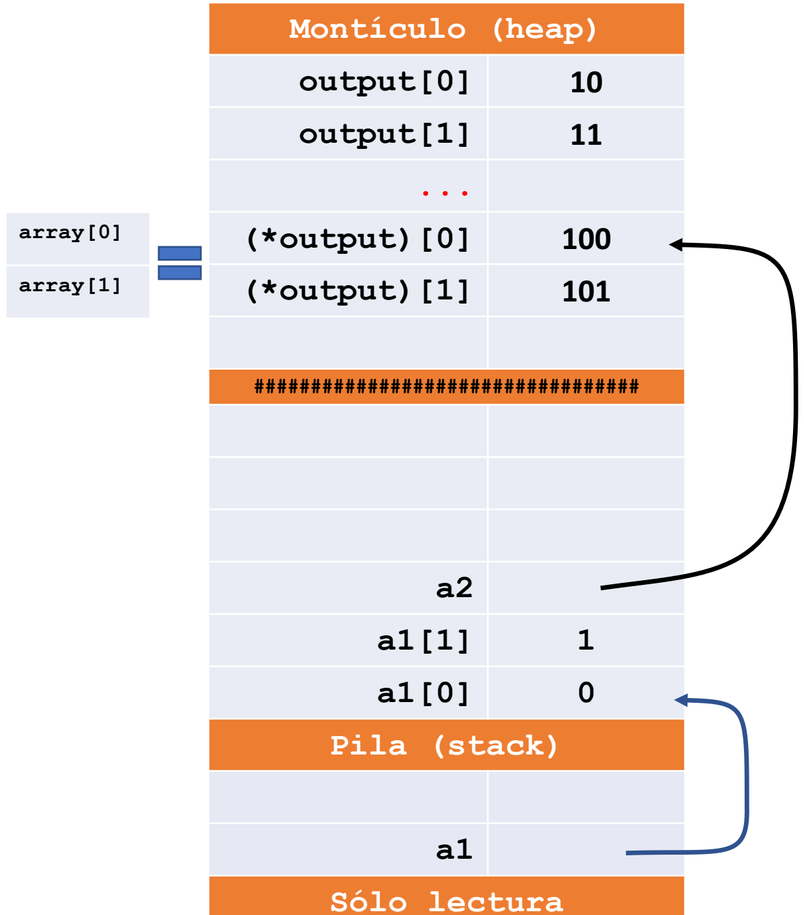
PC



1. El contador de programa (*program counter* o PC) comienza en la primera sentencia del programa. La memoria aún está vacía.
2. Tras ejecutar la **línea 19**, se reserva memoria: (i) en **la pila** para dos enteros, y (ii) en unan zona de **sólo lectura** para un puntero a un entero.
3. La segunda declaración (**línea 20**), reserva memoria **en la pila** para un puntero a un entero.
4. Al invocarse **f**, aparecen dos nuevas variables de tipo puntero en la pila: **input** y **output**, cuyo contenido es **a1** y la dirección de **a2**, respectivamente.
5. Se reserva memoria para dos enteros en el montículo, y se asigna la dirección devuelta por **malloc()** a la variable **input**.
6. Se reserva memoria nuevamente para dos enteros en el montículo, y se asigna la dirección devuelta por **malloc**() a ***output**, es decir, a un **puntero a un entero**.
7. Se crea un nuevo **int *** que apunta a la memoria reservada en la línea 8. Nótese la equivalencia.
8. Se asignan valores.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define T 2
5
6  void f(int * input, int ** output) {
7      input = (int *) malloc(T * sizeof(int));
8      *output = (int *) malloc(T * sizeof(int));
9      int * array = *output;
10
11     input[0] = 10;
12     input[1] = 11;
13
14     array[0] = 100;
15     array[1] = 101;
16 }
17
18 int main() {
19     int   a1[T] = {0,1};
20     int * a2;
21
22     f(a1, &a2);
23
24     return 0;
25 }
```

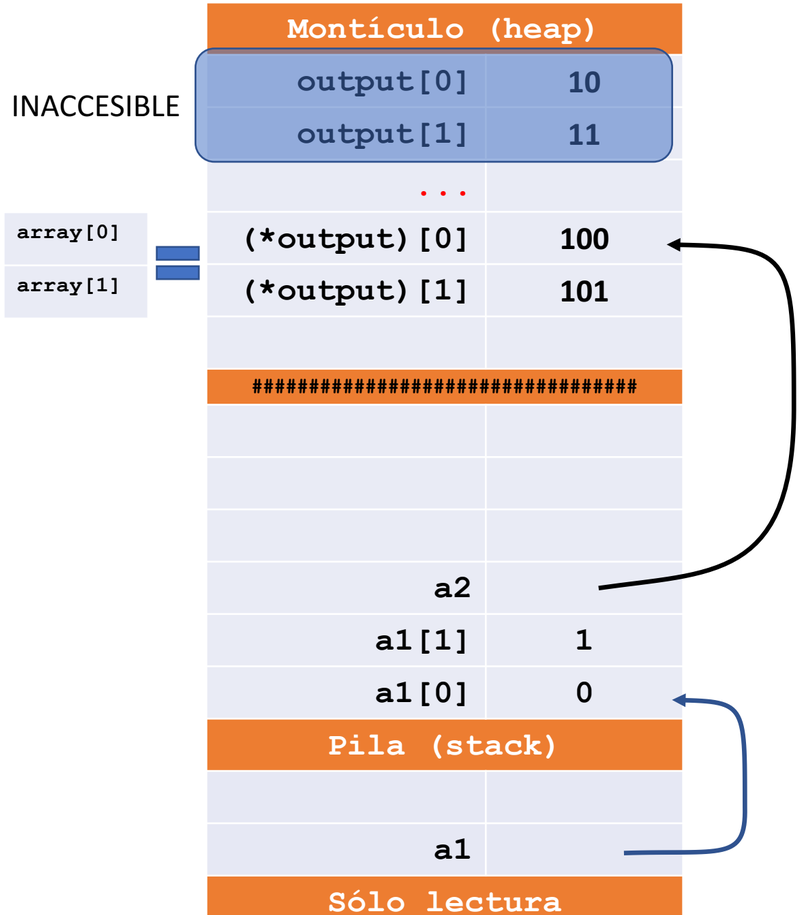
PC



1. El contador de programa (*program counter* o PC) comienza en la primera sentencia del programa. La memoria aún está vacía.
2. Tras ejecutar la **línea 19**, se reserva memoria: (i) en **la pila** para dos enteros, y (ii) en unan zona de **sólo lectura** para un puntero a un entero.
3. La segunda declaración (**línea 20**), reserva memoria **en la pila** para un puntero a un entero.
4. Al invocarse **f**, aparecen dos nuevas variables de tipo puntero en la pila: **input** y **output**, cuyo contenido es **a1** y la dirección de **a2**, respectivamente.
5. Se reserva memoria para dos enteros en el montículo, y se asigna la dirección devuelta por **malloc()** a la variable **input**.
6. Se reserva memoria nuevamente para dos enteros en el montículo, y se asigna la dirección devuelta por **malloc**() a ***output**, es decir, a un **puntero a un entero**.
7. Se crea un nuevo **int *** que apunta a la memoria reservada en la línea 8. Nótese la equivalencia.
8. Se asignan valores.
9. Fin de la función. Se eliminan variables de la pila al regresar al contexto del main().

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define T 2
5
6  void f(int * input, int ** output) {
7      input = (int *) malloc(T * sizeof(int));
8      *output = (int *) malloc(T * sizeof(int));
9      int * array = *output;
10
11     input[0] = 10;
12     input[1] = 11;
13
14     array[0] = 100;
15     array[1] = 101;
16 }
17
18 int main() {
19     int   a1[T] = {0,1};
20     int * a2;
21
22     f(a1, &a2);
23
24     return 0;
25 }
```

PC



Resultado

1. Los contenidos del array **a1** quedan intactos
2. El puntero **a2** ahora apunta a una zona de memoria para dos enteros, accesibles usando también el operador **[ ]**.
3. No hay ningún puntero a los bytes reservados en el montículo en la línea 7. Esa memoria es inaccesible y no se puede liberar. A este problema se le denomina **fuga de memoria**