UNIVERSIDAD INTERAMERICANA DE PANAMÁ

Facultad de Ingeniería, Arquitectura y Diseño

**ESTRUCTURA DE DATOS II**

TAREA INICIAL

Estudiante:

Chantal De Gracia ​ 8-971-1853

Docente:

Leonardo Esqueda

Fecha:

18-06-2022

**Buscar las diferencias entre arreglos y las matrices (con su ejemplo)**

Las matrices y arreglos pueden ser descritos como vectores multidimensionales. Al igual que un vector, únicamente pueden contener datos de un sólo tipo, pero además de largo, tienen más dimensiones.

En un sentido estricto, las matrices son un caso especial de un array, que se distingue por tener específicamente dos dimensiones, un “largo”" y un “alto”. Las matrices son, por lo tanto, una estructura con forma rectangular, con renglones y columnas.

Como las matrices son usadas de manera regular en matemáticas y estadística, es una estructura de datos de uso común.

Los arreglos, por su parte, pueden tener un número arbitrario de dimensiones. Pueden ser cubos, hipercubos y otras formas. Su uso no es muy común en R, aunque a veces es deseable contar con objetos n-dimensionales para manipular datos. Como los arreglos tienen la restricción de que todos sus datos deben ser del mismo tipo, no importando en cuántas dimensiones se encuentren, esto limita sus usos prácticos.

En general, es preferible usar listas en lugar de arreglos, una estructura de datos que además tienen ciertas ventajas que veremos más adelante.

**EJEMPLO MATRIZ:**

#include "iostream"

#include "stdio.h"

#include "string"

using namespace std;

int main()

{

string libros[5][2];

cout << "Por favor ingrese la siguiente información de los Libros: \n";

string titulo ,autor;

for(int i = 0; i < 5; i++)

{

cout << "\n\*\*\*\*\*\*\* Libro " << i + 1 << "\*\*\*\*\*\*\*\*:\n";

cout << "Titulo: ";

getline(cin,titulo);

cout << "Autor: ";

getline(cin,autor);

libros[i][0] = titulo;

libros[i][1] = autor;

}

system("pause");

return 0;}

**EJEMPLO ARREGLO:**

//Leer datos y guardarlos en el arreglo

for(i=0; i<=9; i++){

printf("\nEscriba el valor %d", i);

scanf("%f", &grupo[i]);

}

**Las diferencias y dónde aplicar una pila o una cola (con su ejemplo)**

En Ciencias de la computación, la pila y la cola son dos tipos de datos abstractos que son estructuras de datos simples que utilizan punteros para representar conjuntos dinámicos. Sin embargo, se puede notar una diferencia entre ellos en función de sus implementaciones. Las operaciones básicas de inserción y eliminación de elementos son compatibles tanto con la pila como con la cola. La diferencia principal entre la pila y la cola es que una apilar implementos Política de último en entrar, primero en salir o LIFO, mientras que una cola implementos Política de First In First Out o FIFO.

**EJEMPLO PILA:**

Este es un programa muy simple que escribe en orden inverso una sucesión de números leídos desde el teclado:

Pila p;

int numero;

printf ("ingresa números positivos, termina con –1\n");

scanf(“%d”,&numero);

while (numero!=-1)

{

poner(p,numero);

scanf(“%d”,&numero);

}

while (!vacia(p)) print (“%d ”,sacar(p))

**EJEMPLO COLA:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAXIMA\_LONGITUD\_CADENA 1000

struct NodoDeLista

{

char nombre[MAXIMA\_LONGITUD\_CADENA];

// Aquí podrían ir más datos...

struct NodoDeLista \*siguiente;

};

void agregarNodoALista(struct NodoDeLista \*\*nodo, char \*nombre)

{

struct NodoDeLista \*nuevoNodo = malloc(sizeof(struct NodoDeLista));

strcpy(nuevoNodo->nombre, nombre);

nuevoNodo->siguiente = NULL;

if ((\*nodo) == NULL)

{

\*nodo = nuevoNodo;

}

else

{

agregarNodoALista(&(\*nodo)->siguiente, nombre);

}

}

void imprimirLista(struct NodoDeLista \*nodo)

{

while (nodo != NULL)

{

printf("Encontramos un nombre en la lista: '%s'\n", nodo->nombre);

nodo = nodo->siguiente;

}

}

int main()

{

// https://parzibyte.me/blog

struct NodoDeLista \*apuntadorLista = NULL;

agregarNodoALista(&apuntadorLista, "Luis");

agregarNodoALista(&apuntadorLista, "Parzibyte");

agregarNodoALista(&apuntadorLista, "Link");

agregarNodoALista(&apuntadorLista, "Shovel Knight");

imprimirLista(apuntadorLista);

}

**Que se necesita para realizar una estructura de lista.**

Una lista es una estructura dinámica de datos que contiene una colección de elementos homogéneos (del mismo tipo) de manera que se establece entre ellos un orden. Es decir, cada elemento, menos el primero, tiene un predecesor, y cada elemento, menos el último, tiene un sucesor.

Tanto las estructuras vistas en la sección anterior (pilas) como las que veremos en la siguiente (colas) son tipos de listas.

Podemos distinguir, atendiendo a la organización de los nodos, entre:

* Listas simplemente enlazadas: cada nodo tiene un campo que apunta al siguiente nodo.
* Listas doblemente enlazadas: cada nodo dispone de un puntero que apunta al siguiente nodo, y otro que apunta al nodo anterior.

Otra distinción puede ser:

* Listas lineales: son listas que tienen un comienzo y un final.
* Listas circulares: en estas listas el último elemento apunta al primero, por lo tanto podríamos estar recorriéndolas siempre, ya que no tienen final.