Memoria Dinámica (Heap Memory)

miércoles, 24 de noviembre de 2021 07:24 a. m.

Las funciones para gestionar la memoria dinámica en C, se encuentran en la directiva <stdlib.h> y son las siguientes:

void* malloc(size_t size): Esta función permite reservar tantos bytes consecutivos de memoria como indica su único parámetro. Devuelve la primera dirección de memoria del bloque reservada. Nota: cada vez que se llame a malloc hay que dejar de forma explícita una conversión del tipo de dato void* al que se requiera.

void* calloc(size_t nmembers, size_t size): Reserva espacio para tantos elementos como lo indica su parámetro "nmembers", y cada uno de ellos con un tamaño en bytes como lo indica su segundo parámetro "size". En pocas palabras, se reserva (nmembers * size) bytes consecutivos en memoria. Al igual que malloc, esta función devuelve la primera dirección de la memoria del bloque reservado. Esta función inicializa el bloque en 0's.

void realloc(void* ptr, size_t size): Esta función permite redimensionar una porción de memoria que anteriormente haya sido reservada. El primer parámetro es la primera dirección de memoria del bloque previamente reservado. El segundo parámetro es el nuevo tamaño del bloque (hay que tener cuidado con los decrecimientos). Se devuelve la nueva dirección de memoria (la primera del nuevo bloque reservado). Los datos siempre se conservan en los incrementos, más no en los decrementos.

Nota: Para reservar memoria se debe saber exactamente el número de bytes que ocupa cualquier estructura de datos. Se utilizará la función sizeof(), la cual nos ayudará a conocer la memoria utilizada por los tipos de datos.

Nota2: Toda vez que se llame a malloc, calloc o realloc hay que validar con una condicional que el apuntador devuelto no sea NULL, ya que en caso de que sea NULL es un indicador de que se terminó la memoria.

Ejemplo:

```c
#include <stdio.h>
#include <stdlib.h>
#define TAM 10000
typedef struct{   //48 Bytes
    int a;
    int b;
    int arreglo[TAM];
} Celda;

Celda* celdasGlobales;

void crearCeldasVacias(){
    //Celda* celdasLocales=(Celda*)malloc(sizeof(Celda)*2);
    Celda* celdasLocales=(Celda*)calloc(2,sizeof(Celda));
    if(celdasLocales==NULL){
        puts("Se termino la memoria para celdasLocales");
        exit(1);
```

```
    }
    Celda c1;
    celdasLocales[1]=c1;
    celdasGlobales[1]=c1;
    //Redimensionamientos -> incremento de memoria
    celdasLocales=(Celda*)realloc(celdasLocales,sizeof(Celda)*100);
    if(celdasLocales==NULL){
        puts("Se termino la memoria para celdasLocales, no se pudo
incrementar");
        exit(1);
    }
}
void main(int argc, char** argv){
    puts("Inicializando arreglo de celdas");
    //celdasGlobales=(Celda*)malloc(sizeof(Celda)*2);
    celdasGlobales=(Celda*)calloc(2,sizeof(Celda));
    if(celdasGlobales==NULL){
        puts("Se termino la memoria para celdasGlobales");
        exit(1);
    }
    crearCeldasVacias();
    puts("Finalizando con el arreglo de celdas");
}
```