Clase 2

20/03/2023

Más ejercitación en lenguaje C

0.0) Determinar la salida del programa sin ejecutar. Explicar linea a linea que sucede.

Resolución:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int *p,*q; // declaración de dos punteros a enteros p y q
    p=(int *) malloc(sizeof(int)); // se reserva memoria dinámicamente para p (tamaño de un entero)
    *p=5; // se asigna el valor 5 al puntero p
    q=p;  // q apunta a la misma dirección que p
    *q=2; // se asigna el valor 2 al puntero q
    printf("p apunta a la direccion %X, que contiene %d\n,",p,*p);// se muestran en pantalla el valor y la posición de memoria apuntada por p
    printf("q apunta a la direccion %X, que contiene %d\n",q,*q); // se muestran en pantalla el valor y la posición de memoria apuntada por q
    q=(int *) malloc(sizeof(int)); // se reserva memoria dinámicamente para q (tamaño de un entero)
    *q=8; // se asigna el valor 8 al puntero q
    printf("\n ahora vemos despues de las nuevas asignaciones\n");
    printf("p apunta a la direccion %X que contiene %d\n",p,*p); // se muestran en pantalla el valor y la posición de memoria apuntada por p
    printf("q apunta a la direccion %X que contiene %d\n",q,*q); // se muestran en pantalla el valor y la posición de memoria apuntada por q
    free(p); // se libera la memoria asignada a p
    free(q); // se libera la memoria asignada a q
    return 0;
}
```

💡 ***Explicación codeGPT:***

El código comienza por incluir las bibliotecas estándares de C que se van a utilizar: stdio.h y stdlib.h. A continuación se define la función principal main() que devuelve un entero. Dentro de la función principal se declaran dos punteros a enteros, "p" y "q". Se reserva memoria dinámicamente para el puntero "p" y se le asigna el valor 5. El puntero "q" se asigna a la misma dirección de memoria que "p". Luego se le asigna el valor 2 al puntero "q". Los valores y las direcciones de ambos punteros se imprimen en la pantalla mediante la función printf.

Después, se reserva memoria dinámicamente para el puntero "q" y se le asigna el valor 8. Los valores y direcciones de ambos punteros se vuelven a imprimir en la pantalla. Finalmente, se libera la memoria reservada para ambos punteros utilizando la función free() y se devuelve 0 como resultado de la función main().

En resumen, el código crea y utiliza dos punteros para reservar memoria dinámicamente y asignar valores a dicha memoria, después muestra los valores y direcciones de memoria de los punteros y por último libera la memoria reservada dinámicamente para los punteros.

01.A) Dado el siguiente vector, utilizar un puntero y recorrerlo con él.

```
int lista[] ={24,30,15,45,34};
```

Resolución:

```
#include <stdio.h>
int main()
{
    int lista[] ={24,30,15,45,34};
    int *p=lista;
    printf("lista: ");
    for (int i = 0; i < 5; i++)
    {
        printf("%d, ",*p);
        p++;
    }
    return 0;
}
```

01.B) Hacer un programa que pida al usuario la cantidad de elementos que necesita almacenar para ella realizar una función cargar. Lugo mostrar ese vector, pasandole el titulo de " vector Desordenado "- Luego ordenar ese vector y mostrarlo con el titulo "Vector ordenado"., por último liberar ese espacio.

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h> // para usar la funcion rand y que no aparezcan errores

void cargar(int ,int *);
void mostrar(int ,int *);
void ordenar(int ,int *);

int main ()
{
    srand(time(NULL)); // funcion para generar numeros al azar (semilla aleatoria)
    int n,*vector;
    printf("ingresar cantidad de elementos del vector: ");
    scanf("%d",&n);
    vector=(int *) malloc(n*sizeof(int));
    cargar(n,vector);
    printf("vector desordenado: ");
    mostrar(n,vector);
    printf("vector ordenado: ");
    ordenar(n,vector);
    mostrar(n,vector);
    free(vector);
    return 0;
}

void cargar(int n, int *vector)
{
    for (int i = 0; i < n; i++)
    {
        *vector = rand()%(10)+1;
        vector++;
    }
}

void mostrar(int n, int *vector)
{
    for (int i = 0; i < n; i++)
    {
        printf("%d, ",*vector);
        vector++;
    }
}

void ordenar(int n, int *vector)
{
    int i, j, menor, temp;

    for (i = 0; i < n-1; i++)
    {
        menor = i;

        for (j = i+1; j < n; j++)
        {
            if (*(vector+j) < *(vector+menor))
                menor = j;
        }

        if (menor != i)
        {
            temp = *(vector+i);
            *(vector+i) = *(vector+menor);
            *(vector+menor) = temp;
        }
    }
}
```

```
Explicación codeGPT:
La línea `vector = (int *) malloc(n*sizeof(int));` se utiliza para reservar
memoria dinámicamente para un array de enteros, el cual tiene un tamaño
determinado por `n` y se almacena en la dirección de memoria apuntada
por el puntero `vector`.
La función `malloc` asigna un bloque de memoria del tamaño especificado
por el parámetro `n*sizeof(int)`. El tipo de datos `int*`
se utiliza para que sepa que se devolverá un puntero a un arrays de enteros.
La cantidad de memoria a reservar se especifica
multiplicando `n` por `sizeof(int)` (tamaño en bytes de un `int`).
```

01.C) Realizar un programa que, dada la captura de una cadena, se pueda determinar lo siguiente:

1) El largo de la cadena, realizando una funcion llamada cuentaCaracteres

2) La cantidad de palabras ingresadas, realizando una función llamada cuentaPal

3) La cantidad de letras y números que posee ese texto ingresado, realizando una función cuentaLetras

4) Convertir la cadena ingresada en mayuscula, en esa conversión dejarla en otra cadena, realizarlo en una función Mayu

5) Mostrar la cadena en forma invertida (recursividad)

6) Dada una cadena que ingrese devolver la última palabra de esa cadena en otro vector, Hacer una función llamada última

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define n 30

void cuentaCaracteres(char [n]);
void cuentaPal(char [n]);
void cuentaLetras(char [n]);
void mayu(char [n]);
void invertirCadena(char [], int);
void ultima(char [n]);
int main()
{
    char cad [n];
    printf("ingresar un texto: ");
    gets(cad);
    cuentaCaracteres(cad);
    cuentaPal(cad);
    cuentaLetras(cad);
    mayu(cad);
    printf("\nCadena invertida: ");
    invertirCadena(cad, strlen(cad));
    ultima(cad);
    return 0;
}

void cuentaCaracteres(char cad [n])
{
    int i=0;
    while (cad[i]!='\0')
    {
        i++;
    }
    printf("largo de la cadena: %d",i);
}

void cuentaPal(char cad [n])
{
    int i=0,palabras=0;
    while (cad[i]!='\0')
    {
        if ((cad[i] >= 'a' && cad[i] <= 'z') || (cad[i] >= 'A' && cad[i] <= 'Z') || (cad[i] >= '0' && cad[i] <= '9' ))
        {
            if (i == 0 || cad[i-1] == ' ')
            {
                palabras++;
            }
        }
        i++;
    }
    printf("\ncantidad de palabras: %d",palabras);
}

void cuentaLetras(char cad [n]) {
    int letras = 0;
    int numeros = 0;
    int i = 0;

    while (cad[i] != '\0') {
        if ((cad[i] >= 'a' && cad[i] <= 'z') || (cad[i] >= 'A' && cad[i] <= 'Z')) {
            letras++;
        }
        else if (cad[i] >= '0' && cad[i] <= '9') {
            numeros++;
        }
        i++;
    }

    printf("\nCantidad de letras: %d", letras);
    printf("\nCantidad de numeros: %d", numeros);
}

void mayu(char cad[n])
{
    char cadMayu[n];
    for (int i = 0; i <= strlen(cad); i++)
    {
        cadMayu[i]=toupper(cad[i]);
    }
    printf("\nCadena en mayuscula: %s", cadMayu);
}

void invertirCadena(char cad[], int len){
    if(len==0){
        return;
    }
    else{
        invertirCadena(cad+1, len-1);
        printf("%c", cad[0]);
    }
}
```

```
void ultima(char cad[n])
{
    char ultPalabra[n];
    int len = strlen(cad);
    int i = len - 1;

    while (i >= 0 && cad[i] != ' ')
    {
        i--;
    }

    if (i < 0)
    {
        strcpy(ultPalabra, cad);
    }
    else
    {
        strcpy(ultPalabra, &cad[i+1]);
    }

    printf("\nUltima palabra: %s", ultPalabra);
}
```