LA *PROGRAMACIÓN* NO TIENE LÍMITES, CON UN POCO DE *FE*

Developeralta

Sistemas de E.D.O.

Versión 1.0

Manual Técnico

Software de aplicación

Evalúa sistemas de ecuaciones diferenciales ordinarias

Peralta Mario
developeralta@gmail.com
01 de julio de 2018

ÍNDICE

1. DESCRIPCIÓN DEL PROBLEMA

Una ecuación diferencial es una igualdad matemática que relaciona una función con sus derivadas y un sistema de ecuaciones diferenciales es un conjunto de una o más ecuaciones en las que aparecen una o más funciones incógnitas, pero todas ellas dependiendo de una sola variable independiente. Éstas juegan un rol primordial en diversas disciplinas, incluyendo la ingeniería, física, química, economía y biología. Un método genérico para la aproximación de soluciones de sistemas de ecuaciones diferenciales ordinarias con valores iniciales es denominado ***Runge Kutta***.

Por lo tanto, se requiere un software que implemente dicho método de 4° orden para cualquier sistema de ecuaciones diferenciales ordinarias con valores iniciales.

1.1 REQUERIMIENTOS BÁSICOS

- ✓ El usuario deberá introducir un sistema de E.D.O. explicitas $\frac{du_1}{dt} = f(t, u_1, u_2, \dots, u_m)$, … , $\frac{du_m}{dt} = f(t, u_1, u_2, \dots, u_m)$, los valores iniciales, los límites de la variable t, el número de iteraciones a realizar y el programa deberá crear un archivo con la información correspondiente. Por ejemplo si se introduce la ecuación ***f1(t,a,b)=b***, ***f2(t,a,b)=5*t*a-2*b+e^(-2*t)***, ***a(0)=1***, ***b(0)=1***, ***[0,1]*** y ***n=10***, el programa deberá crear el siguiente archivo de texto

t	***a(t)***	***b(t)***
0	***1***	***1***
0.1	***1.0958***	***0.9255***
0.2	***1.1868***	***0.9037***
0.3	***1.2783***	***0.9358***
0.4	***1.3758***	***1.0231***
0.5	***1.4849***	***1.6787***
0.6	***1.6114***	***1.2738***
0.7	***1.7619***	***1.6482***
0.8	***1.9437***	***2.0015***

0.9	***2.1653***	***2.4490***
1.0	***2.4373***	***3.0121***

1.2 RESTRICCIONES BÁSICAS

- ✓ Las E.D.O deberán ser explicitas.
- ✓ Las variables podrán ser:
  - ***A…Z***
  - ***a…z***
- ✓ Las variables son insensibles a las mayúsculas y minúsculas.
- ✓ Los operadores válidos serán:
  - **+**
  - **–**
  - ***
  - **/**
  - **^**
  - **/**
- ✓ Las operaciones serán binarias.
- ✓ La función no puede incluir espacios, porque el programa no los ignora.
- ✓ El intervalo deberá pertenecer a la ecuación.
- ✓ Método de Runge-Kutta sistemas de E.D.O.
  - El número de ecuaciones del sistema debe ser igual a las variables independientes de la función.

2. PROPUESTA DE SOLUCIÓN

2.1 PLANTEAMIENTO DE LA SOLUCIÓN

El algoritmo es el siguiente:

1. Leer el número de variables independientes de la función.
2. Leer las E.D.O.
3. Leer el intervalo.
4. Leer los valores iniciales de las ecuaciones.
5. Leer el número de iteraciones.
6. Obtener el valor de h con la siguiente fórmula $h = \frac{b-a}{n}$.
7. Escribir los valores iniciales en el archivo.
8. Aplicar los pasos 9) a 10) para cada iteración.
9. Calcular la siguiente iteración con las siguientes fórmulas:

$$w_{i+1,j} = w_i + \frac{h}{6}\left(k_{1,j} + 2k_{2,j} + 2k_{3,j} + k_{4,j}\right)$$

$$k_{j,1} = f_j\left(t_i, w_{i,1}, \dots, w_{i,m}\right)$$

$$k_{j,2} = f_j\left(t_i + \frac{h}{2}, w_{i,1} + \frac{1}{2}k_{t,1}h, \dots, w_{i,m} + \frac{1}{2}k_{t,1}h\right)$$

$$k_{j,3} = f_j\left(t_i + \frac{h}{2}, w_{i,1} + \frac{1}{2}k_{t,2}h\,, \dots, w_{i,m} + \frac{1}{2}k_{t,2}h\right)$$

$$k_{j,4} = f_j\left(t_i + h, w_{i,1} + k_{t,3}h, \dots, w_{i,m} + k_{t,3}h\right)$$

$$Donde: i\ es\ el\ número\ de\ particiones, j\ y\ t\ es\ el\ número\ de\ ecuaciones.$$

10. Escribir la $t_i, w_{i+1,1}, \dots, w_{i+1,m}$ en el archivo. Volver al paso 7).
11. Mostrar la información del archivo en pantalla.

2.2 DEFINICIÓN DE ENTRADAS Y SALIDAS

- ✓ Entradas
  - Sistema de ecuaciones diferenciales ordinarias: Arreglo de cadenas
  - Valores iniciales: Arreglo de reales
  - Intervalo: Real
  - Número de iteraciones: Entero
- ✓ Salidas
  - Aproximación de las soluciones: Archivo de texto

2.3 LIMITANTES

- Las posibles soluciones de la E.D.O. tiene un cierto grado de error.
- El número de variables independientes debe ser igual o menor a 25.

3. REPRESENTACIÓN DE LA SOLUCIÓN (Pseudocódigo)

```
PROGRAMA Sistema de E.D.O. 1.0
/*
AUTOR: Developeralta.
FECHA: 01 de Julio del 2018.
PROPÓSITO: Un método genérico para la aproximación de soluciones de sistemas de ecuaciones
diferenciales ordinarias con valores iniciales es denominado Runge Kutta.
Por lo tanto, se requiere un software que implemente dicho método de 4° orden para cualquier sistema
de ecuaciones diferenciales ordinarias con valores iniciales.
LIMITANTES:
•       Las posibles soluciones de la E.D.O. tiene un cierto grado de error.
•       El número de variables independientes debe ser igual o menor a 25.
*/
CONSTANTES
| MaxEcua=25
TIPOS DE DATOS
| TArregloC=ARREGLO [1..MaxEcua] DE CADENA
```

```
| TArregloR=ARREGLO [1..MaxEcua] DE REAL
| TKutta=REGISTRO
| | PFaz : TArregloC
| | Faz : TArregloR
| | Tope : ENTERO
| | ta : REAL
| | tb : REAL
| | n : ENTERO
| FIN_REGISTRO
| TMatrizk=ARREGLO [1..4,1..MaxEcua] DE REAL
| TArchivo : ARCHIVO DE TEXTO
VARIABLES
| /*VARIABLES DE ENTRADA*/
| Datos : TKutta
| NArchivo : CADENA
INICIO
| REPITE
| | COMIENZA
| | | LIMPIAR PANTALLA
| | | MostrarEncabezado("Sistema de E.D.O. 1.0")
| | | LeerEntero(Datos.Tope,"¿Cuántas ecuaciones tiene el sistema?")
| | | SI(Datos.Tope>1 Y Datos.Tope<=MaxEcua)ENTONCES
| | | | COMIENZA
| | | | | LeerDatos(Datos)
| | | | | ESCRIBE "Escriba el nombre del nuevo archivo:"
| | | | | LEE(NArchivo)
| | | | | NArchivo<-NArchivo+".txt"
| | | | | SI(CrearArchivo(NArchivo))ENTONCES
| | | | | | COMIENZA
| | | | | | | SI(ConvertirPostfija(Datos))ENTONCES
| | | | | | | | Enter("ERROR 03: Memoria RAM sin espacio.")
| | | | | | | SINO
| | | | | | | | COMIENZA
| | | | | | | | | ESCRIBE "Procesando información, por favor espere..."
| | | | | | | | | SI(SERungeKutta4(Datos,NArchivo))ENTONCES
| | | | | | | | | | COMIENZA
| | | | | | | | | | | Enter("ERROR 03: Memoria RAM sin espacio.")
| | | | | | | | | | | SI(EliminarArchivo(NArchivo)=FALSO)ENTONCES
| | | | | | | | | | | | Enter("ERROR 04: Inconveniente al eliminar archivo.")
| | | | | | | | | | | | FIN_SI
| | | | | | | | | | | TERMINA
| | | | | | | | | | SINO
| | | | | | | | | | | COMIENZA
| | | | | | | | | | | | SI(MostrarArchivo(NArchivo))ENTONCES
| | | | | | | | | | | | | Enter("ERROR 05: Inconveniente al mostrar el archivo.")
| | | | | | | | | | | | SINO
| | | | | | | | | | | | | ESCRIBE "Información almacenada correctamente en ",NArchivo,"."
| | | | | | | | | | | | FIN_SI
| | | | | | | | | | | TERMINA
| | | | | | | | | | FIN_SI
| | | | | | | | | TERMINA
| | | | | | | | FIN_SI
| | | | | | | TERMINA
| | | | | SINO
| | | | | | Enter("ERROR 02: Memoria secundaría sin espacio.")
| | | | | FIN_SI
| | | | TERMINA
| | | SINO
| | | | Enter("ERROR 01: Número de ecuaciones fuera de rango (2<=x<=25).")
| | | FIN_SI
| | TERMINA
| HASTA(MenuSalida()=2)
| Enter("Cerrando programa...")
FIN

/* PROCEDIMIENTOS Y FUNCIONES */
FUNCIÓN CrearArchivo(VAL NombreFisico : CADENA) : BOOLEANO
/*
UTILIDAD: Crear un archivo de 'TArchivo'.
PRECONDICIÓN: Si el archivo existe, entonces éste perderá sus datos.
```

```
POSCONDICIÓN: Se devuelve verdadero si se crea el archivo y falso en caso contrario.
*/
VARIABLES
| Arch : TArchivo
| Creado : BOOLEANO
COMIENZA
| Creado<-FALSO
| ASIGNA(Arch,NombreFisico)
| CREA(Arch)
| SI(NO EXISTIÓ ERROR AL CREAR EL ARCHIVO)ENTONCES
| | COMIENZA
| | | CERRAR(Arch)
| | | Creado<-VERDADERO
| | TERMINA
| FIN_SI
| REGRESA(Creado)
TERMINA

PROCEDIMIENTO LeerDatos(REF Datos : TKutta)
/*
UTILIDAD: Leer los datos de las ecuaciones del teclado.
PRECONDICIÓN: Si 'Datos' tiene información está se perderá y 'Datos.Tope' tiene el número de
ecuaciones.
POSCONDICIÓN: 'Datos' tiene la información leída del teclado.
*/
VARIABLES
| IE : ENTERO
| CadenaA,Funcion : CADENA
COMIENZA
| LeerEntero(Datos.n,"¿Cuántas iteraciones quiere realizar?")
| LeerIntervalo(Datos.ta,Datos.tb)
| Funcion<-"F(t,a, ... ,"+CVars[Datos.Tope]+")="
| PARA(IE<-1,Datos.Tope,IE<-IE+1)
| | COMIENZA
| | | LeerFaz(Datos.PFaz[IE],Funcion,CDatos.Tope)
| | | CadenaA<-CVars[IE]+"("+CADENA(Datos.ta)+")="
| | | LeerReal(Datos.Faz[IE],CadenaA)
| | TERMINA
| FIN_PARA
TERMINA

PROCEDIMIENTO LeerIntervalo(REF ta : REAL,REF tb : REAL)
/*
UTILIDAD: Leer un intervalo del teclado.
PRECONDICIÓN: Ninguna.
POSCONDICIÓN: Se lee un intervalo válido del teclado. (ta>tb)
*/
VARIABLES
| Valido : BOOLEANO
COMIENZA
| Valido<-FALSO
| REPITE
| | COMIENZA
| | | LeerReal(ta,"Valor de a=")
| | | LeerReal(tb,"Valor de b=")
| | | SI(ta>=tb)ENTONCES
| | | | Enter("ERROR: Intervalo fuera de rango.")
| | | SINO
| | | | Valido<-VERDADERO
| | | FIN_SI
| | TERMINA
| HASTA(Valido)
TERMINA

FUNCIÓN ConvertirPostfija(REF Datos : TKutta) : BOOLEANO
/*
UTILIDAD: Convertir las funciones infijas a postfijas.
PRECONDICIÓN: 'Datos.Tope' tiene el número de ecuaciones y 'Datos.PFaz' tiene las ecuaciones
infijas.
```

```
POSCONDICIÓN: Se devuelve falso si no existió error entonces'Datos' tiene las funciones postfijas
y verdadero en caso contrario.
*/
VARIABLES
| IF : ENTERO
| ErrorM : BOOLEANO
COMIENZA
| ErrorM<-FALSO
| IF<-1
| REPITE
| | COMIENZA
| | | ErrorM<-PostfijaFaz(Datos.PFaz[IF],Datos.PFaz[IF])
| | | IF<-IF+1
| | TERMINA
| HASTA(IF>Datos.Tope O ErrorM)
| REGRESA(ErrorM)
TERMINA

FUNCIÓN SERungeKutta4(VAL CDatos : TKutta,VAL NArchivo : CADENA) : BOOLEANO
/*
UTILIDAD: Crear un archivo 'NArchivo' con las iteraciones del método Runge Kutta Orden 4.
PRECONDICIÓN: 'NArchivo' debe existir además de no contener registros y 'CDatos' tiene información
válida.
POSCONDICIÓN: Se devuelve falso sino existió error entonces 'NArchivo' tiene como registro las
iteraciones del método Runge Kutta Orden 4 y verdadero
en caso contrario.
*/
VARIABLES
| h : REAL
| Registro,Registro2 : CADENA
| Archivo : TArchivo
| ErrorM : BOOLEANO
| It,Ie : ENTERO
| K : TMatrizk
| TempFaz : ARREGLO [1..3] DE TArregloR
COMIENZA
| ErrorM<-FALSO
| ASIGNA(Archivo,NArchivo)
| ABRE Archivo PARA ESCRITURA
| ESCRIBE "*****SISTEMA DE E.D.O. 1.0*****" EN Archivo
| Registro<-CADENA(CDatos.ta)
| Registro2<-"t"
| PARA(It<-1,CDatos.Tope,It<-It+1)
| | COMIENZA
| | | Registro<-" "+CADENA(CDatos.Faz[It])
| | | Registro2<-" "+CVars[It]+"(t)"
| | TERMINA
| FIN_PARA
| ESCRIBE Registro2 EN Archivo
| ESCRIBE Registro EN Archivo
| h<-(CDatos.tb-CDatos.ta)/CDatos.n
| It<-1
| REPITE
| | COMIENZA
| | | Registro<-CADENA(CDatos.ta+h)
| | | Ie<-1
| | | REPITE //K1
| | | | COMIENZA
| | | | | ErrorM<-EvaluarFaz(CDatos.PFaz[Ie],CDatos.ta,CDatos.Faz,K[1,Ie])
| | | | | TempFaz[1,Ie]<-CDatos.Faz[Ie]+K[1,Ie]*h/2
| | | | | Ie<-Ie+1
| | | | TERMINA
| | | HASTA(Ie>CDatos.Tope O ErrorM)
| | | Ie<-1
| | | MIENTRAS(Ie<=CDatos.Tope Y ErrorM=FALSO)//K2
| | | | COMIENZA
| | | | | ErrorM<-EvaluarFaz(CDatos.PFaz[Ie],CDatos.ta+h/2,TempFaz[1],K[2,Ie])
| | | | | TempFaz[2,Ie]<-CDatos.Faz[Ie]+K[2,Ie]*h/2
| | | | | Ie<-Ie+1
| | | | TERMINA
```

```
| | | FIN_PARA
| | | Ie<-1
| | | MIENTRAS(Ie<=CDatos.Tope Y ErrorM=FALSO)//K3
| | | | COMIENZA
| | | | | ErrorM<-EvaluarFaz(CDatos.PFaz[Ie],CDatos.ta+h/2,TempFaz[2],K[3,Ie])
| | | | | TempFaz[3,Ie]<-CDatos.Faz[Ie]+K[3,Ie]*h
| | | | | Ie<-Ie+1
| | | | TERMINA
| | | FIN_PARA
| | | Ie<-1
| | | MIENTRAS(Ie<=CDatos.Tope Y ErrorM=FALSO)//K4
| | | | COMIENZA
| | | | | ErrorM<-EvaluarFaz(CDatos.PFaz[Ie],CDatos.ta+h,TempFaz[3],K[4,Ie])
| | | | | Ie<-Ie+1
| | | | TERMINA
| | | FIN_PARA
| | | SI(ErrorM=FALSO)ENTONCES
| | | | COMIENZA
| | | | | PARA(Ie<-1,CDatos.Tope,Ie<-Ie+1) //Icognitas de las ecuaciones.
| | | | | | COMIENZA
| | | | | | | CDatos.Faz[Ie]<-CDatos.Faz[Ie]+h*(K[1,Ie]+2*K[2,Ie]+2*K[3,Ie]+K[4,Ie])/6
| | | | | | | Registro<-" "+CADENA(CDatos.Faz[Ie])
| | | | | | TERMINA
| | | | | FIN_PARA
| | | | | ESCRIBE Registro EN Archivo
| | | | | CDatos.ta<-CDatos.ta+h
| | | | | It<-It+1
| | | | TERMINA
| | | FIN_SI
| | TERMINA
| HASTA(It>CDatos.n O ErrorM)
| CERRAR(Archivo)
| REGRESA(ErrorM)
TERMINA

FUNCIÓN EliminarArchivo(VAL NombreFisico : CADENA) : BOOLEANO
/*
UTILIDAD: Eliminar un archivo del disco de almacenamiento.
PRECONDICIÓN: Existe un archivo con 'NombreFisico' como nombre.
POSCONDICIÓN: Se devuelve verdadero si se elimina el archivo y falso en caso contrario.
*/
VARIABLES
| Eliminado : BOOLEANO
| Archivo : TArchivo
COMIENZA
| Eliminado<-FALSO
| ASIGNA(Archivo,NombreFisico)
| ELIMINA(Archivo)
| SI(NO EXISTIÓ ERROR AL ELIMINAR ARCHIVO)ENTONCES
| | Eliminado<-VERDADERO
| FIN_SI
| REGRESA(Eliminado)
TERMINA

FUNCIÓN MostrarArchivo(VAL NombreAO : CADENA) : BOOLEANO
/*
UTILIDAD: Mostrar los registros del archivo 'NombreAO'.
PRECONDICIÓN: 'NombreAO' existe.
POSCONDICIÓN: Se devuelve falso si se muestran en pantalla todos los registros del archivo 'NombreAO'
y verdadero en caso de error.
*/
VARIABLES
| Arch : TArchivo
| Registro : CADENA
| Error : BOOLEANO
COMIENZA
| Error<-FALSO
| ASIGNA(Arch,NombreAO)
| ABRE Arch PARA LECTURA
| MIENTRAS(NO(FDA(Arch))) Y (Error=FALSO)
```

```
| | COMIENZA
| | | LEE Registro DE Arch
| | | SI(NO EXISTIÓ ERROR AL LEER REGISTRO)ENTONCES
| | | | ESCRIBE Registro
| | | SINO
| | | | Error<-VERDADERO
| | | FIN_SI
| | TERMINA
| FIN_MIENTRAS
| CERRAR(Arch)
| REGRESA(Error)
TERMINA
```

4. PRUEBAS DE SOFTWARE

4.1 PRUEBAS UNITARIAS

El archivo con extensión PDF contiene los experimentos de cada módulo. Ctrl + Clic para visualizarlo.

4.2 PRUEBAS DE INTREGACIÓN

El archivo con extensión PDF contiene la prueba del programa principal. Ctrl + Clic para visualizarlo.

4.3 PLAN DE PRUEBAS

No. de Prueba	Juegos de Datos		Resultados Esperados			Observaciones
	Variables de Entrada	Variables				
			t	a(t)	b(t)	
	n=5		0.00000000	0.00000000	0.00000000	
	[0,0.5]		0.10000000	0.53825520	0.31962623	
1	***F(t,a,… , b)=6+3*b-4*a***		0.20000000	0.96849871	0.56878214	
	F(t,a,… , b)=0		0.30000000	1.31071902	0.76073308	
	F(t,a,… , b)=3.6+1.6*b-2.4*a		0.40000000	1.58126521	0.90632054	
	F(t,a,… , b)=0		0.50000000	1.79350745	1.01440231	

2	***n=10*** ***[0,1]*** ***F(t,a,… , b)=b+0*** ***F(t,a,… , b)=1*** ***F(t,a,… , b)=5*t*a-2*b+(2.718281828)^(0-2*t)*** ***F(t,a,… , b)=1***		t 0.00000000 0.10000000 0.20000000 0.30000000 0.40000000 0.50000000 0.60000000 0.70000000 0.80000000 0.90000000 1.00000000	a(t) 1.00000000 1.09584553 1.18686173 1.27838426 1.37586635 1.48492095 1.61146620 1.76196185 1.94373378 2.16539414 2.43737627	b(t) 1.00000000 0.92555893 0.90372944 0.93586814 1.02317549 1.16781780 1.37388380 1.64826347 2.00154840 2.44906327 3.01215305	

5. REQUERIMIENTOS MINIMOS DE HARDWARE Y SOFTWARE PARA EL DESARROLLO Y MANTENIMIENTO

5.1 HARDWARE

- ✓ Procesador Intel® Pentium® 233 MHz o superior.
- ✓ 64 MB RAM (se recomiendan 128 MB).
- ✓ 400 MB de espacio disponible en disco duro.
- ✓ Unidad grabadora de CD-ROM (Para instalar los programas requeridos).
- ✓ Monitor con resolución 800 x 600 p o superior.
- ✓ Mouse.

5.2 SOFTWARE

- ✓ Microsoft® Windows® XP o superior.
- ✓ Code::Blocks Versión 17.12.
- ✓ Notepad++ Versión 7.5.6.
- ✓ Word 2016.
- ✓ Excel 2016.

6. IMPLEMENTACIÓN (Codificación Comentada)

```
//PROGRAMA Sistema de E.D.O. 1.0
/*
AUTOR: Developeralta.
FECHA: 01 de Julio del 2018.
PROPÓSITO: Un método genérico para la aproximación de soluciones de sistemas de ecuaciones diferenciales ordinarias con valores iniciales es denominado Runge Kutta.
Por lo tanto, se requiere un software que implemente dicho método de 4° orden para cualquier sistema de ecuaciones diferenciales ordinarias con valores iniciales.
LIMITANTES:
•       Las posibles soluciones de la E.D.O. tiene un cierto grado de error.
•       El número de variables independientes debe ser igual o menor a 25.
*/
//LIBRERÍAS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "SistemasEDO.h"
//TIPOS DE DATOS
 typedef CADENA TArregloC[MaxEcua];
 typedef struct PTKutta{
  TArregloC PFaz;
  TArregloR Faz;
  int Tope;
  double ta;
  double tb;
  int n;
 }TKutta;
 typedef double TMatrizk[4][MaxEcua];
 typedef FILE *TArchivo; //ARCHIVO DE TEXTO
//PROTOTIPOS DE FUNCIONES Y PROCEDIMIENTOS
BOOLEANO CrearArchivo(const CADENA NombreFisico);
void LeerDatos(TKutta *Datos);
```

```
void LeerIntervalo(double *ta,double *tb);
BOOLEANO ConvertirPostfija(TKutta *Datos);
BOOLEANO SERungeKutta4(TKutta CDatos,const CADENA NArchivo);
BOOLEANO RungeKutta4(TKutta CDatos,CADENA NArchivo);
BOOLEANO EliminarArchivo(const CADENA NombreFisico);
BOOLEANO MostrarArchivo(const CADENA NombreAO);

int main(){
//VARIABLES
 /*VARIABLES DE ENTRADA*/
 TKutta Datos;
 //VARIABLES AUXILIARES
 CADENA CadenaAux;
 CADENA NArchivo;
 do
  {
   system("cls");
   MostrarEncabezado("Sistema de E.D.O. 1.0");
   sprintf(CadenaAux,"%cCu%cntas ecuaciones tiene el sistema?",168,160);
   LeerEntero(&Datos.Tope,CadenaAux);
   if(Datos.Tope>1 && Datos.Tope<=MaxEcua)
    {
     LeerDatos(&Datos);
     printf("Escriba el nombre del nuevo archivo: ");
     fflush(stdin);
     gets(NArchivo);
     strcat(NArchivo,".txt");
     if(CrearArchivo(NArchivo))
      {
       if(ConvertirPostfija(&Datos))
        Enter("ERROR 03: Memoria RAM sin espacio.");
       else
        {
         printf("Procesando informaci%cn, por favor espere...\n",162);
         if(SERungeKutta4(Datos,NArchivo))
          {
           Enter("ERROR 03: Memoria RAM sin espacio.");
           if(EliminarArchivo(NArchivo)==FALSO)
            Enter("ERROR 04: Inconveniente al eliminar archivo.");
          }
         else
          {
           if(MostrarArchivo(NArchivo))
            Enter("ERROR 05: Inconveniente al mostrar el archivo.");
           else
            printf("Informaci%cn almacenada correctamente en %s.\n",162,NArchivo);
          }
        }
      }
     else
      Enter("ERROR 02: Memoria secundaría sin espacio.");
    }
   else
    {
     sprintf(CadenaAux,"ERROR 01: N%cmero de ecuaciones fuera de rango (2<=x<=25).",163);
     Enter(CadenaAux);
    }
  }
 while(MenuSalida()!=2);
 Enter("Cerrando programa...");
 return 0;
}

/* PROCEDIMIENTOS Y FUNCIONES */
BOOLEANO CrearArchivo(const CADENA NombreFisico)
/*
UTILIDAD: Crear un archivo de 'TArchivo'.
PRECONDICIÓN: Si el archivo existe, entonces éste perderá sus datos.
POSCONDICIÓN: Se devuelve verdadero si se crea el archivo y falso en caso contrario.
*/
```

```
{
//VARIABLES
 TArchivo Arch;
 BOOLEANO Creado;
 Creado=FALSO;
 //ASIGNA(Arch,NombreFisico) Se asigna al crear el archivo.
 Arch=fopen(NombreFisico,"wb");
 if(Arch!=NULL)
  {
   fclose(Arch);
   Creado=VERDADERO;
  }
 return(Creado);
}

void LeerDatos(TKutta *Datos)
/*
UTILIDAD: Leer los datos de las ecuaciones del teclado.
PRECONDICIÓN: Si 'Datos' tiene información está se perderá y 'Datos.Tope' tiene el número de
ecuaciones.
POSCONDICIÓN: 'Datos' tiene la información leída del teclado.
*/
{
//VARIABLES
 int IE;
 CADENA CadenaA,Funcion;
 sprintf(CadenaA,"%cCu%cntas iteraciones quiere realizar?",168,160);
 LeerEntero(&Datos->n,CadenaA);
 LeerIntervalo(&Datos->ta,&Datos->tb);
 sprintf(Funcion,"F(t,a, ... ,%c)=",CVars[Datos->Tope-1]);
 //Se cambia porque en C, los arreglos inician en cero
 for(IE=0;IE<Datos->Tope;IE=IE+1)
  {
   LeerFaz(Datos->PFaz[IE],Funcion,Datos->Tope-1);
   sprintf(CadenaA,"%c(%lf)=",CVars[IE],Datos->ta);
   LeerReal(&Datos->Faz[IE],CadenaA);
  }
}

void LeerIntervalo(double *ta,double *tb)
/*
UTILIDAD: Leer un intervalo del teclado.
PRECONDICIÓN: Ninguna.
POSCONDICIÓN: Se lee un intervalo válido del teclado. (ta>tb)
*/
{
//VARIABLES
 BOOLEANO Valido;
 Valido=FALSO;
 do
  {
   LeerReal(ta,"Valor de a=");
   LeerReal(tb,"Valor de b=");
   if(ta>=tb)
    Enter("ERROR: Intervalo fuera de rango.");
   else
    Valido=VERDADERO;
  }
 while(Valido==FALSO); //Se cambia la condición porque en C, es mientras la condición es verdadera.
}

BOOLEANO ConvertirPostfija(TKutta *Datos)
/*
UTILIDAD: Convertir las funciones infijas a postfijas.
PRECONDICIÓN: 'Datos.Tope' tiene el número de ecuaciones y 'Datos.PFaz' tiene las ecuaciones
infijas.
POSCONDICIÓN: Se devuelve falso si no existió error entonces'Datos' tiene las funciones postfijas
y verdadero en caso contrario.
*/
{
```

```
//VARIABLES
 int IF;
 BOOLEANO ErrorM;
 CADENA CadenaAux; //Se utiliza para convertir a expresión postfija
 ErrorM=FALSO;
 //Se cambia porque en C, los arreglos inician en cero
 IF=0;
 do
  {
   ErrorM=PostfijaFaz(Datos->PFaz[IF],CadenaAux);
   strcpy(Datos->PFaz[IF],CadenaAux);
   IF=IF+1;
  }
 while(IF<Datos->Tope && ErrorM==FALSO); //Se cambia la condición porque en C, es mientras la
condición es verdadera.
 return(ErrorM);
}

BOOLEANO SERungeKutta4(TKutta CDatos,const CADENA NArchivo)
/*
UTILIDAD: Crear un archivo 'NArchivo' con las iteraciones del método Runge Kutta Orden 4.
PRECONDICIÓN: 'NArchivo' debe existir además de no contener registros y 'CDatos' tiene información
válida.
POSCONDICIÓN: Se devuelve falso sino existió error entonces 'NArchivo' tiene como registro las
iteraciones del método Runge Kutta Orden 4 y verdadero
en caso contrario.
*/
{
//VARIABLES
 double h;
 CADENA Registro,Registro2;
 TArchivo Archivo;
 BOOLEANO ErrorM;
 int It,Ie;
 TMatrizk K;
 TArregloR TempFaz[3];
 ErrorM=FALSO;
 //ASIGNA(Archivo,NArchivo) Se asigna al abrir el archivo.
 Archivo=fopen(NArchivo,"r+t");
 fprintf(Archivo,"%s\n","*****SISTEMA DE E.D.O. 1.0*****");
 sprintf(Registro,"%.8lf",CDatos.ta);
 sprintf(Registro2,"%c",'t');
 //Se cambia porque en C, los arreglos inician en cero
 for(It=0;It<CDatos.Tope;It=It+1)
  {
   sprintf(Registro,"%s\t\t%.8lf",Registro,CDatos.Faz[It]);
   sprintf(Registro2,"%s\t\t\t%c(t)",Registro2,CVars[It]);
  }
 fprintf(Archivo,"%s\n",Registro2);
 fprintf(Archivo,"%s\n",Registro);
 h=(CDatos.tb-CDatos.ta)/CDatos.n;
 //Se cambia porque en C, los arreglos inician en cero
 It=0;
 do
  {
   sprintf(Registro,"%.8lf",CDatos.ta+h);
   //Se cambia porque en C, mientras la condición es verdadera se cumple el ciclo.
   Ie=0;
   do //K1
    {
     ErrorM=EvaluarFaz(CDatos.PFaz[Ie],CDatos.ta,CDatos.Faz,&K[0][Ie]);
     TempFaz[0][Ie]=CDatos.Faz[Ie]+K[0][Ie]*h/2;
     Ie=Ie+1;
    }
   while(Ie<CDatos.Tope && ErrorM==FALSO);
   //Se cambia porque en C, mientras la condición es verdadera se cumple el ciclo.
   Ie=0;
   while(Ie<CDatos.Tope && ErrorM==FALSO)//K2
    {
     ErrorM=EvaluarFaz(CDatos.PFaz[Ie],CDatos.ta+h/2,TempFaz[0],&K[1][Ie]);
```

```
     TempFaz[1][Ie]=CDatos.Faz[Ie]+K[1][Ie]*h/2;
     Ie=Ie+1;
    }
   //Se cambia porque en C, mientras la condición es verdadera se cumple el ciclo.
   Ie=0;
   while(Ie<CDatos.Tope && ErrorM==FALSO)//K3
    {
     ErrorM=EvaluarFaz(CDatos.PFaz[Ie],CDatos.ta+h/2,TempFaz[1],&K[2][Ie]);
     TempFaz[2][Ie]=CDatos.Faz[Ie]+K[2][Ie]*h;
     Ie=Ie+1;
    }
   //Se cambia porque en C, mientras la condición es verdadera se cumple el ciclo.
   Ie=0;
   while(Ie<CDatos.Tope && ErrorM==FALSO)//K4
    {
     ErrorM=EvaluarFaz(CDatos.PFaz[Ie],CDatos.ta+h,TempFaz[2],&K[3][Ie]);
     Ie=Ie+1;
    }
   if(ErrorM==FALSO)
    {
     //Se cambia porque en C, mientras la condición es verdadera se cumple el ciclo.
     for(Ie=0;Ie<CDatos.Tope;Ie=Ie+1) //Icognitas de las ecuaciones.
      {
       CDatos.Faz[Ie]=CDatos.Faz[Ie]+h*(K[0][Ie]+2*K[1][Ie]+2*K[2][Ie]+K[3][Ie])/6;
       sprintf(Registro,"%s\t\t%.8lf",Registro,CDatos.Faz[Ie]);
      }
     fprintf(Archivo,"%s\n",Registro);
     CDatos.ta=CDatos.ta+h;
     It=It+1;
    }
  }
 while(It<CDatos.n && ErrorM==FALSO); //Se cambia porque en C, mientras la condición es verdadera
se cumple el ciclo.
 fclose(Archivo);
 return(ErrorM);
}

BOOLEANO EliminarArchivo(const CADENA NombreFisico)
/*
UTILIDAD: Eliminar un archivo del disco de almacenamiento.
PRECONDICIÓN: Existe un archivo con 'NombreFisico' como nombre.
POSCONDICIÓN: Se devuelve verdadero si se elimina el archivo y falso en caso contrario.
*/
{
//VARIABLES
 BOOLEANO Eliminado;
 //TArchivo Archivo; No se emplea porque para eliminar se utiliza el nombre físico.
 Eliminado=FALSO;
 //ASIGNA(Archivo,NombreFisico) Se asigna al eliminar el archivo.
 if(remove(NombreFisico)==0)
  Eliminado=VERDADERO;
 return(Eliminado);
}

BOOLEANO MostrarArchivo(const CADENA NombreAO)
/*
UTILIDAD: Mostrar los registros del archivo 'NombreAO'.
PRECONDICIÓN: 'NombreAO' existe.
POSCONDICIÓN: Se devuelve falso si se muestran en pantalla todos los registros del archivo 'NombreAO'
y verdadero en caso de error.
*/
{
//VARIABLES
 TArchivo Arch;
 CADENA Registro;
 BOOLEANO Error;
 Error=FALSO;
 //ASIGNA(Arch,NombreAO) //Se asigna al abri el archivo.
 Arch=fopen(NombreAO,"rb");
 while(!(feof(Arch)) && Error==FALSO)
```

```
  {
   if(fscanf(Arch,"%[^\n]\n",Registro))
    puts(Registro);
   else
    Error=VERDADERO;
  }
 fclose(Arch);
 return(Error);
}
```

7. RECOMENDACIONES

En futuras versiones, el programa puede optimizarse con base a las siguientes sugerencias:

- Mejorar la interfaz del usuario para incrementar la usabilidad.
- Implementar el método de Runge-Kutta de orden superior.
- Obtener el error absoluto de las aproximaciones.

8. BIBLIOGRAFÍA

Joyanes, L. (2003). ***Fundamentos de programación. Algoritmos, Estructuras de datos y Objetos:*** McGraw Hill. Tercera edición.

Chapra, S. (2003). ***Métodos Numéricos para Ingenieros:*** McGraw Hill. Quinta edición.