# Memoria dinámica arreglos

lunes, 29 de noviembre de 2021 07:17 a. m.

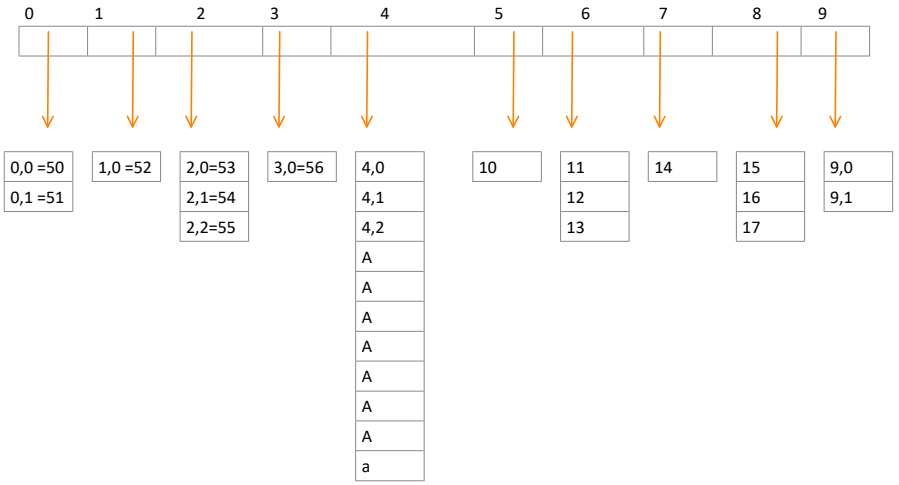
Arreglo estático argv:

| | | | | | | |
|---|---|---|---|---|---|---|
| H | o | l | a | '\0' | | |
| 1 | 0 | '\0' | '\0' | '\0' | | |
| 1 | 0 | . | 5 | '\0' | | |
| E | S | C | U | E | L | A |
| | | | | | | |

void free(void* ptr): Esta función permite liberar la memoria obtenida por malloc, calloc o realloc. Es función solo le indica al S.O. que esa sección de memoria ha quedado libre.

Ejemplo:

Arreglo multidimensional: Generar la siguiente estructura dinámica para 20 datos enteros: arreglo2dim:




Código:

```
#include "memoriaDinamica1.h"

/*
Esdta función genera una matriz de tam NxM.
@param tamPD Valor que representa el tamaño del arreglo en filas (N, como primera
dimensión).
@param tamSD Valor que representa el tamaño del arreglo en columnas (M. como segunda
dimensión).
*/
void crearTablaDinamicaNxM(int tamPD, int tamSD){
    //Se reserva memoria para la primera dimensión, es decir las filas de la matriz
    int** arreglo2dim=(int**) malloc(sizeof(int*)*tamPD);
    if(arreglo2dim==NULL){
        puts("Se termino la memoria");
        exit(1);
    }
    int i, j;
    for(i=0; i<tamPD; i++){
        //Se reserva memoria para la segunda dimensión, es decir las columnas matriz,
N(tamSD) columnas por cada fila
        arreglo2dim[i]=(int *)malloc(sizeof(int)*tamSD);
        if(arreglo2dim[i]==NULL){
            puts("Se termino la memoria para las columnas");
            exit(1);
        } else{
            printf("Se reservo memoria para la fila %d\n.", i);
        }
    }
    //Se llena de información la matriz
    for(i=0; i<tamPD; i++){
        for(j=0; j<tamSD; j++){
            arreglo2dim[i][j]=i+j;
        }
    }
    //Se imprime todo el contenido de la matriz
    for(i=0; i<tamPD; i++){
        for(j=0; j<tamSD; j++){
           printf("El valor en pos i=%d, j=%d es %d.\n", i,j, arreglo2dim[i][j]);
        }
    }
    //Liberación de memoria, cuando ya se terminó de utilizar la variable dinámica (en
este caso la matriz):
    for(i=0; i<tamPD; i++)
        free(arreglo2dim[i]);//Se libera la memoria de todas las columnas por cada fila
    //Se libera la memoria de las filas
    free(arreglo2dim);
}
/*
@param tamPrimeraDimension Valor que representa el tamaño del arreglo en filas.
@retur devuelve el arreglo de dos dimensiones ubicado en la heap memory
*/
int** crearTablaDinamica(int tamPrimeraDimension){
    //Reservar memoria para la primera dimensión-> filas
    int** arreglo2dim=(int**) malloc(sizeof(int*)*tamPrimeraDimension);
    int i;
    if(arreglo2dim==NULL){ //No devolvio apuntador
        puts("Se termino la memoria para las filas");
```

```
        exit(1);
    } else{
        for(i=0; i<tamPrimeraDimension; i++){
            if(i==0 || i==9)
                arreglo2dim[i]=(int *) malloc(sizeof(int)*2);
            else if(i%2!=0)
                arreglo2dim[i]=(int *) malloc(sizeof(int)*1);
            else
                arreglo2dim[i]=(int *) malloc(sizeof(int)*3);
        }
    }
    //utilizar la matriz generada



    //liberar los recursos:

.
}
void iniciaOperacion(int argc, char** argv){
    //Aqui hacer todo lo que hace la función main
    puts("Inicio de programa");
    crearTablaDinamicaNxM(4000,4000);
    puts("Termino el programa exitosamente");
}
```

Segundo ejemplo (utilizando funciones que gestionan la creación, pruebas y destrucción de una tabla dinámica)

Archivo.c:

```
#include "memoriaDinamica1.h"

/*
Esdta función genera una matriz de tam NxM.
@param tamPD Valor que representa el tamaño del arreglo en filas (N, como primera
dimensión).
@param tamSD Valor que representa el tamaño del arreglo en columnas (M. como segunda
dimensión).
*/
void crearTablaDinamicaNxM(int tamPD, int tamSD){
    //Se reserva memoria para la primera dimensión, es decir las filas de la matriz
    int** arreglo2dim=(int**) malloc(sizeof(int*)*tamPD);
    if(arreglo2dim==NULL){
        puts("Se termino la memoria");
        exit(1);
    }
    int i, j;
    for(i=0; i<tamPD; i++){
         //Se reserva memoria para la segunda dimensión, es decir las columnas matriz,
N(tamSD) columnas por cada fila
        arreglo2dim[i]=(int *)malloc(sizeof(int)*tamSD);
        if(arreglo2dim[i]==NULL){
            puts("Se termino la memoria para las columnas");
            exit(1);
        } else{
            printf("Se reservo memoria para la fila %d\n.", i);
        }
    }
    //Se llena de información la matriz
    for(i=0; i<tamPD; i++){
        for(j=0; j<tamSD; j++){
            arreglo2dim[i][j]=i+j;
        }
    }
    //Se imprime todo el contenido de la matriz
    for(i=0; i<tamPD; i++){
        for(j=0; j<tamSD; j++){
           printf("El valor en pos i=%d, j=%d es %d.\n", i,j, arreglo2dim[i][j]);
        }
    }
    //Liberación de memoria, cuando ya se termino de utilizar la variable dinámica (en
este caso la matriz):
    for(i=0; i<tamPD; i++)
        free(arreglo2dim[i]);//Se libera la memoria de todas las columnas por cada fila
    //Se libera la memoria de las filas
    free(arreglo2dim);
}
/*
@param tamPrimeraDimension Valor que representa el tamaño del arreglo en filas.
@retur devuelve el arreglo de dos dimensiones ubicado en la heap memory
*/
int** crearTablaDinamica(int tamPrimeraDimension){
    //Reservar memoria para la primera dimensión-> filas
    int** arreglo2dim=(int**) malloc(sizeof(int*)*tamPrimeraDimension);
    int i;
    if(arreglo2dim==NULL){ //No devolvio apuntador
        puts("Se termino la memoria para las filas");
        exit(1);
    } else{
        for(i=0; i<tamPrimeraDimension; i++){
            if(i==0 || i==9){
```

```
                arreglo2dim[i]=(int *) malloc(sizeof(int)*2);
                if(arreglo2dim[i]==NULL){ //No devolvio apuntador
                    puts("Se termino la memoria para las columnas donde i=9 o i=0");
                    exit(1);
                }
            }
            else if(i%2!=0){
                arreglo2dim[i]=(int *) malloc(sizeof(int)*1);
                if(arreglo2dim[i]==NULL){ //No devolvio apuntador
                    puts("Se termino la memoria para las columnas donde i es impar");
                    exit(1);
                }
            }
            else{
                arreglo2dim[i]=(int *) malloc(sizeof(int)*3);
                if(arreglo2dim[i]==NULL){ //No devolvio apuntador
                    puts("Se termino la memoria para las columnas donde i es par");
                    exit(1);
                }
            }
        }
    }
    //Return
    return arreglo2dim;
}
void pruebaTablaDinamica(int** arreglo2dim, int tamPrimeraDimension){
    int i;
    //utilizar la matriz generada
    for(i=0; i< tamPrimeraDimension; i++)  //50 --> 70
        switch(i){
            case 0:
                arreglo2dim[i][0]=50;
                arreglo2dim[i][1]=51;
                break;
            case 1:
                arreglo2dim[i][0]=52;
                break;
            case 2:
                arreglo2dim[i][0]=53;
                arreglo2dim[i][1]=54;
                arreglo2dim[i][2]=55;
                break;
            case 3:
                arreglo2dim[i][0]=56;
                break;
            case 4:
                arreglo2dim[i][0]=57;
                arreglo2dim[i][1]=58;
                arreglo2dim[i][2]=59;
                break;
            case 5:
                arreglo2dim[i][0]=60;
                break;
            case 6:
                arreglo2dim[i][0]=61;
                arreglo2dim[i][1]=62;
                arreglo2dim[i][2]=63;
                break;
            case 7:
                arreglo2dim[i][0]=64;
                break;
            case 8:
                arreglo2dim[i][0]=65;
                arreglo2dim[i][1]=66;
                arreglo2dim[i][2]=67;
                break;
            case 9:
                arreglo2dim[i][0]=68;
                arreglo2dim[i][1]=69;
                break;
            default:
                puts("Desbordamiento");
                break;
        }

    /*Asignación valida:
    arreglo2dim[0][0]=10;
    arreglo2dim[0][1]=11;
    arreglo2dim[2][0]=12;
    arreglo2dim[2][1]=13;
    */
    //Pruebas de impresión de datos
    printf("El valor que se encuentra en la pos 2,1 es :%d\n",arreglo2dim[2][1]);
    printf("(desbordamiento)El valor que se encuentra en la pos 2,3 es :%d
\n",arreglo2dim[2][3]);
    printf("El valor que se encuentra en la pos 3,0 es :%d\n",arreglo2dim[3][0]);
    printf("El valor que se encuentra en la pos 9,1 es :%d\n",arreglo2dim[9][1]);
}
bool destruyeTabla(int** arreglo, int tamPrimeraDimension){
    int i;
    for(i=0; i<tamPrimeraDimension; i++)
        free(arreglo[i]);
    free(arreglo);
    return true;
}
void iniciaOperacion(int argc, char** argv){
    //Aqui hacer todo lo que hace la función main
    puts("Inicio de programa");
    //crearTablaDinamicaNxM(4000,4000);
    int** tablaX=crearTablaDinamica(10);
    //utilizando la tabla dinámica en una función:
    pruebaTablaDinamica(tablaX,10);
```

```
    destruyeTabla(tablaX, 10);
    puts("Termino el programa exitosamente");
}
```

Archivo.h:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
void crearTablaDinamicaNxM(int tamPD, int tamSD);
int** crearTablaDinamica(int tamPrimeraDimension);
void iniciaOperacion(int argc, char** argv);
void pruebaTablaDinamica(int**, int);
bool destruyeTabla(int** , int);
```

Test.c:

```
#include "memoriaDinamica1.h"
void main(int argc, char** argv){
    iniciaOperacion(argc,argv);
}
```

//Prueba de almacenamiento con cadenas exactas:

```
#include "memoriaDinamica1.h"
/*
@param tamPrimeraDimension Valor que representa el tamaño del arreglo en filas.
@retur devuelve el arreglo de dos dimensiones ubicado en la heap memory
*/
char** crearTablaDinamica(int tamPrimeraDimension){
    //Reservar memoria para la primera dimensión-> filas
    char** arreglo2dim=(char**) calloc(tamPrimeraDimension,sizeof(char*));
    int i;
    if(arreglo2dim==NULL){ //No devolvio apuntador
        puts("Se termino la memoria para las filas");
        exit(1);
    } else{
        for(i=0; i<tamPrimeraDimension; i++){
            if(i==0 || i==9){
                arreglo2dim[i]=(char*) calloc(2,sizeof(char));
                if(arreglo2dim[i]==NULL){ //No devolvio apuntador
                    puts("Se termino la memoria para las columnas donde i=9 o i=0");
                    exit(1);
                }
            }
            else if(i%2!=0){
                arreglo2dim[i]=(char*) calloc(1,sizeof(char));
                if(arreglo2dim[i]==NULL){ //No devolvio apuntador
                    puts("Se termino la memoria para las columnas donde i es impar");
                    exit(1);
                }
            }
            else{
                arreglo2dim[i]=(char*) calloc(3,sizeof(char));
                if(arreglo2dim[i]==NULL){ //No devolvio apuntador
                    puts("Se termino la memoria para las columnas donde i es par");
                    exit(1);
                }
            }
        }
    }
    //Return
    return arreglo2dim;
}
void pruebaTablaDinamica(char** arreglo2dim, int tamPrimeraDimension){
    int i;
    char* cadena=(char *) calloc(50,sizeof(char)); //arreglo dinámico para 50 caracteres
    //utilizar la matriz generada
    for(i=0; i< tamPrimeraDimension; i++){  //50 --> 70
        puts("Dame una plabra:");
        fflush(stdin);
        scanf("%s",cadena);
          //Se ajusta la memoria al tamaño de la palabra dada por el usuario.
        arreglo2dim[i]=realloc(arreglo2dim[i],sizeof(char)*strlen(cadena)+1);
        strcpy(arreglo2dim[i],cadena);
    }
    printf("La primera palabra es :%s\n",arreglo2dim[0]);
    printf("La tercera palabra es :%s\n",arreglo2dim[2]);
    free(cadena);
}
bool destruyeTabla(char** arreglo, int tamPrimeraDimension){
    int i;
    for(i=0; i<tamPrimeraDimension; i++)
        free(arreglo[i]);
    free(arreglo);
    return true;
}
void iniciaOperacion(int argc, char** argv){
    //Aqui hacer todo lo que hace la función main
    puts("Inicio de programa");
    //crearTablaDinamicaNxM(4000,4000);
    char** tablaX=crearTablaDinamica(10);
```

```
    //utilizando la tabla dinámica en una función:
    pruebaTablaDinamica(tablaX,10);
    destruyeTabla(tablaX, 10);
    puts("Termino el programa exitosamente");
}
```

Archivo.h

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
char** crearTablaDinamica(int tamPrimeraDimension);
void iniciaOperacion(int argc, char** argv);
void pruebaTablaDinamica(char**, int);
bool destruyeTabla(char** , int);
```