Arreglos dinámicos

martes, 30 de noviembre de 2021 07:12 a. m.

void free(void* ptr): Esta función requiere un apuntador devuelto por malloc, calloc o realloc debido a que esta función libera la memoria solicitada.

Definir un arreglo dinámico de tam NxM ---> Matriz --> crearArregloNxM

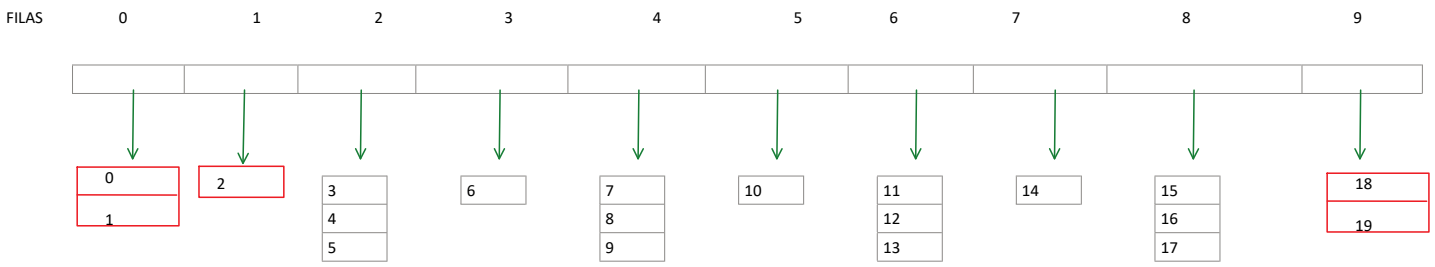
Columnas (M)

0,0 int *	0,1	0,2	...	0,M
1,0 int*				
2,0 int*				
... int*				
N,0 int*				N,M

Filas (N)

int arregloMatriz[N][M];

Generar la siguiente estructura para el almacenamiento de 20 valores enteros



Ejemplo:

archivo.h

```
#include <stdio.h>
#include <stdlib.h>
void crearArregloNxM(int tamPrimeraDim, int tamSegundaDim);
int** crearArregloDin(int tamFilas);
void iniciaOperacion(int argc, char** argv);
```

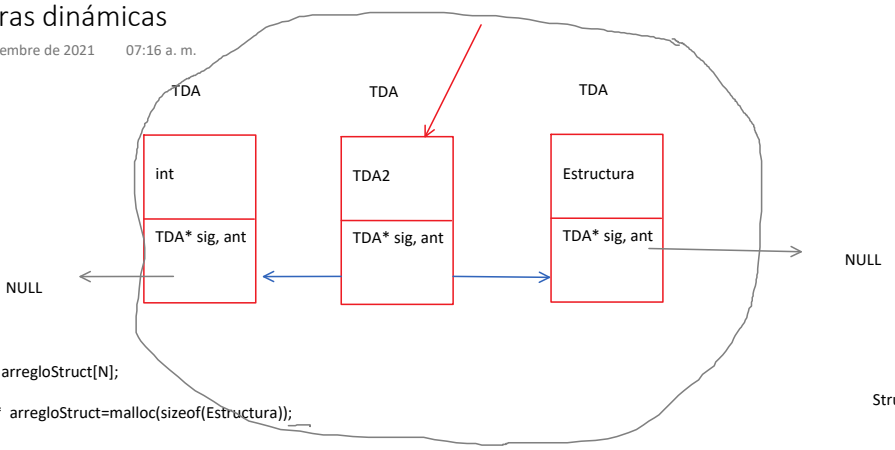
archivo.c

```
#include "arregloD.h"
/*
 Esta función genera una matriz de N filas (tamPrimeraDim) con M columnas cada
fila (tamSegundaDim)
 @param tamPrimeraDim Valor del tamaño de la primera dimensión de la matriz (N
filas)
 @param tamSegundaDim Valor del tamaño de la segunda dimensión de la matriz (M
columnas)
*/
void crearArregloNxM(int tamPrimeraDim, int tamSegundaDim){
    //Se reserva memoria dinámica para la primera dimensión
    int** arregloDosDim=(int**)malloc(sizeof(int*)*tamPrimeraDim);
    int indice, fila, columna;
    if(arregloDosDim==NULL){
        puts("Se termino la memoria, no se pueden reservar filas para la
matriz");
        exit(1);
    }else{
        //Se reserva memoria para cada fila, se le asignan M
columnas(tamSegundaDim)
        for(indice=0; indice<tamPrimeraDim; indice++){
            arregloDosDim[indice]=(int*)malloc(sizeof(int)*tamSegundaDim);
            if(arregloDosDim[indice]==NULL){
                puts("Se termino la memoria, no se pueden reservar las columnas
para la matriz");
                exit(1);
            }
        }
    }
    //Llenando la matriz
    for(fila=0; fila<tamPrimeraDim; fila++){  //Recorre filas
        for(columna=0; columna <tamSegundaDim; columna++){ //Recorre cada columna
            arregloDosDim[fila][columna]=fila+columna;
        }
    }
    //Imprimiendo la matriz
    for(fila=0; fila<tamPrimeraDim; fila++){  //Recorre filas
        for(columna=0; columna <tamSegundaDim; columna++){ //Recorre cada
columna
            printf("El valor en la fila[%d], columna[%d] es %d.
\n",fila,columna,arregloDosDim[fila][columna]);
        }
    }

    //Liberación de memoria dinámica: Liberando columnas
```

```
    for(indice=0; indice<tamPrimeraDim; indice++){
        free(arregloDosDim[indice]);
    }
    //Liberando la memoria de las filas
    free(arregloDosDim);
}
int** crearArregloDin(int tamFilas){
    int i;
    int** arregloEstructura=(int**) calloc(tamFilas,sizeof(int*));
     if(arregloEstructura==NULL){
        puts("Se termino la memoria, no se pueden reservar filas para la
matriz");
        exit(1);
    }
    for(i=0; i<tamFilas; i++){
        if(i==0 || i==9){
            arregloEstructura[i]= (int*) calloc(2, sizeof(int));
        } else if(){
        } else if(){

        }
    }
}
void iniciaOperacion(int argc, char** argv){
    puts("Iniciando el programa");
    //Operaciones de la función main.
    crearArregloNxM(40000,40000);
    puts("Fin del programa");
}
```

Estructuras dinámicas

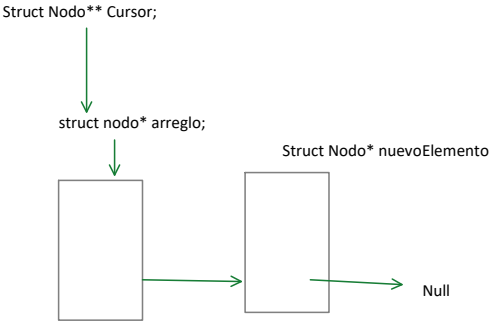
viernes, 3 de diciembre de 2021 07:16 a. m.

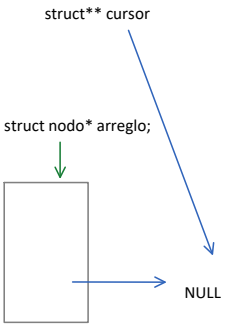


Estructura arregloStruct[N];

Estructura* arregloStruct=malloc(sizeof(Estructura));

Ejemplo:





Ejemplo:

Archivo .h

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

typedef struct{
    int entero;
    char caracter;
    char* arregloD;
    int arrayInt[10];
} Dato;
//TDA
struct Nodo{
    Dato dato;
    struct Nodo* siguienteNodo;
};

struct Nodo* creaNodo(Dato datoUsuario);
void agregarNodoAlArreglo(struct Nodo* arreglo, Dato datoUsuario);
void imprimirArregloDatos(struct Nodo* arreglo);
void destruirArreglo(struct Nodo* arreglo);
void iniciaOperacion(int argc, char** argv);
```

Archivo. c

```c
#include "tdaD.h"
struct Nodo* creaNodo(Dato datoUsuario){
```

```
    struct Nodo* nodo=(struct Nodo*)malloc(sizeof(struct Nodo));
    if(nodo==NULL){
        puts("Fin del programa, se termino la memoria");
        exit(1);
    }
    nodo->dato=datoUsuario;
    (*nodo).dato.arregloD=(char*) malloc(sizeof(char)*10);
    if(nodo->dato.arregloD==NULL){
        puts("Fin del programa, se termino la memoria");
        exit(1);
    }
    nodo->siguienteNodo=NULL;
    return nodo;
}
void agregarNodoAlArreglo(struct Nodo* arreglo, Dato datoUsuario){
    struct Nodo* nodoNuevo=creaNodo(datoUsuario);
    struct Nodo** cursor=&arreglo;
    //Recorriendo con un doble apuntador todos los elementos dinámicos del
arreglo
    while((*cursor)->siguienteNodo!=NULL){
        cursor=&((*cursor)->siguienteNodo);
    }
    (*cursor)->siguienteNodo=nodoNuevo;
}
void imprimirArregloDatos(struct Nodo* arreglo){
    struct Nodo** cursor=&arreglo;
    //Recorriendo con un doble apuntador todos los elementos dinámicos del
arreglo
    while((*cursor)!=NULL){
        puts("Información del usuario en el nodo:");
        printf("El valor entero es: %d\n", (*cursor)->dato.entero);
        printf("El valor caracter es: %c\n", (*cursor)->dato.caracter);
        cursor=&((*cursor)->siguienteNodo);
    }
}
void destruirArreglo(struct Nodo* arreglo){
    struct Nodo* aux=arreglo;
    while(aux->siguienteNodo!=NULL){
        arreglo=arreglo->siguienteNodo;
        free(aux);
        aux=arreglo;
    }
}
void iniciaOperacion(int argc, char** argv){
    puts("Inicia programa");
    Dato datoPrueba;
    datoPrueba.caracter='a';
    datoPrueba.entero=10;

    struct Nodo* arreglo=creaNodo(datoPrueba);
    datoPrueba.caracter='g';
    datoPrueba.entero=56;
    agregarNodoAlArreglo(arreglo,datoPrueba);
    datoPrueba.caracter='t';
    datoPrueba.entero=99;
    agregarNodoAlArreglo(arreglo,datoPrueba);

    imprimirArregloDatos(arreglo);
    puts("Fin del programa");
}
```