



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación
Salas A y B

Profesor:

Martínez Quintana Marco Antonio

Asignatura:

Estructura de Datos y Algoritmos 1.

Grupo:

17

No de Práctica(s):

4

Integrante(s):

Ruiz Godoy Franco

*No. de Equipo de
cómputo empleado:*

7

No. de Lista o Brigada:

Semestre:

2020-2

Fecha de entrega:

29 de Febrero de 2020

Observaciones:

CALIFICACIÓN:

Objetivo.

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

Introducción.

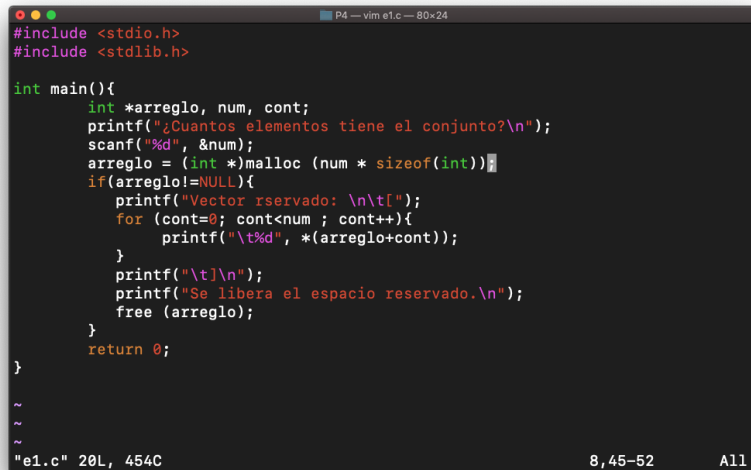
La memoria dinámica es memoria que se reserva en tiempo de ejecución. Su uso es necesario cuando no se conoce el número de datos/elementos a tratar y depende del espacio que se va a usar.

Existen diferentes funciones de memoria dinámica como “malloc” que sirve para reservar un espacio de memoria tan grande como se especifica dentro de la función, “calloc” que sirve para reservar un espacio de memoria tan grande como se especifica dentro de la función y al mismo tiempo inicializa todos a 0, “realloc” que redimensiona un tipo de dato que asignamos con malloc pero conservando sus valores y “free” que sirve para que la memoria dinámica que es reservada sea eliminada siempre al terminar la ejecución del programa por el propio sistema operativo.

Desarrollo y Resultados.

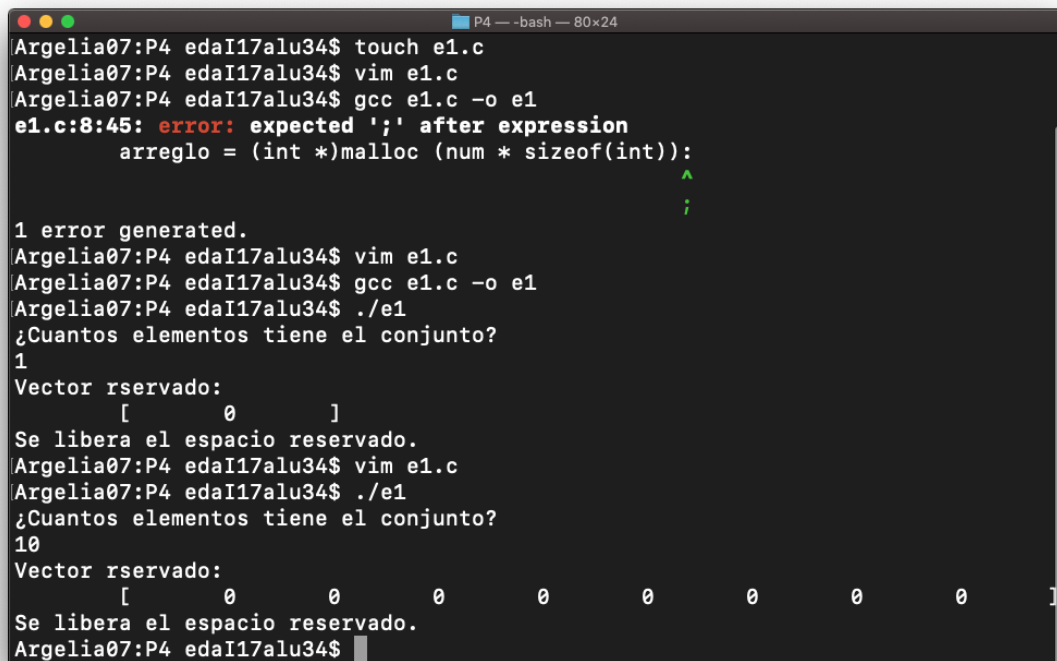
Código 1.

Capturas de pantalla.



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

int main(){
    int *arreglo, num, cont;
    printf("¿Cuántos elementos tiene el conjunto?\n");
    scanf("%d", &num);
    arreglo = (int *)malloc (num * sizeof(int));
    if(arreglo!=NULL){
        printf("Vector rservado: \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;
}
```



```
Argelia07:P4 edaI17alu34$ touch e1.c
Argelia07:P4 edaI17alu34$ vim e1.c
Argelia07:P4 edaI17alu34$ gcc e1.c -o e1
e1.c:8:45: error: expected ';' after expression
    arreglo = (int *)malloc (num * sizeof(int));
                                   ^
1 error generated.
Argelia07:P4 edaI17alu34$ vim e1.c
Argelia07:P4 edaI17alu34$ gcc e1.c -o e1
Argelia07:P4 edaI17alu34$ ./e1
¿Cuántos elementos tiene el conjunto?
1
Vector rservado:
[ 0 ]
Se libera el espacio reservado.
Argelia07:P4 edaI17alu34$ vim e1.c
Argelia07:P4 edaI17alu34$ ./e1
¿Cuántos elementos tiene el conjunto?
10
Vector rservado:
[ 0 0 0 0 0 0 0 0 0 0 ]
Se libera el espacio reservado.
Argelia07:P4 edaI17alu34$
```

```
Argelia07:P4 edaI17alu34$ vim e3.c
Argelia07:P4 edaI17alu34$ ./e1
¿Cuantos elementos tiene el conjunto?
-1
Argelia07:P4 edaI17alu34$ ./e1
¿Cuantos elementos tiene el conjunto?
0
Vector rservado:
[ ]
Se libera el espacio reservado.
Argelia07:P4 edaI17alu34$ ./e1
¿Cuantos elementos tiene el conjunto?
0.9
Vector rservado:
[ ]
Se libera el espacio reservado.
Argelia07:P4 edaI17alu34$ vim e1.c
Argelia07:P4 edaI17alu34$ ./e1
¿Cuantos elementos tiene el conjunto?
5
Vector rservado:
[ 0 0 0 0 0 0 ]
Se libera el espacio reservado.
Argelia07:P4 edaI17alu34$
```

Explicación.

Como se usa la función "malloc" lo que hace el código al ejecutarse es pedir el parámetro de número de bytes para reservarla como memoria, una vez ingresado muestra entre corchetes los espacios reservados. Si le damos un número en decimal lo mandara directamente a NULL ya que la variable a la que estamos dando valor es de tipo entero, dando un número negativo de igual forma nos mandara directamente a NULL, no se puede reservar el espacio en la memoria.

Código 2.

Capturas de pantalla.

```
P4 — vim e2.c — 80x24
#include <stdio.h>
#include <stdlib.h>

int main (){
    int *arreglo, num, cont;
    printf("¿Cuántos 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;
}
~
~
~
-- INSERT --                               6,25-31    All
```

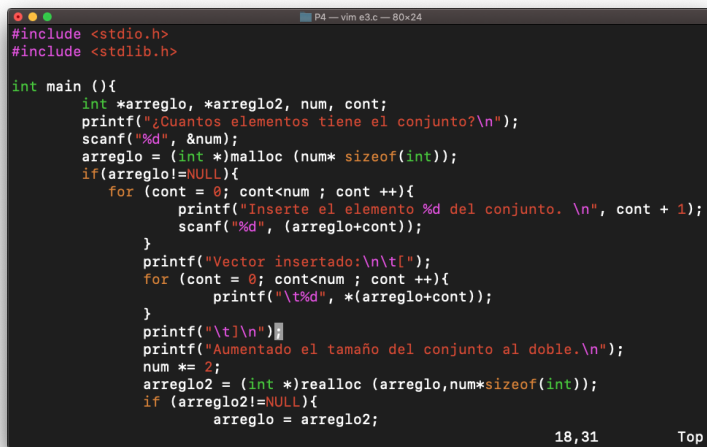
```
P4 — -bash — 80x24
Argelia07:P4 edaI17alu34$ vim e2.c
Argelia07:P4 edaI17alu34$ gcc e2.c -o e2
Argelia07:P4 edaI17alu34$ ./e2
¿Cuántos elemrntos tiene el conjunto?
2
Vector reservado:
[      0      0      ]
Se libera el espacio reservado.
Argelia07:P4 edaI17alu34$ vim e2.c
Argelia07:P4 edaI17alu34$ ./e2
¿Cuántos elemrntos tiene el conjunto?
1
Vector reservado:
[      0      ]
Se libera el espacio reservado.
Argelia07:P4 edaI17alu34$ vim e2.c
Argelia07:P4 edaI17alu34$ gcc e2.c -o e2
Argelia07:P4 edaI17alu34$ ./e2
¿Cuántos elementos tiene el conjunto?
3
Vector reservado:
[      0      0      0      ]
Se libera el espacio reservado.
Argelia07:P4 edaI17alu34$
```

Explicación.

La función “calloc” en este código al momento de ejecutarse pide como parámetro el número de bytes en la memoria que desea reservarse, una vez dado este parámetro inicializa a todos a 0. Básicamente hace lo mismo que la función de “malloc” agregando que inicializa a 0 y es más lento.

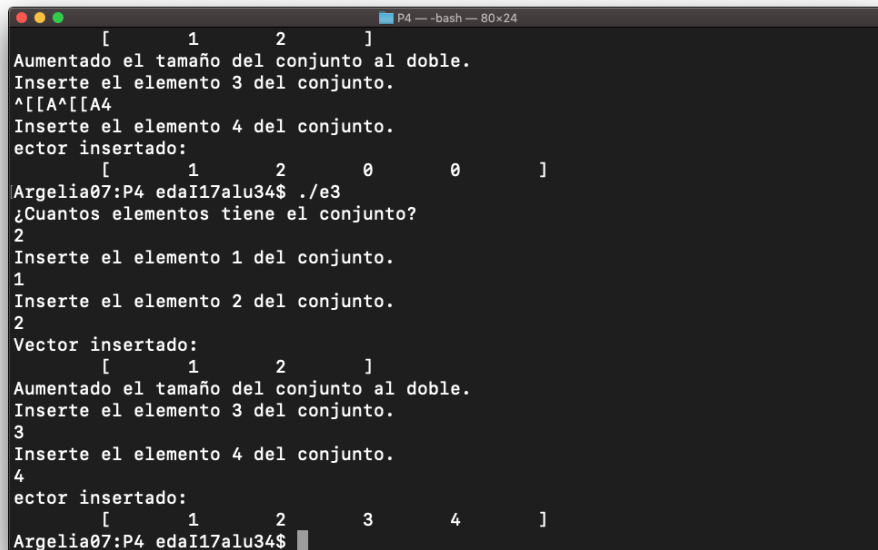
Código 3.

Capturas de pantalla.



```
#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 *)calloc (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("Aumentado el tamaño del conjunto al doble.\n");
        num *= 2;
        arreglo2 = (int *)realloc (arreglo,num*sizeof(int));
        if (arreglo2!=NULL){
            arreglo = arreglo2;
        }
    }
}
```



```
[ 1 2 ]
Aumentado el tamaño del conjunto al doble.
Inserte el elemento 3 del conjunto.
^[[A^[A4
Inserte el elemento 4 del conjunto.
ector insertado:
[ 1 2 0 0 ]
Argelia07:P4 edaI17alu34$ ./e3
¿Cuántos elementos tiene el conjunto?
2
Inserte el elemento 1 del conjunto.
1
Inserte el elemento 2 del conjunto.
2
Vector insertado:
[ 1 2 ]
Aumentado el tamaño del conjunto al doble.
Inserte el elemento 3 del conjunto.
3
Inserte el elemento 4 del conjunto.
4
ector insertado:
[ 1 2 3 4 ]
Argelia07:P4 edaI17alu34$
```

```
P4 — -bash — 80x24
1
Inserte el elemento 2 del conjunto.
2
Vector insertado:
[ 1 2 ]
Aumentado el tamaño del conjunto al doble.
Inserte el elemento 3 del conjunto.
3
Inserte el elemento 4 del conjunto.
4
ector insertado:
[ 1 2 3 4 ]
Argelia07:P4 edaI17alu34$ ./e3
¿Cuántos elementos tiene el conjunto?
-2
Argelia07:P4 edaI17alu34$ ./e3
¿Cuántos elementos tiene el conjunto?
0.9
Vector insertado:
[ ]
Aumentado el tamaño del conjunto al doble.
ector insertado:
[ ]
Argelia07:P4 edaI17alu34$
```

Explicación.

La función “realloc” trabaja en conjunto con “malloc” , por su parte malloc tomando el parámetro el número de bytes ingresado para la reservación de memoria y realloc por su parte en este caso redimensiona al doble el número de bytes que se ingresaron desde malloc. De igual forma si se le envía un decimal te enviara directo a NULL por no ser un entero y en caso de un número negativo hará lo mismo ya que no se puede reservar en la memoria.

Conclusión.

El uso de las funciones de la memoria dinámica es una herramienta útil en ciertos aspectos ya que aporta grandes ventajas como dar la capacidad de modificar tamaño de y memoria reservada y a la vez esta puede ser liberada con la función "free". Pero cuenta con más desventajas como el volverse más complejo el código y que se detecten con más dificultad los errores en dichos códigos, por último hace más lento un programa que la implemente pues ocupa más instrucciones.

Referencias.

- Escuela Politécnica de Ejercito.. (2012, 19 diciembre). Memoria dinámica. Recuperado 29 febrero, 2020, de <https://es.slideshare.net/gusolis93/memoria-dinamica-15706001>
- Oscar Rosas, R. O. (2016, 28 diciembre). Memoria Dinámica. Recuperado 29 febrero, 2020, de <https://compilandoconocimiento.com/2016/12/24/memoria-dinamica/>