MEMORIA DINÁMICA Y PASAJE DE PARÁMETROS

- Explicación Práctica -

TALLER DE LENGUAJES 1

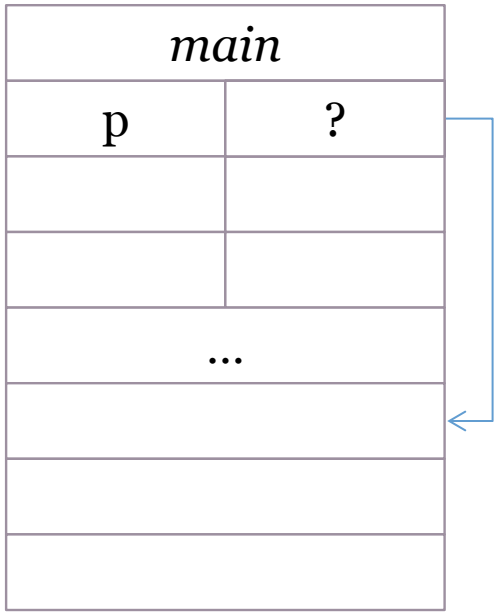
Ejercicio 1

Escriba un programa que reserve memoria en forma dinámica para un vector de 10 números enteros. Luego, imprima en la pantalla la dirección donde se encuentra el vector. Por último, libere la memoria reservada.

Solución propuesta

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

#define CANT 10

int main(){

    int * p = NULL;

    p = (int * ) malloc(CANT*sizeof(int));

    if (p == NULL)
        printf("p es NULL.");
    else
        printf("p = %p \n",p);

    free(p);

    return 0;
}
```

46FA



p = 46FA

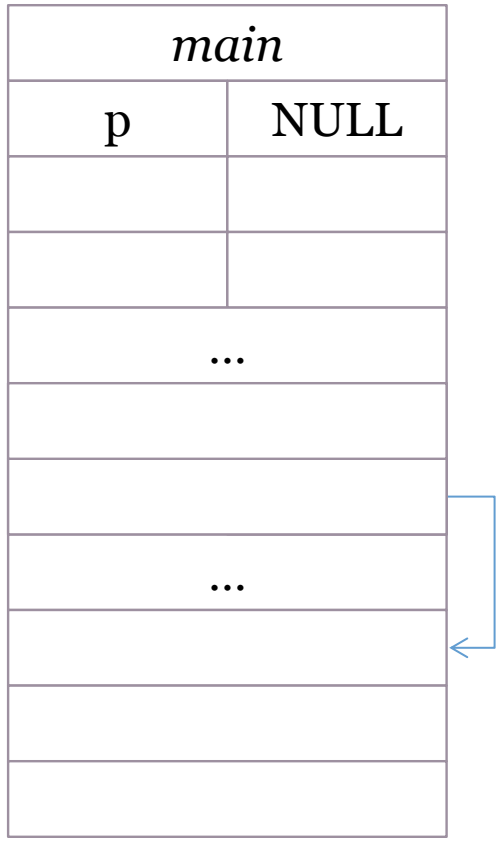
¿Qué ocurre con la memoria?

Ejercicio 2

Rehaga el Ejercicio 1 pero en este caso modularice la reserva de memoria.

Solución propuesta 1

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

#define CANT 10

void reservar (int * p);

int main(){

    int * p = NULL;

    reservar(p);

    if (p == NULL)
        printf("p es NULL.\n");
    else
        printf("p = %p \n",p);

    free(p);

    return 0;
}

void reservar (int * p) {
    p = (int * ) malloc(CANT*sizeof(int));
}
```



46FA

p es NULL

¿Qué ocurre con la memoria?

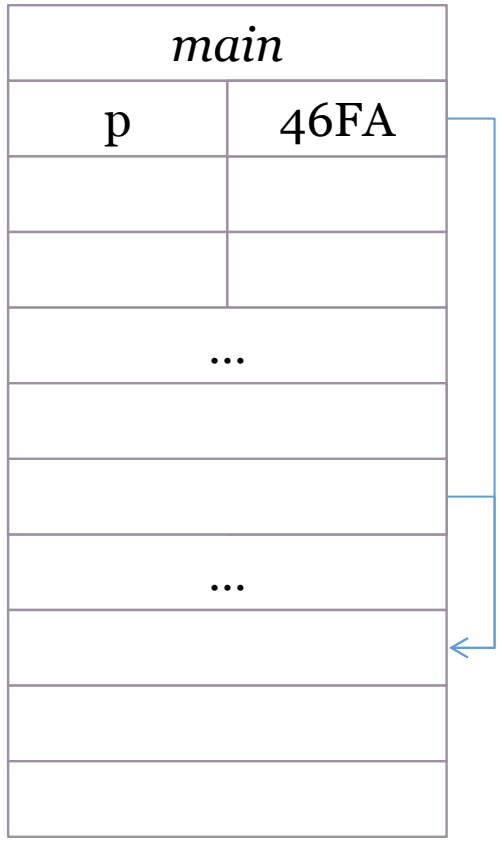
Solución propuesta 1

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

#define CANT 10

void reservar (int * p);

int main(){

    int * p = NULL;

    reservar(p);

    if (p == NULL)
        printf("p es NULL.");
    else
        printf("p = %p \n",p);

    free(p);

    return 0;
}

void reservar (int * p) {
    p = (int * ) malloc(CANT*sizeof(int));
}
```

En C todos los parámetros pasan por valor. El puntero *p* de la función *reservar* es una copia del puntero *p* de la función *main*.

Solución propuesta 2

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

#define CANT 10

int * reservar ();

int main(){

    int * p = NULL;

    p = reservar();

    if (p == NULL)
        printf("p es NULL.");
    else
        printf("p = %p \n",p);

    free(p);

    return 0;
}

int * reservar () {
  int * p = (int * ) malloc(CANT*sizeof(int));
  return p;
}
```



46FA

p = 46FA

¿Qué ocurre con la memoria?

Solución propuesta 2

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

#define CANT 10

int * reservar ();

int main(){

    int * p = NULL;

    p = reservar();

    if (p == NULL)
        printf("p es NULL.");
    else
        printf("p = %p \n",p);

    free(p);

    return 0;
}

int * reservar () {
  int * p = (int * ) malloc(CANT*sizeof(int));
  return p;
}
```

El puntero *p* de la función *main* se actualiza con el valor modificado del puntero *p* de la función *reservar*.

Ejercicio 3

Rehaga el Ejercicio 2 pero en este caso agregue una función que inicialice el vector con números ingresados desde teclado.

Solución

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

#define CANT 10

int * reservar ();

void inicializar (int * p);

int main(){

  int * p = NULL;

  p = reservar();

  if (p == NULL)
      printf("p es NULL.");
  else
      printf("p = %p \n",p);

  inicializar(p);

  free(p);

  return 0;
}
```

```c
int * reservar () {
  int * p = NULL;
  p = (int *) malloc(CANT*sizeof(int));
  return p;
}

void inicializar (int * p) {
  int i;
  for (i=0; i < CANT; i++)
    scanf("%d",&p[i]);
}
```

Notar que en el caso de la función *inicializar* no es necesario retornar el puntero *p*, ya que no se modifica la dirección a la que apunta sino los valores apuntados.