Tarea #6

Inyerman Alexander Xap Chin 20210097

Escuela de Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos de Guatemala.

A. Resumen

Recreamos los códigos dados por el ingeniero y los corrimos para comprobar su funcionamiento.

El programa de calculadora básica corre sin ningún problema y tras las pruebas funcionales vemos que funciona como debería.

Sin embargo, al probar el segundo programa dedicado a los pesos de los estudiantes nos dimos cuenta que aunque corre sin problema al momento de hacerle pruebas funcionales no funciona como debería tras hacer nuestra investigación de varios métodos de ordenamiento llegamos a la creación del programa que bautizamos como ordenamiento con un código diferente que cumple con los requerimientos.

B. Objetivos

- Crear un código de una calculadora básica que luego de dar los resultados te vuelva a llevar al menú después de 5 seg.
- Hacer que el programa se limpie por si mismo.
- Crear un programa de ordenamiento de valor ascendente.

C. Marco Teórico

Algoritmos de Ordenación:

• Ordenación o clasificación es el proceso de reordenar un conjunto de objetos en un orden específico.

• El propósito de la ordenación es facilitar la búsqueda de elementos en el conjunto ordenado.

• Existen muchos algoritmos de ordenación, siendo la diferencia entre ellos la eficiencia en tiempo de ejecución.

• Los métodos de ordenación se pueden clasificar en dos categorías: ordenación de ficheros o externo y ordenación de arrays o interno.

• Aquí trataremos sólo del ordenamiento interno.

El problema del ordenamiento puede establecerse mediante la siguiente acción: Dados los elementos:

Ordenar consiste en permutar esos elementos en un orden: tal que dada una función de ordenamiento f:

• Normalmente, la función de ordenamiento se guarda como un componente explícito (campo) de cada item (elemento). El valor de ese campo se llama la llave del item. • Un método de ordenamiento es estable si el orden relativo de elementos con igual llave permanece inalterado por el proceso de ordenamiento.

Se entiende que los métodos de ordenamiento buscan un uso eficiente de la memoria por lo que las permutaciones de elementos se hará in situ, es decir, usando el mismo

array original. • En lo que sigue se considera que el array a ordenar consiste de números reales: float a[MAX]; siendo MAX el número máximo de elementos del array. • El orden de los elementos después de la ordenación se considera ascendente.

Ordenación por burbuja

Es un método caracterizado por la comparación e intercambio de pares de elementos hasta que todos los elementos estén ordenados. • En cada iteración se coloca el elemento más pequeño (orden ascendente) en su lugar correcto, cambiándose además la posición de los demás elementos del array. • La complejidad del algoritmo es O(n2).

Ordenación por burbuja: código

```
/*************************************************************************\
* Programa: ordena_bubble.c                                              *
* Descripción: Programa que ordena un array mediante el metodo de burbuja*
* Autor: Pedro Corcuera                                                  *
* Revisión: 1.0 2/02/2008                                                *
\*************************************************************************/
#include <stdio.h>

main()
{
 float t, a[]={10.0, 8.0, 5.5, 3.4, 3.2, 2.5, 2.2, 1.5};
 int i, j, n, ninterc = 0;

 n = sizeof(a)/sizeof(float);

 for(i = 0; i < n; i++)
  for(j = n-1; j >= i; j--)
   if (a[j] > a[j+1])      /* orden ascendente */
   {
    t = a[j];
    a[j] = a[j+1];
    a[j+1] = t;
    ninterc++;             /* Contador de intercambios */
   }
 for(i = 0;i < n;i++)
   printf("%d  %10.2f \n",i,a[i]);
 printf("Numero de intercambios = %d \n",ninterc);
}
```

Ordenación por shakesort (agitación)

Ordenación por inserción

• Método usado para ordenar una mano de naipes.

• Los elementos están divididos conceptualmente en

una secuencia destino y una secuencia fuente .

• En cada paso, comenzando con i=2 e incrementando i en uno, el elemento i-ésimo de la secuencia fuente se toma y se transfiere a la secuencia destino insertándolo en el lugar adecuado.

• Es decir, en el i-ésimo paso insertamos el i-ésimo elemento a[i] en su lugar correcto entre a[1], a[2],…., a[i-1], que fueron colocados en orden previamente.• Método usado para ordenar una mano de naipes.

• Los elementos están divididos conceptualmente en una secuencia destino y una secuencia fuente .

• En cada paso, comenzando con i=2 e incrementando i en uno, el elemento i-ésimo de la secuencia fuente se toma y se transfiere a la secuencia destino insertándolo en el lugar adecuado.

• Es decir, en el i-ésimo paso insertamos el i-ésimo elemento a[i] en su lugar correcto entre a[1], a[2],…., a[i-1], que fueron colocados en orden previamente está ordenado el algoritmo es O(n).

```
/*************************************************************************\
* Programa: skahesort.c                                                  *
* Descripción: Programa que ordena un array mediante el metodo agitacion *
* Autor: Pedro Corcuera                                                  *
* Revisión: 1.0 2/02/2008                                                *
\*************************************************************************/
#include <stdio.h>

main()
{
 float t,
/*  a[]={10.0, 8.0, 5.5, 3.4, 3.2, 2.5, 2.2, 1.5};*/
 a[]={1.5, 2.2, 2.5, 3.2, 3.4, 5.5, 8.0, 10.0};
 int i,j,k,l,r,n,ninterc=0;

 n = sizeof(a)/sizeof (float);
 l = 1;   /* indice para ordenar en sentido directo */
 r = n-1; /* indice para ordenar en sentido inverso */
 k = n-1; /* indice para registrar el intercambio */
 do
 {
  for(j = r; j >= l; j--)
   if (a[j-1] > a[j])
   {
    t = a[j-1]; a[j-1] = a[j]; a[j] = t;
    k = j;
    ninterc++;
   }
```

```
 l = k+1;
  for(j = l; j <= r; j++)
   if (a[j-1] > a[j])
   {
    t = a[j-1];
    a[j-1] = a[j];
    a[j] = t;
    k = j;
    ninterc++;
   }
  r = k-1;
 }
 while (l < r);

/* Impresión del array ordenado */
 for(i=0;i<n;i++)
   printf("%d  %10.2f \n",i,a[i]);
 printf("Numero de intercambios = %d \n",ninterc);
}
```

Ordenación por inserción

• Método usado para ordenar una mano de naipes. • Los elementos están divididos conceptualmente en una secuencia destino y una secuencia fuente . • En cada paso, comenzando con i=2 e incrementando i en uno, el elemento i-ésimo de la secuencia fuente se toma y se transfiere a la secuencia destino insertándolo en el lugar adecuado. • Es decir, en el i-ésimo paso insertamos el i-ésimo elemento a[i] en su lugar correcto entre a[1], a[2],…., a[i-1], que fueron colocados en orden previamente.

• Este algoritmo puede mejorarse fácilmente si vemos que la secuencia destino está ordenada, por lo que usamos una búsqueda binaria para determinar el punto de inserción.

• La complejidad del algoritmo es O(n2).

```
/*************************************************************************\
* Programa: ord_insercion.c                                               *
* Descripción: Programa que ordena un array mediante el metodo insercion  *
* Autor: Pedro Corcuera                                                   *
* Revisión: 1.0 2/02/2008                                                 *
\*************************************************************************/
#include <stdio.h>

main()
{
 float t,a[]={10.0,8.0,5.5,3.4,3.2,2.5,2.2,1.5};
 int i,j,n;

 n = sizeof(a)/sizeof (float);
 for(i = 1; i < n; i++)
 {
  j = i-1;
  t = a[i];
  while (j >= 0 && t < a[j])
  {
    a[j+1] = a[j];
    j = j-1;
  }
  a[j+1] = t;
 }
 for(i = 0; i < n; i++)
   printf("%d  %10.2f \n",i,a[i]);
}
```

Ordenación por selección

• En éste método, en el i-ésimo paso seleccionamos el elemento con la llave de menor valor, entre a[i],…, a[n] y lo intercambiamos con a[i].

• Como resultado, después de i pasadas, el i-ésimo elemento menor ocupará a[1],…, a[i] en el lugar ordenado.

• La complejidad del algoritmo es O(n2).

```
/*************************************************************************\
* Programa: ord_seleccion.c                                               *
* Descripción: Programa que ordena un array mediante el metodo seleccion  *
* Autor: Pedro Corcuera                                                   *
* Revisión: 1.0 2/02/2008                                                 *
\*************************************************************************/
#include <stdio.h>

main()
{
 float t,a[]={10.0,8.0,5.5,3.4,3.2,2.5,2.2,1.5};
 int i,j,k,n;

 n = sizeof(a)/sizeof (float);
 for(i = 0; i < n-1; i++)
 {
  k = i; t = a[i];
  for (j = i+1; j < n; j++)
  {
    if (a[j] < t)
    {  t = a[j]; k = j;
    }
    a[k] = a[i]; a[i] = t;
  }
 }
 for(i = 0; i < n; i++)
   printf("%d  %10.2f \n",i,a[i]);
}
```

D. Marco Practico

Empezamos por recrear los códigos dados por el ingeniero luego los compilamos y corrimos para ver que funcionaran como era debido.

Al momento de hacerlo nos percatamos que el programa de la calculadora estaba funcionando correctamente sin embargo el programa dedicado al peso de los estudiantes no estaba funcionando como debería luego de verificar el código. No pudimos dar con la falla por lo que procedimos a investigar los distintos tipos de códigos de ordenamiento que existen para verificar nuestro código.

Luego de la investigación se llegó a la creación de pesos2 y posteriormente a ordenamiento.c

que fue el resultado final, dando fin a nuestro proyecto.

E. Código

Código de Calculadora Básica parte 1/2

```c
C: > Codigos769 > C calculadorabasica.c > main()
#include<stdio.h> //incluye la biblioteca para las funciones basicas de entrada y salida
#include<stdlib.h> //incluye la biblioteca para la funcion system ("clear");
#include<unistd.h> //incluye la biblioteca para la funcion sleep

/*Este programa genera una calculadora basica.*/

int main(){
    int op,uno,dos;
    do{
        system("cls");
        printf("---Calculadora---\n");
        printf("
        printf("     (char [11])"2) Restar\n"
        printf("2) Restar\n");
        printf("3) Multiplicar\n");
        printf("4) Dividir\n");
        printf("5) Salir\n");
        scanf("%d",&op);
        switch(op){
        case 1:
            printf("\tSumar\n");
            printf("Introduzca los numeros a sumar separados por comas\n");
            scanf("%d,%d",&uno,&dos);
            printf("%d+%d=%d\n",uno,dos,(uno+dos));
            sleep(5);// Pausa el programa durante 5 segundos
            break;
        case 2:
            printf("\tRestar\n");
            printf("Introduzca los numeros a restar separados por comas\n");
            scanf("%d,%d",&uno,&dos);
            printf("%d-%d=%d\n",uno,dos,(uno-dos));
            sleep(5);// Pausa el programa durante 5 segundos
            break;
        case 3:
            printf("\tMultiplicar\n");
            printf("Introduzaca los numeros a multiplicar separados por comas\n");
            scanf("%d,%d",&uno,&dos);
```

Código de Calculadora Básica parte 2/2

```c
            scanf("%d,%d",&uno,&dos);
            printf("%d*%d=%d\n",uno,dos,(uno*dos));
            sleep(5);// Pausa el programa durante 5 segundos
            break;
        case 4:
            printf("\tDividir\n");
            printf("Introduzaca los numeros a Dividir separados por comas\n");
            scanf("%d,%d",&uno,&dos);
            printf("%d/%d=%.2lf\n",uno,dos,((double)uno/dos));
            sleep(5);// Pausa el programa durante 5 segundos
            break;
        case 5:
            printf("\tSalir\n");
            sleep(5);// Pausa el programa durante 5 segundos
            break;
        default:
            printf("\tOpcion invalida.\n");
            sleep(5);// Pausa el programa durante 5 segundos
        }
    }while (op !=5);
    system("clear");
    return 0;
}
```

Código de pesos de estudiantes en c

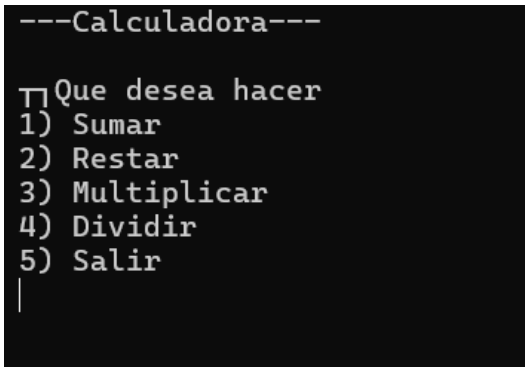
```c
C: > Codigos769 > C pesos2.c > main()
#include<stdio.h>
#include<stdlib.h>
void main()
{
    float*pesos,temp;
    int i,j,nest;
    printf("Cuantos estudiantes son?:");
    scanf("%d",&nest);
    pesos=(float*)malloc(nest*sizeof(float));
    if (pesos==NULL)
    {
        printf("Insuficiente Espacio de Memoria\n");
        exit(-1);//Salir del Programa
    }
    for (i=0;i<nest;i++)
    {
        printf("Peso del Estudiante[%d]:",i+1);
        scanf("%f",(pesos+i));
    }
    printf("\n***ARRAY ORIGINAL***\n");
    for (i=0;i<nest;i++)
    printf("Peso[%d]:%.1f\n",i+1,*(pesos+i));
    for(i=0;i<nest;i++){
    for(j=0;j<(nest-1);j++)
    if(pesos[i]>pesos[j+1])
        {
            temp=pesos[i];
            pesos[j]=pesos[j+1];
            pesos[j+1]=temp;
        }
    }
    printf("\n***ARRAY ORDENADO EN FORMATO ASCENDENTE***\n");
    for(i=0;i<nest;i++)
    printf("Peso[%d]:%.1f\n",i+1,*(pesos+i));
}
```

Codigo de ordenamiento en c

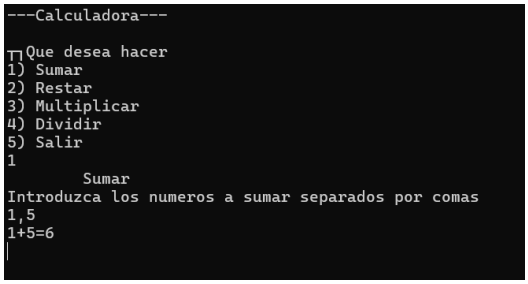
```c
C: > Codigos769 > C ordenamiento.c > main()
#include <stdio.h>

int main()
{
  int n, arreglo[5], c, d, t, flag = 0;

  printf("Introduce el numero de estudiantes\n");
  scanf("%d", &n);

  printf("Introduce %d pesos enteros\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &arreglo[c]);

  for (c = 1 ; c <= n - 1; c++) {
    t = arreglo[c];

    for (d = c - 1 ; d >= 0; d--) {
      if (arreglo[d] > t) {
        arreglo[d+1] = arreglo[d];
        flag = 1;
      }
      else
        break;
    }
    if (flag)
      arreglo[d+1] = t;
  }

  printf("La lista ordenada ascendentemente es:\n");

  for (c = 0; c <= n - 1; c++) {
    printf("%d\n", arreglo[c]);
  }

  return 0;
}
```

F. Resultados

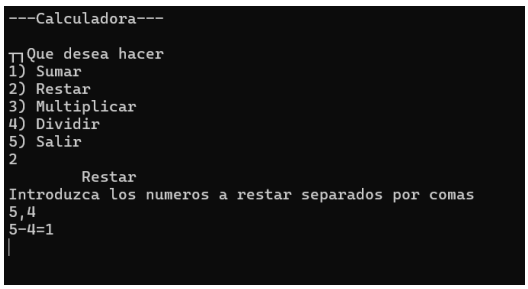
Código de calculadora básica corriendo 1/6



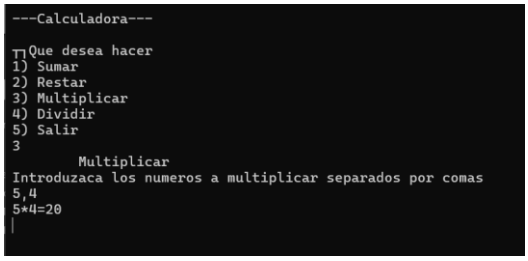
Código de calculadora básica corriendo 2/6 (Sumando)



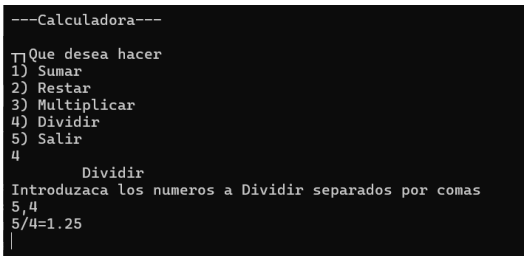
Código de calculadora básica corriendo 3/6 (Restando)



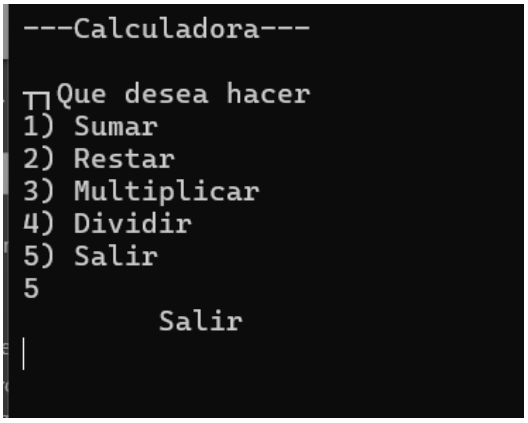
Código de calculadora básica corriendo 4/6 (Multiplicando)



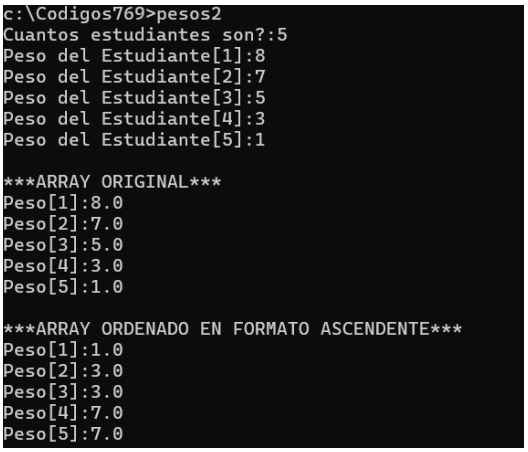
Código de calculadora básica corriendo 5/6 (Dividiendo)



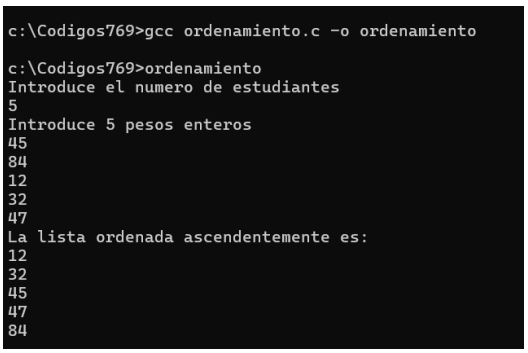
Código de calculadora básica corriendo 6/6 (Saliendo)



Código De pesos de estudiantes corriendo



Código de ordenamiento corriendo



G. Conclusiones

- Creamos un código de una calculadora básica que nos lleva de vuelta al menú luego de 5 seg después de operar.
- Logramos hacer que el programa se limpie por si mismo utilizando el comando system(“clear”).
- Creamos un programa de ordenamiento de valor ascendente.

H. Referencias

https://architecnologia.es/programacion-c-algoritmos-ordenamiento-busqueda-apuntadores

https://personales.unican.es/corcuerp/progcomp/slides/C_8n.pdf