Carátula para entrega de prácticas

Facultad de Ingeniería Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	*Marco Antonio Martinez Quintana*
Asignatura:	*Estructuras de datos y algoritmos I*
Grupo:	*17*
No de Práctica(s):	*4*
Integrante(s):	*González Cuellar Arturo*
No. de equipo de cómputo empleado:	*38*
No. de Lista o Brigada	
Semestre:	*2020-2*
Fecha de entrega:	*29 - Febrero - 2020*
Observaciones:	

Calificación: _____

Objetivo:

Utilizarás funciones en lenguaje *C* que permiten reservar y almacenar información de manera dinámica (en tiempo de ejecución).

Introducción:

La memoria dinámica es memoria que se reserva en tiempo de ejecución, esta tiene muchas ventajas, la principal es que esta puede variar durante la ejecución del programa. El uso de la memoria dinámica es necesario cuando no se sabe el número exacto de datos con los que se va a trabajar.

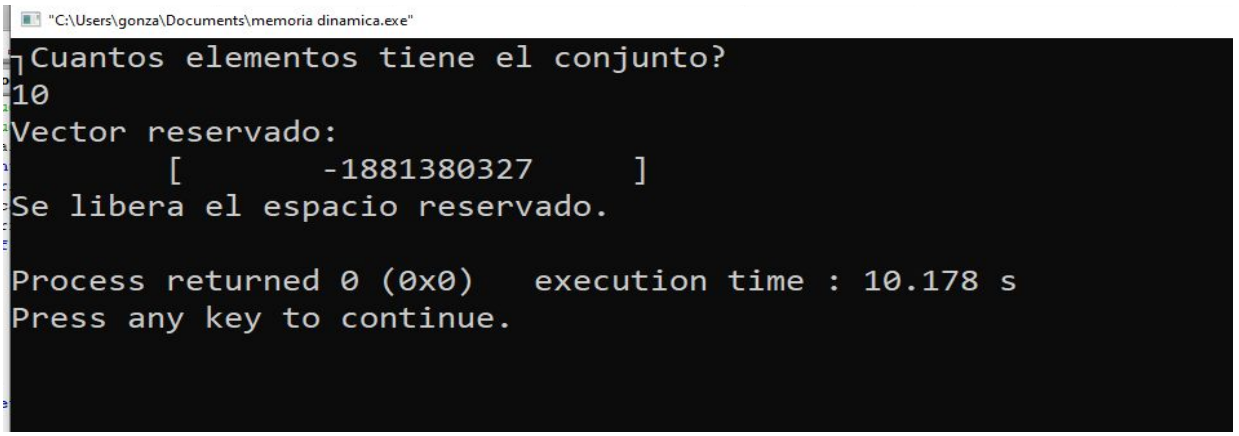
Desarrollo y resultados.

Códigos:

//Código 1

Se definen dos variables, un apuntador y dos enteros, primero se piden los elementos del conjunto se pone el arreglo con el tipo de dato apuntado, seguido de la función malloc, este tiene como parametro el numero de bytes de la variable.
Se pone un ciclo if con la condición de que si el arreglo no es cero, entonces entra a un ciclo for en el cual se recorre todo el arreglo que está funcionando con la función malloc, si malloc es incapaz de reservar el bloque de memoria, porque no hay memoria suficiente u otras, entonces nos devuelve un puntero nulo.

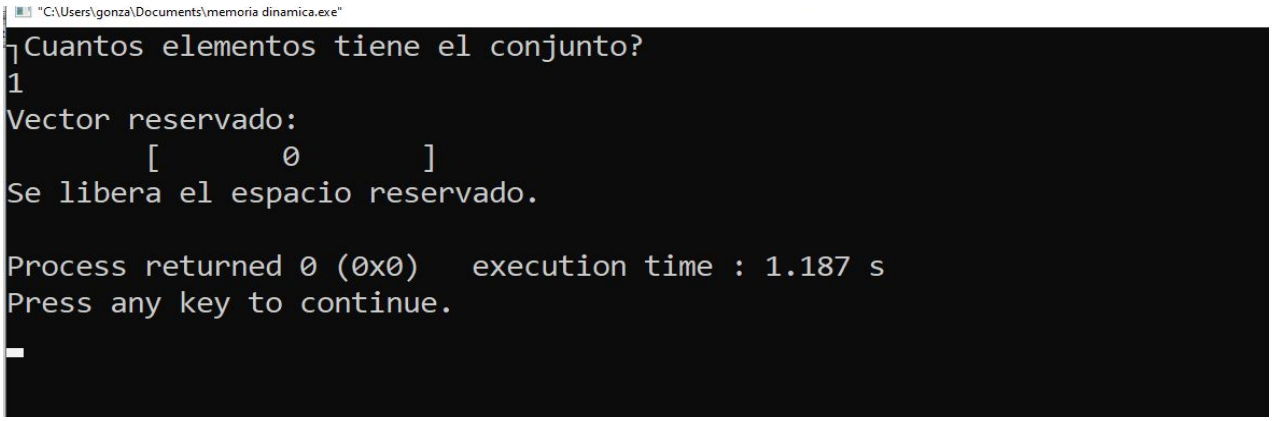
```
#include <stdio.h>
#include <stdlib.h>
int main (){
   int *arreglo, num, cont;
   printf("¿Cuantos elementos tiene el conjunto?\n");
   scanf("%d",&num);
   arreglo = (int *)malloc (num * sizeof(int));
   if (arreglo!=NULL) {
      printf("Vector reservado:\n\t["); for (cont=0 ; cont<num ; cont++){
      }
      printf("\t%d",*(arreglo+cont));
      printf("\t]\n");
      printf("Se libera el espacio reservado.\n");
      free(arreglo);
   }
   return 0;
}
```



//Codigo 2

En este código se declara un arreglo con puntero y dos variables de tipo entero, para empezar se pide cuántos elementos tiene el conjunto y se guarda en una de las dos variables definidas.
Posteriormente se asigna al arreglo con la función calloc, para empezar se pone el tipo de dato, en este caso entero pero este está apuntado, después se pone la función calloc con dos parámetros, el primero es el número de elementos y en segundo es el tamaño pero con la función sizeof.
Después se declara un if el cual primero revisa que el arreglo no este vacio, si esto no es cierto entonces entra a un ciclo for en el cual se va recorriendo los datos del arreglo para poder liberar la memoria, al final solo imprime el vector nulo ya que se libero la memoria y además de que la función free no retorna ningún valor.

```
/*#include <stdio.h>
#include <stdlib.h>
int main (){
   int *arreglo, num, cont;
   printf("¿Cuantos elementos tiene el conjunto?\n");
   scanf("%d",&num);
   arreglo = (int *)calloc (num, sizeof(int));
   if (arreglo!=NULL) {
      printf("Vector reservado:\n\t["); for (cont=0 ; cont<num ; cont++){
      }
      printf("\t%d",*(arreglo+cont));
      printf("\t]\n");
      printf("Se libera el espacio reservado.\n");
      free(arreglo);
   }
   return 0;
}*/
```



//Codigo 3

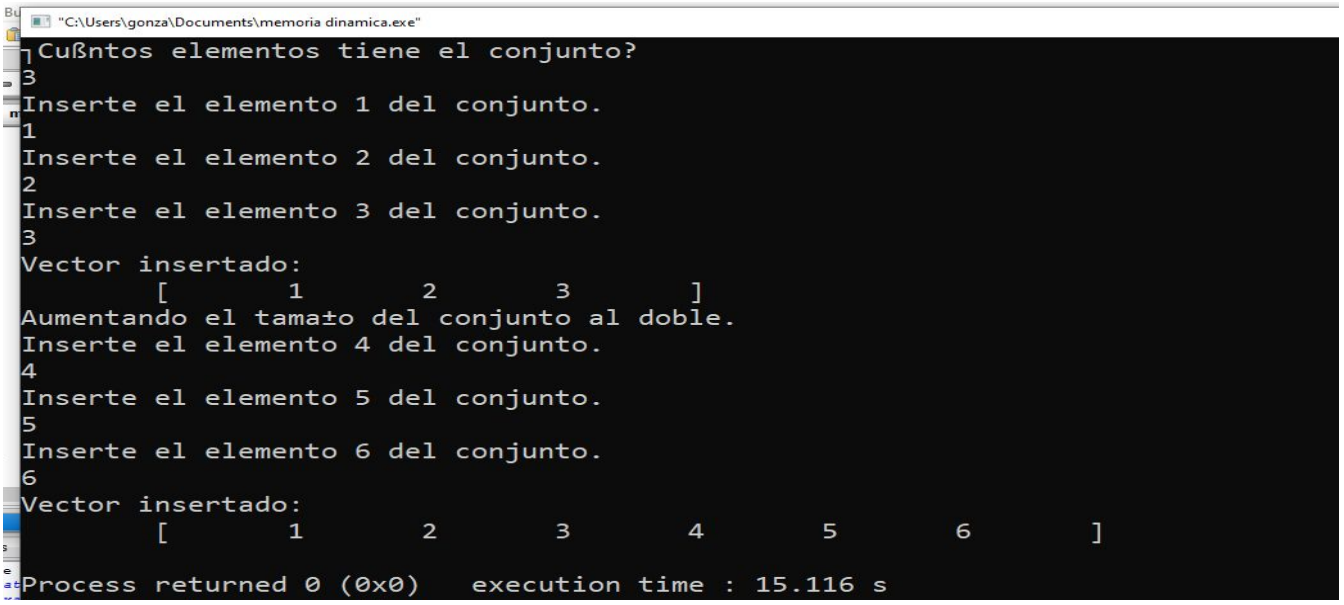
En este código se realiza los mismo que el primer código con la diferencia de que ahora muestra los números ingresados por el usuario,se hace uso de la función malloc para reservar el espacio de memoria, cuando se ingresa el número de elementos que tiene el vector se solicitan los elementos y se almacenan y posteriormente con la función realloc se duplica el número de elementos, estás función nos ayuda a poder cambiar el tamaño del objeto apuntado.

```
#include <stdio.h>
#include <stdlib.h>
int main (){
   int *arreglo, *arreglo2, num, cont;
   printf("¿Cuántos elementos tiene el conjunto?\n");
   scanf("%d",&num);
   arreglo = (int *)malloc (num * sizeof(int));
   if (arreglo!=NULL) {
      for (cont=0 ; cont < num ; cont++){
         printf("Inserte el elemento %d del conjunto.\n",cont+1);
         scanf("%d",(arreglo+cont));
      }
      printf("Vector insertado:\n\t[");
      for (cont=0 ; cont < num ; cont++){ printf("\t%d",*(arreglo+cont));
      }
      printf("\t]\n");
      printf("Aumentando el tamaño del conjunto al doble.\n");
      num *= 2;
      arreglo2 = (int *)realloc (arreglo,num*sizeof(int));
      if (arreglo2 != NULL) {
         arreglo = arreglo2;
         for (; cont < num ; cont++){
```

```
            printf("Inserte el elemento %d del conjunto.\n",cont+1);
               scanf("%d",(arreglo2+cont));

            }
            printf("Vector insertado:\n\t[");
            for (cont=0 ; cont < num ; cont++){

            printf("\t%d",*(arreglo2+cont));
            }
            printf("\t]\n");
         }
            free (arreglo);
      }
            return 0;
   }
```



Conclusión:

El desarrollo de esta práctica fue de gran importancia ya que con ejercicios entendimos de mejor manera el funcionamiento de la memoria dinámica, sus ventajas y como se puede aplicar esta, además las funciones disponibles para trabajar con ella. Para el desarrollo de esta práctica es necesario tener conocimientos de punteros y de arreglos, ya que todo esto va ligado, así como que hay que tener mucho paciencia y cuidado ya que se pueden cometer errores en asignaciones, códigos, lógica, etc.

Referencias:

https://consejotecnologico.com/memoria-dinamica-en-c/
Apuntes de clase.