Inicialización de punteros

Al declarar un puntero, o si se dejar de utilizar y luego se utiliza nuevamente, es recomendable inicializar el valor del puntero para evitar que se interprete accidentalmente como una dirección de memoria.

Para evitar esta situación, los punteros se pueden inicializar a un valor que se conoce como nulo:

NULL

Si se declara el puntero:

int *puntero;

La siguiente instrucción lo inicializa a NULL:

puntero = *NULL*;

En C++ también se puede escribir:

puntero = nullptr;

o

puntero = *0*;

Reservar y liberar memoria desde nuestros programas

Cuando nuestros programas necesitan utilizar más memoria de la que se les asigna a través de la declaración de variables, para almacenamiento temporal de datos, debemos solicitar al sistema operativo que ceda más recurso de memoria. Al terminar de utilizar esta memoria en nuestros programas, podemos devolverla al sistema operativo para que la tenga disponible para otras aplicaciones.

Podemos hablar, entonces, de que nuestros programas están haciendo un uso dinámico de la memoria de la computadora. para ello, desde las primeras versiones de C, se cuenta con las funciones:

*void *malloc(entero)*

void free(puntero)

La función *malloc* reserva un espacio de memoria del tamaño que se indica dentro de los paréntesis y retorna su dirección de inicio. Suele combinarse con la función *sizeof( )*, que devuelve un entero que indica el tamaño, en bytes, del identificador o tipo colocado dentro de los paréntesis.

La función *free* libera la memoria a la que apunta el puntero que se coloca dentro de sus paréntesis y la pone nuevamente a disposición del sistema operativo.

Ambas se encentran en la librería *cstdlib* y en el espacio de memoria *std*.

El lenguaje C++ provee los comandos *new* y *delete*. Ambos tienen varias formas de utilizarse, algunas son:

new tipo, retorna la dirección de inicio del espacio de memoria reservado del tamaño de *tipo*. Este puede ser un tipo primitivo o un tipo definido por el programador.

new tipo(argumentos), lo mismo que el anterior e invoca al constructor del objeto del tipo.

new tipo[entero], reserva un espacio suficientemente grande para almacenar un arreglo de objetos del tipo especificado.

delete puntero, libera el espacio de memoria que fue reservado con *new*, y que es apuntado por el puntero.

delete [ ] puntero, libera todo el espacio que fue reservado con *new*, para un arreglo deobjetos del tipo del puntero.

Ambos se encentran en la librería *new* o puede no colocarse nombre de librería en la cabecera del programa.

Ejemplo:

Observar en el siguiente programa las formas de uso de new y las formas de uso de delete.

```
#include <iostream>

using namespace std;

int main(void)
{

   cout << endl;
   cout << "ADICIÓN DE MEMORIA DINÁMICA EN UN PROGRAMA" << endl << endl;

   ////////////////////////////

   cout << "PRIMER EJEMPLO" << endl << endl;

   float *p;

   p = NULL;

   if(!p) // Forma abreviada de p == NULL
      cout << "El puntero está a NULL" << endl;
   else
      cout << "El puntero NO está a NULL" << endl;

   p = new float;

   *p = 3.57;

   cout << "Valor almacenado en el espacio creado: " << *p << endl << endl;

   free(p);

   p = NULL;

   ////////////////////////////

   cout << "SEGUNDO EJEMPLO" << endl << endl;

   float *q = new float(8.4); // Aquí estamos haciendo tres cosas a la vez:
                              // declarando un puntero, asignándole la
                              // dirección de un espacio de memoria creado
                              // dinámicamente y asignándole un valor inicial
                              // al espacio de memoria.

   cout << "Valor almacenado en el espacio creado: " << *q << endl;

   *q = 25.4;
```

```
    cout << "Otro valor asignado a ese espacio: " << *q << endl << endl;

    free(q);

    q = NULL;

    ////////////////////////////

    cout << "TERCER EJEMPLO" << endl << endl;

    int *r = (int *) malloc(8 * sizeof(int)); // Crear un arreglo de
                                              // ocho casillas de
                                              // forma dinámica.
    //int *r = new int[8];

    for(int i = 0; i <= 8; i++)
       *(r + i) = i * 2;

    for(int i = 0; i <= 8; i++)
       cout << *(r + i) << "  ";

    free(q);
    //delete [] q;


    cout << endl << endl;

    return 0;
}
```