

Proyecto Fin de Carrera
Ingeniero Industrial

**Estudio y diseño numérico de un evaporador de doble
tubo con flujo bifásico utilizando mapas de flujo**

ANEXO A: Manual de empleo del programa evapora

Autor: Carlos Andrés Ribas
Director: Lluís Albert Bonals Muntada
Convocatoria: Enero 2004 (Plan 94)



Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona



ANEXO A: MANUAL DE EMPLEO DEL PROGRAMA “EVAPORA” _ 85

A.1.	Arranque del programa.....	85
A.2.	Confirmación de datos.....	86
A.3.	Pantalla de resultados	87
A.3.	Programa evapora	89

Anexo A: Manual de empleo del programa “evapora”

Para la simulación informática del evaporador se ha utilizado el lenguaje C++ y se ha compilado mediante el software Borland C++ 5.02. El programa recibe el nombre de evapora.

A.1. Arranque del programa

Es aconsejable arrancar el programa desde MS-DOS, ya que si se arranca desde el archivo ejecutable, es posible que tras la última iteración o presentación de resultados se cierre la ventana.

Tras escribir el nombre del ejecutable, aparece la pantalla de introducción de datos, que se rellenará según sean las variables de entrada empleadas y en las unidades requeridas y especificadas en la pantalla del programa.

```
***** PROYECTO: 'SIMULACION DE UN EVAPORADOR' *****  
  
***** Introduccion de datos *****  
  
Temperatura de entrada del agua? [en C] 25.0  
Presion de entrada del agua?[en kPa] 101.3  
velocidad de entrada del agua? [en m/s] 1.0  
Temperatura de entrada de los gases? [en C] (no recomendable mas de 1150 C) 700  
Presion de entrada de los gases?[en kPa] 101.3  
velocidad de entrada de los gases? [en m/s] vs1=50  
Longitud del evaporador ? [m] 4.0  
  
numero de divisiones? [max 199] 100
```

Fig. A.1. Pantalla de introducción de datos.



A.2. Confirmación de datos

Una vez se han introducido los datos, el programa realiza unos primeros cálculos para determinar las variables de entrada más significativas, los caudales y flujos básicos, y los números de Prandtl y de Reynolds.

Si se ha cometido algún error al escribir los datos de entrada, y se sospecha de algún valor incorrecto, es posible volver a la pantalla de introducción de datos. Si todo es correcto, el programa seguirá con la simulación.

```
*****Los datos introducidos son los siguientes:*****
**Geometria del Evaporador**
L=4.0 m          n=100.0 divisiones
_____ LADO ENVOLVENTE (gases de combustion) _____
T_s1=700.0 C      P_s1=101.3 kPa   V_s=50.0 m/s
m_s=0.092 kg/s      G_s=19.18 kg/m2s
Re_s=14411.57      Pr_s=0.78
----- LADO TUBO (agua) -----
T_I1=25.0 C          P_I1=101.3 kPa      V_I=1.0 m/s
m_I=0.028 kg/s = 101.79 l/h      G_I=996.75 kg/m2s
Re_I=6339.63      Pr_I=6.32
pulse '1' para cambiar datos o '0' para continuar
```

Fig. A.2. Pantalla de confirmación de datos.



A.3. Pantalla de resultados

Si las variables de entrada son confirmadas, el programa realiza la simulación del evaporador. Acabadas las iteraciones, el programa pregunta al usuario si quiere ver resultados. La primera pantalla de resultados (figura A.3.) presenta el dibujo esquemático de un intercambiador de doble-tubo, dividido en las zonas correspondientes a las regiones principales. Además presenta las longitudes de cada zona y las variables de salida más representativas.

Con este esquema el usuario se hace una idea general de estado del fluido en el interior del evaporador, conociendo la limitación geométrica del flujo en estado bifásico.

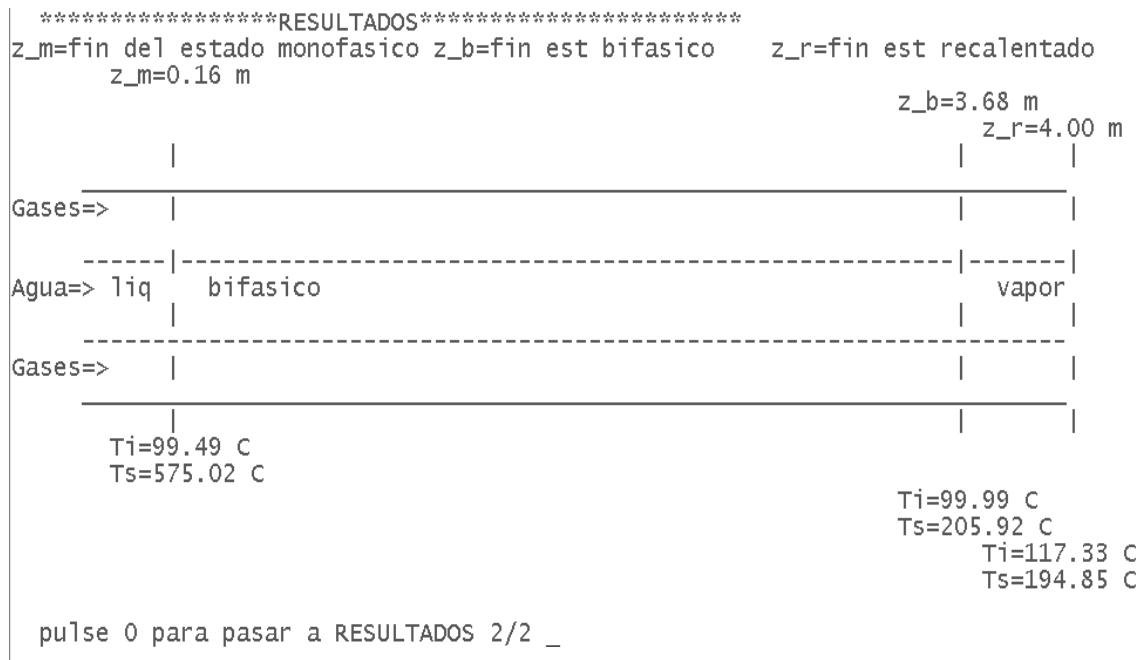


Fig. A.3. Pantalla de resultados: esquema del evaporador

La siguiente pantalla de resultados es más específica. Primeramente presenta ante el usuario la opción de elegir la evolución de las variables más importantes a lo largo del tubo (figura A.4.)



```
*****RESULTADOS      2/2*****
pulse '1' para ver temperaturas de liquido
pulse '2' para ver temperaturas gas
pulse '3' para ver temperaturas pared
pulse '4' para ver coef de conveccio interior
pulse '5' para ver fraccion volumetrica
pulse '6' para ver vector de titulos
pulse '7' para ver flowpattern
```

Fig. A.4. Pantalla de resultados: elección de variables

En esta pantalla se puede elegir una de las 7 opciones y los resultados aparecerán por pantalla. Cuando se presentan unos resultados existe la posibilidad de volver a la pantalla de elección de variables para elegir otra opción. Si no es así, apretando el '0' el programa finaliza.

```
2
620.00  605.44  590.16  575.02  572.14  563.50  552.09  540.47  528.93  517.56
506.39  495.44  484.71  474.21  463.93  453.89  444.07  436.70  429.51  422.49
415.64  408.96  402.44  396.09  389.90  383.86  377.97  372.24  366.65  361.20
355.90  350.73  345.70  340.80  336.03  331.38  326.86  322.46  318.18  314.02
309.96  306.02  302.19  298.46  294.83  291.31  287.88  284.55  281.32  278.17
275.12  272.15  269.27  266.47  263.75  261.11  258.55  256.07  253.66  251.32
249.05  246.85  244.72  242.65  240.65  238.71  236.83  235.01  233.24  231.54
229.89  228.29  226.75  225.26  223.81  222.42  221.08  219.78  218.52  217.32
216.15  215.03  213.95  212.91  211.90  210.94  210.02  209.13  208.27  207.45
206.67  205.92  204.53  203.19  201.88  200.62  199.39  198.20  197.05  195.93
194.85

pulse 1 para ver MAS resultados o 0 para salir
0
fin de resultados
```

Los resultados han quedado grabados en c:resultados.dat**
*****FIN DEL PROGRAMA*****

Fig. A.5. Pantalla de resultados: resultados por pantalla



De todas maneras, la evolución de dichas variables y de los regímenes de flujo a lo largo del doble tubo, quedan grabados en c:\ resultados.dat , pudiendo abrir posteriormente este archivo para un análisis más detallado de todo el proceso.

A.3. Programa evapora

A continuación se presenta el algoritmo del programa evapora, es decir, todo el programa en lenguaje C++ que ha servido para simular el evaporador de doble-tubo y realizar el presente proyecto. Las variables utilizadas, así como los nombres de funciones y acciones no tienen porque coincidir con las descritas a lo largo del proyecto, ya que el lenguaje C++ limita considerablemente el uso de signos, letras y números para su asignación.

A.3.1 Algoritmo del programa

```
/*PROGRAMA EN C++ DE CÁLCULO DE EVAPORADORES*/  
/*inclusiones*/  
#include <assert.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <math.h>  
#include <ctype.h>  
#include <conio.h>  
/*definición de constantes*/  
#define PI 3.14159265358979323846  
#define M 200  
/*definición de tipos estructurados*/  
typedef float taula[M];  
typedef struct{
```



```
taula tabla;
int ntablal;
}tuplanl;

typedef struct{
taula tabla;
int ntablac;
}tuplanc;

typedef struct{
float tempsi,tempssf;
float tempii,tempif;
float pressi,pressf;
float presii,presif;
float Hconv;
float titol;
}elemento;

typedef elemento telementos[M];

/*********************CABECERAS DE FUNCIONES*******************/



float LlegirReal (void);
int LlegirEnter(void);
void clrscr(void);
void presentarResultats(int n,float L,float l1,float l2,float l3,float Tg1,float Tw1,float Tg2,
float Tw2,float Tg3,float Tw3,float x_tit);
void presentarResultats2(tuplanl a,tuplanl b, tuplanc d,tuplanc e,tuplanl f,tuplanl g,tuplanl h);
void resultados( int n,float L,tuplanl a,tuplanl b, tuplanc d,tuplanc e,tuplanl f,tuplanl
g,tuplanl h);

/*cabecera propiedad gases*/
```



```
float CPS (float T);  
float LAMS (float T);  
float RHOS (float T);  
float VISCS (float T);  
float PRS (float T);  
/*cabecera propiedades agua liquida*/  
double CPIL (float T);  
double LAMIL (float T);  
float RHOIL (float T);  
double VISCIL (float T);  
double PRIL (float T);  
/*cabecera propiedades vapor de agua*/  
double LAMIV(float T);  
float RHOIV(float T,float P);  
double VISCIIV(float T,float P);  
float PRIV (float T,float P);  
float Tempsat(float P);  
/*cabecera propiedades coeficientes de transferencia de calor*/  
float H_bif_Kand(float PIM,float TIM,float q,float x2new,float GI, float D1);  
float H_bif_Katt(float PIM,float TIM,float q,float x2new,float GI, float D1,float epsg,float  
anglestrat,float Gwavy,float Gstrat,  
float classif);  
float enthalpyvap(float T);  
float H_gases (float TSM, float GS, float L, float DHS,float TSW);  
float H_int_monof (float TIM, float GI, float L, float D1,float TIW);  
float H_int_vapor (float PIM,float TIM, float GI, float L, float D1,float TIW);  
/*cabeceras perdidas de presion*/
```



```
float DeltaPS (float DHS,float L,float TI1,float TS1,float TS2new,float GS,float TSW);  
float DeltaPI (float DI, float L, float TI1, float TS1, float TI2new, float GI, float TIW);  
float DeltaPIvapor (float PIM,float DI,float L,float TI1,float TS1,float TI2new,  
float GI,float TIW);  
float DeltaPI_bif(float PIM,float DI,float TI1,float TI2new,float GI,float x2new);  
/*cabeceras mapas de flujo*/  
float flowp(float PIM,float TIM,float Aint,float GI,float MI1,float D1,tuplanl g,int i);  
void flowp2 (float PIM,float TIM,float q,float x,float GI,float D1,float eps_g,float *const  
Gwavy,float *const Gstrat,  
float *const anglestrat,float *const classif);  
/*otras cabeceras*/  
float Tempsat(float P);  
bool bifasico (float P,float T);  
bool estado_vapor (float x,float VI1);  
float enthalpyvap(float T);  
float FraccioVol(float PIM,float TIM,float x2new,float GI);  
void MetodoPRN (float D0, float DI, float L,float A0t,float ef,float Af, float LAMW, float TI1,  
float TS1, float HS,  
float HI, float CPI, float CPS, float MI, float MS,float *const TS2new,  
float *const TI2new, float * const TSWnew,float *const q,float *const R,float *const  
N,float *const P, float *const TIWnew,float *const RI,float *const  
RW,float *const RS,float *const UA,float *const Cc,float *const Ch,float *const DTI);  
void MetodoMLDT (float D0, float DI,float L,float A0t,float ef,float Af, float LAMW,  
float TI1, float TS1, float HS,float HI,float CPS, float MI, float MS,float TS2,  
float hfg,float x1,float *const TS2new, float *const TSWnew,float *const TIWnew,  
float *const x2new,float *const qnova);
```



```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
+++++*/  
  
/*********************INICIO DEL ALGORITMO PRINCIPAL*****/  
  
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  
+++++*/  
  
void main(void){  
float  
  
/*Datos iniciales del problema que se entrarán por pantalla*/  
  
TS1,          /* temperatura de entrada lado envolvente-gases */  
TS2,          /* temperatura de salida lado envolvente-gases */  
TSM,          /* temperatura media lado envolvente-gases */  
PS1,          /* presion de entrada lado envolvente-gases */  
DPSz,         /* perdida de presion lado envolvente-gases */  
VS1,          /* velocidad entrada gases*/  
MS1,          /* caudal masico entrada gases */  
RES_1,         /*Reynolds inicial entrada tubo*/  
PRS_1,         /*Prandtl inicial entrada tubo*/  
Aext,          /*area envolvente total*/  
Aextp,         /*area envolvente primaria*/  
TI1,          /* temperatura de entrada lado tubo */  
TI2,          /* temperatura de salida lado tubo */  
PI1,          /* presion de entrada lado tubo */  
DPlz,         /* perdida de presion lado tubo */  
VI1,          /*velocidad entrada agua liquida*/  
MI1,          /* caudal masico entrada tubo*/  
MI1lh,         /* caudal masico entrada tubo*/
```



```

REIL_1,           /*Reynolds inicial entrada tubo*/
PRIL_1,           /*Prandtl inicial entrada tubo*/
Aint,             /*area parte interior*/
/*Geometria del evaporador*/
L,                /* longitud del intercambiador*/
n,                /* numero de divisiones*/
D1,               /*diametro interior*/
D2,               /*diametro exterior*/
D3,               /*diametro calorifugado*/
DHS,              /*diametro hidraulico de los gases*/
incre_z,
z_monof,
z_bifasic,
z_reescalfat,    /* longitud del nodo a trabajar */
incre_ts,         /* incremento de la temperatura para iniciar la taula*/
incre_ti,
qtotal,
qtotal2;          /* potencia termica intercambiada total*/
int
ini_t,
ini_w;

float
GI, /*flujo masico inicial de liquido*/
GS, /*flujo masico parte envolvente gases*/
/*Propiedades de fluidos a temperatura inicial*/

```



```
RHOIL_1,  
RHOS_1,  
/*propiedades*/  
CPS_M,  
CPIL_M,  
/*conductivitat termica de la pared*/  
LAMW;  
/* Definicion tuplas */  
tuplanl temp_i ;  
tuplanl temp_s;  
tuplanl press_i;  
tuplanl press_s;  
tuplanc temp_iw;  
tuplanc temp_sw;  
tuplanl titol_x;  
tuplanl fvv;  
tuplanl flowpt;  
tuplanc coefconi;  
tuplanc coefcons;  
tuplanl vvapor;  
tuplanl vliquid;  
  
/*Definición variables aletas*/  
float Lf, /*altura de las aletas*/  
nf, /*numero de aletas*/  
A0, /*superficie primaria total m2/m (sup exterior de tubo no ocupada por las  
aletas)*/
```



```
A0t,  
Af, /*Superficie extendida*/  
DELTA, /*semigrosor de las aletas*/  
ef, /*eficiencia de las aletas*/  
m;  
/*variables bifasico*/  
float  
x1,  
x2new,  
eps_g,  
qb,  
pressat,  
qnova,  
DPIzbif,  
hfg;  
float TSmonof,Tlmonof;  
float TSbif,Tlbif;  
float TSre,Tlre;  
float fin_tubo;  
/*variables flowpattern*/  
float Gwavy,  
Gstrat,anglestrat,  
classif;  
/*variables ficheros*/  
/* Pide los datos por pantalla */  
int x0,y0;
```



```
int car;  
  
x0=5;y0=3;  
  
clrscr();  
  
gotoxy(x0+5,y0);cprintf(" **** PROYECTO: 'SIMULACION DE UN EVAPORADOR'  
****");  
  
printf("\n\n\n");  
  
gotoxy(x0+5,y0+3);cprintf(" **** Introduccion de datos ****");  
  
printf("\n\n\n");  
  
  
  
  
printf("Temperatura de entrada del agua? [en C] "); TI1=LlegirReal()+273.15;  
  
printf("Presion de entrada del agua?[en kPa] "); PI1=LlegirReal();  
  
printf("velocidad de entrada del agua? [en m/s] "); VI1=LlegirReal();  
  
printf("Temperatura de entrada de los gases? [en C] ");printf("(no recomendable mas de  
1150 C)"); TS1=LlegirReal()+273.15;  
  
printf("Presion de entrada de los gases?[en kPa] ");PS1=LlegirReal();  
  
printf("velocidad de entrada de los gases? [en m/s] "); /*VS1=LlegirReal();  
*/printf("VS1=50"); printf("\n");  
  
printf("Longitud del evaporador ? [m] "); L=LlegirReal(); printf("\n");  
  
printf("numero de divisiones? [max 199] "); n=LlegirEnter(); printf("\n");  
  
VS1=50;  
  
D3=0.08;  
  
D2=0.01;  
  
D1=0.006;  
  
nf=5.0;  
  
DELTA=0.0005;  
  
/*inicialización variables aletas*/  
  
Lf=(D3-D2)/2;
```



```
Af=2*L*Lf*nf;  
A0=PI*D2-nf*2*DELTA;  
A0t=A0*L;  
TS2=TI1;  
TSM=(TS1+TS2)/2 ;  
CPS_M=CPS(TSM);  
  
/*calculo de secciones, caudales y flujos másicos*/  
Aextp=(PI*(pow(D3,2.0)-pow(D2,2.0))/4.0;  
Aext=((PI*(pow(D3,2.0)-pow(D2,2.0))/4.0)-(nf*Lf*2*DELTA);  
RHOS_1=RHOS(TS1);  
MS1=RHOS_1*VS1*Aextp;  
GS=MS1/Aext;  
PRS_1=PRS(TS1);  
RES_1=GS*D1/VISCIL(TI1);  
DHS=4.0*Aext/((PI*(D2+D3)-nf*2*2*DELTA)+2*nf*Lf);  
PRS_1=PRS(TS1);  
RES_1=GS*DHS/VISCS(TS1);  
RHOIL_1=RHOIL(TI1);  
Aint=(PI *pow(D1,2.0))/4.0;  
MI1=RHOIL_1*VI1*Aint;  
GI=MI1/Aint;  
MI1lh=(MI1*1000*(3600/RHOIL_1));  
PRIL_1=PRIL(TI1);  
REIL_1=GI*D1/VISCIL(TI1);
```



```
Clrscr();  
  
gotoxy(x0,1);cprintf("*****  
PROYECTO: 'SIMULACION DE UN  
EVAPORADOR'*****");  
  
gotoxy(x0+3,y0);printf("*****Los datos introducidos son los siguientes:*****");  
  
gotoxy(x0,y0+2);printf(" **Geometria del Evaporador**");printf("\n\n");  
  
printf("\t");printf ("L=");printf("%1f",L);printf(" m");printf("\t\t");  
  
printf ("n=");printf("%1f",n);printf(" divisiones");printf("\t");  
  
printf("\n\n");  
  
gotoxy(x0,y0+6);printf(" ____ LADO ENVOLVENTE (gases de combustion)____ ");  
printf("\n\n");  
  
printf("\t");printf ("T_s1=");printf("%1f",TS1-273.15);printf(" C");  
  
printf("\t");printf ("P_s1=");printf("%1f",PS1);printf(" kPa");printf("\t");  
  
printf ("V_s=");printf("%1f",VS1);printf(" m/s");printf("\n\n");  
  
printf("\t");printf ("m_s=");printf("%3f",MS1);printf(" kg/s");printf("\t\t");  
  
printf ("G_s=");printf("%2f",GS);printf(" kg/m2s");printf("\t\t");printf("\n\n");  
  
printf("\t");printf ("Re_s=");printf("%2f",RES_1);printf("\t\t");printf  
("Pr_s=");printf("%2f",PRS_1);  
  
printf("\n\n");  
  
gotoxy(x0,y0+14);printf("-----LADO TUBO (agua)-----");printf("\n\n");  
  
printf("\t");printf ("T_I1=");printf("%1f",TI1-273.15);printf(" C");  
  
printf("\t\t");printf ("P_I1=");printf("%1f",PI1);printf(" kPa");printf("\t\t");  
  
printf ("V_I=");printf("%1f",VI1);printf(" m/s");  
  
printf("\n\n");  
printf("\t");printf ("m_I=");printf("%3f",MI1);printf(" kg/s");printf (" = ");
```



```

printf("%.2f",MI1lh);printf(" l/h");

printf("\t\t");printf ("G_I=");printf("%.2f",GI);printf(" kg/m2s");printf("\n\n");

printf("\t");printf ("Re_I=");printf("%.2f",REIL_1);

printf("\t\t");printf ("Pr_I=");printf("%.2f",PRIL_1);

printf("\n\n");

if (REIL_1<=2400){

    printf("¡¡ALERTA!! Re_I es muy bajo(reg.laminar). El programa
puede no funcionar");printf("\n\n");

} else if (!(REIL_1<=2400)) {

} else assert (0);

printf("pulse '1' para cambiar datos o '0' para continuar ");printf("\n");

car = LlegirEnter();

while( !(car==0)){

clrscr();

gotoxy(x0+5,y0);cprintf(" **** PROYECTO: 'SIMULACION DE UN EVAPORADOR' ****");

printf("\n\n\n");

gotoxy(x0+5,y0+3);cprintf(" **** Entrada de datos ****");

printf("\n\n\n");

printf("Temperatura de entrada del agua? [en C] "); TI1=LlegirReal()+273.15;

printf("Presion de entrada del agua?[en kPa] "); PI1=LlegirReal();

printf("velocidad de entrada del agua? [en m/s] "); VI1=LlegirReal();

printf("Temperatura de entrada de los gases? [en C] ");printf("(no recomendable mas de
1150 C) "); TS1=LlegirReal()+273.15;

```



```
printf("Presion de entrada de los gases?[en kPa] ");PS1=LlegirReal();

printf("velocidad de entrada de los gases? [en m/s] " ); /*VS1=LlegirReal();
/*printf("VS1=50"); printf("\n");

printf("Longitud del evaporador ? [m] "); L=LlegirReal(); printf("\n");

printf("numero de divisiones? [max 199] "); n=LlegirEnter(); printf("\n");

/*inicialización variables aletas*/

Lf=(D3-D2)/2;

Af=2*L*Lf*nf;

A0=PI*D2-nf*2*DELTA;

A0t=A0*L;

TS2=TI1;

TSM=(TS1+TS2)/2 ;

CPS_M=CPS(TSM);

/*calculo de secciones, caudales y flujos máicos*/

Aextp=(PI*(pow(D3,2.0)-pow(D2,2.0))/4.0;

Aext=((PI*(pow(D3,2.0)-pow(D2,2.0))/4.0)-(nf*Lf*2*DELTA);

RHOS_1=RHOS(TS1);

MS1=RHOS_1*VS1*Aextp;

GS=MS1/Aext;

DHS=4.0*Aext/((PI*(D2+D3)-nf*2*2*DELTA)+2*nf*Lf);

PRS_1=PRS(TS1);

RES_1=GS*DHS/VISCS(TS1);

RHOIL_1=RHOIL(TI1);

Aint=(PI *pow(D1,2.0))/4.0;
```



```

MI1=RHOIL_1*VI1*Aint;
GI=MI1/Aint;
MI1lh=(MI1*1000*(3600/RHOIL_1));
PRIL_1=PRIL(TI1);
REIL_1=GI*D1/VISCIL(TI1);

clrscr();

gotoxy(x0,1);cprintf("*****");
PROYECTO:      'SIMULACION      DE      UN
EVAPORADOR*****;

gotoxy(x0+3,y0);printf("*****Los datos introducidos son los siguientes:*****");
gotoxy(x0,y0+2);printf(" **Geometria del Evaporador**");printf("\n\n");
printf("\t");printf ("L=");printf("%.1f",L);printf(" m");printf("\t\t");
printf ("n=");printf("% .1f",n);printf(" divisiones");printf("\t");
printf("\n\n");

gotoxy(x0,y0+6);printf(" ____ LADO ENVOLVENTE (gases de combustion)____ ");
printf("\n\n");
printf("\t");printf ("T_s1=");printf("%.1f",TS1-273.15);printf(" C");
printf("\t");printf ("P_s1=");printf("%.1f",PS1);printf(" kPa");printf("\t");
printf ("V_s=");printf("% .1f",VS1);printf(" m/s");printf("\n\n");
printf("\t");printf ("m_s=");printf("%.3f",MS1);printf(" kg/s");printf("\t\t");
printf ("G_s=");printf("%.2f",GS);printf(" kg/m2s");printf("\t\t");printf("\n\n");
printf("\t");printf ("Re_s=");printf("%.2f",RES_1);printf("\t\t");printf
("Pr_s=");printf("%.2f",PRS_1);

printf("\n\n");

gotoxy(x0,y0+14);printf("-----LADO TUBO (agua)-----");printf("\n\n");
printf("\t");printf ("T_l1=");printf("%.1f",TI1-273.15);printf(" C");

```



```
printf("\t\t");printf ("P_I1=");printf("% .1f",PI1);printf(" kPa");printf("\t\t");
printf ("V_I=");printf("% .1f",VI1);printf(" m/s");
printf("\n\n");
printf("\t");printf ("m_I=");printf("% .3f",MI1);printf(" kg/s");printf (" = ");
printf("% .2f",MI1lh);printf(" l/h");
printf("\t\t");printf ("G_I=");printf("% .2f",GI);printf(" kg/m2s");printf("\n\n");
printf("\t");printf ("Re_I=");printf("% .2f",REIL_1);
printf("\t\t");printf ("Pr_I=");printf("% .2f",PRIL_1);
printf("\n\n");
if (REIL_1<=2400){
    printf("¡¡ALERTA!! Re_I es muy bajo(reg.laminar). El programa
puede no funcionar");printf("\n\n");
} else if (!(REIL_1<=2400)) {
} else assert (0);
printf("pulse '1' para cambiar datos o '0' para continuar ");printf("\n");
car = LlegirEnter();
}

/*Cálculo del calor total disponible para la temperatura inicial de los gases*/
qtotal=MS1*CPS_M*(TS1-TS2);
TI2=(qtotal/(MI1*4200))+TI1;
qtotal2=0.0;
incre_z=L/n;
incre_ts=(TS1-TS2)/n;
incre_ti=(TI2-TI1)/n;
/*inicializacion tablas de temperaturas interior y exterior, presiones y título*/
temp_i.ntablal=n+1;
```



```
temp_s.ntablal=n+1;  
press_i.ntablal=n+1;  
press_s.ntablal=n+1;  
titol_x.ntablal=n+1;  
fvv.ntablal=n+1;  
flowpt.ntablal=n+1;  
vvapor.ntablal=n+1;  
vliquid.ntablal=n+1;  
ini_t=0;  
vliquid.tabla[ini_t]=VI1;  
vvapor.tabla[ini_t]=0.0;  
titol_x.tabla[ini_t]=0.0;  
fvv.tabla[ini_t]=0.0;  
flowpt.tabla[ini_t]=0.0;  
press_i.tabla[ini_t]=PI1;  
press_s.tabla[ini_t]=PS1;  
temp_s.tabla[ini_t]=TS1;  
temp_i.tabla[ini_t]=TI1;  
while(ini_t<n){  
    temp_s.tabla[ini_t+1]=temp_s.tabla[ini_t] -incre_ts;  
    temp_i.tabla[ini_t+1]=temp_i.tabla[ini_t] +incre_ti;  
    ini_t=ini_t+1;  
}  
/* fin de inicializacion tablas de temperaturas interior y exterior*/  
/*inicializacion tablas de temperaturas pared*/  
coefcons.ntablac=n;
```



```
coefconi.ntablac=n;
temp_iw.ntablac=n;
temp_sw.ntablac=n;
ini_w=0;
while(ini_w<n){
    temp_iw.tabla[ini_w]=TI1;
    coefcons.tabla[ini_w]=0.0;
    coefconi.tabla[ini_w]=0.0;
    temp_sw.tabla[ini_w]=TI1;
    ini_w=ini_w+1;
}

fin_tubo=0.0;
*****|INICIO TUBO PRIMERA DISCRETIZACIÓN*****|
int i;
i=0;
z_monof=0.0;
z_bifasic=0.0;
z_reescalfat=0.0;
/*defino variables locales que emplearemos para la iteración de cada sección*/
float TIz2,TIz1,TSz2,TSz1,TIWz,TSWz;
float error;
float TSMz,TIMz;
float HS,HI,Hlmonof,Hlbif,Hlbif2,Hlvapor;
/*inicio de dichas variables locales de la parte de flujo monofásico*/
TIz1=temp_i.tabla[i];
```



```
TSz1= temp_s.tabla[i];
/*definicion de variables nuevas para la iteración*/
float Tlz2n,TSz2n;
TIWz=temp_iw.tabla[i];
TSWz=temp_sw.tabla[i];
/*TSW=TSWz;*/
TSz2n=(Tlz1+TSz1)/2;
Tlz2n=TSz2n;
LAMW=20.0;
error=0.001;
float niter;
niter=0.0;
z_monof=incre_z;
/*Variables del Metodo PRN a emplear en la zona monofásica*/
float qparcial,q,R,P,N,RI,RW,RS,UA,Cc,Ch,DTI;
qparcial=MI1*4200*(Tlz2n-Tlz1);
q=0.0;
printf("%.2f",qparcial);
printf("%.2f",temp_i.tabla[i]);
printf("%.2f",temp_i.tabla[i+1]);
/*********************************************&&&&&&&&&&&&&******/
/*********************************************INICIO TUBO******/
/*********************************************&&&&&&&&&&&&&&******/
Tlz2n=Tlz1;
while(fin_tubo<0.5){
/*INICIO PARTE FLUJO MONOFASICO******/
```



```
while((z_monof<=L)&&(!bifasico(press_i.tabla[i],Tlz2n))){  
/*inicialización variables de la sección a estudiar*/  
Tlz1=temp_i.tabla[i];  
TSz1= temp_s.tabla[i];  
Tlz2=temp_i.tabla[i+1];  
TSz2= temp_s.tabla[i+1];  
TIWz=temp_iw.tabla[i];  
TSWz=temp_sw.tabla[i];  
TSz2n=(Tlz1+TSz1)/2;  
Tlz2n=TSz2n;  
/*entrada en el bucle para iterar las temperaturas*/  
while((fabs(TSz2-TSz2n)+fabs(Tlz2-Tlz2n))>error){  
Tlz2=Tlz2n;  
TSz2=TSz2n;  
TSMz=(TSz1+TSz2)/2.0;  
TIMz=(Tlz1+Tlz2)/2.0;  
  
/*calculo las propiedades del agua liquida a temperatura media*/  
CPIL_M=CPIL(TIMz);  
/*calculo las propiedades de gases a temperatura media*/  
CPS_M=CPS(TSMz);  
/*Calculo de los coeficientes de convección*/  
Hlmonof=H_int_monof (TIMz,GL,incre_z,D1,TIWz);  
Hl=Hlmonof;  
HS =H_gases (TSMz,GS,incre_z,DHS,TSWz);  
/*Cálculo de propiedades de las aletas*/
```



```
m=pow((HS/(LAMW*DELTA)),0.5);
ef=(tanh(m*Lf))/(m*Lf);
```

```
MetodoPRN(D2,D1,incre_z,A0t,ef,Af,LAMW,TIz1,TSz1,HS,HI,CPII_M,CPS_M,
```

```
MI1,MS1,&TSz2n,&TIz2n,&TSWz,&q,&R,&N,&P,&TIWz,&RI,&RW,&RS,&UA,&Cc,&Ch,&D
TI);
```

```
/*Contador del número de iteraciones realizadas*/
```

```
niter=niter+1.0;
```

```
}
```

```
/*Asignación de las variables ya iteradas */
```

```
temp_i.tabla[i+1]=TIz2n;
```

```
temp_s.tabla[i+1]=TSz2n;
```

```
TImonof =TIz2n;
```

```
TSmonof =TSz2n;
```

```
temp_iw.tabla[i]=TIWz;
```

```
temp_sw.tabla[i]=TSWz;
```

```
coefcons.tabla[i]=HS;
```

```
coefconi.tabla[i]=HI;
```

```
DPIz=DeltaPI ( D1,incre_z, TIz1, TSz1, TIz2n, GI, TIWz);
```

```
DPSz=DeltaPS(DHS,incre_z, TIz1, TSz1, TSz2n, GS, TSWz);
```

```
press_s.tabla[i+1]=press_s.tabla[i]-DPSz;
```

```
press_i.tabla[i+1]=press_i.tabla[i]-DPIz;
```

```
vliquid.tabla[i+1]=VI1;
```



```
vvapor.tabla[i+1]=0.0;  
titol_x.tabla[i+1]=0.0;  
fvv.tabla[i+1]=0.0;  
flowpt.tabla[i+1]=0.0;  
qtotal2=qtotal2+q;  
z_monof=z_monof+incre_z;  
i=i+1;  
}  
if (z_monof>=L) {  
    fin_tubo=1.0;  
  
} else if (!(z_monof>=L)) {  
/*sentencia condicional para saber si se ha llegado a fin de tubo*/  
/*************INICIO FLUJO BIFASICO*****/  
z_bifasic=z_monof;  
titol_x.tabla[i]=0.001;  
pressat=press_i.tabla[i];  
eps_g=FraccVol(pressat,TIMz,titol_x.tabla[i],GI);  
fvv.tabla[i]=eps_g;  
temp_i.tabla[i]=Tempsat(pressat)-0.5;  
  
flowp2(pressat,temp_i.tabla[i],q,titol_x.tabla[i],GI,D1,eps_g,&Gwavy,&Gstrat,  
&anglestrat,&classif);  
flowpt.tabla[i]=classif;  
Tlmonof=temp_i.tabla[i];  
qnova=q;  
while ((z_bifasic<=L)&&(!estado_vapor(titol_x.tabla[i],VI1))){
```



```

/*inicialización variables de la sección a estudiar*/
Tlz1=Tempsat(pressat);
Tlz2=Tlz1;
TSz1= temp_s.tabla[i];
TSz2= temp_s.tabla[i+1];
TIWz=temp_iw.tabla[i];

TSWz=temp_sw.tabla[i];
TSz2n=(Tlz1+TSz1)/2;

x1= titol_x.tabla[i];
eps_g=fvv.tabla[i];

x2new=(titol_x.tabla[i+1]+titol_x.tabla[i])/2;
/*entrada en el bucle para iterar las variables*/
while((fabs(TSz2-TSz2n))>error){

TSz2=TSz2n;
TSMz=(TSz1+TSz2)/2.0;
TIMz=(Tlz1+Tlz2)/2.0;
CPS_M=CPS(TSMz);
qb=700;
hfg=enthalpyvap(TIMz);

/*Calculo de los coeficientes de convección */
Hlbif=H_bif_Kand( press_i.tabla[i],TIMz,qb,x2new,GI,D1);

Hlbif2=H_bif_Katt(press_i.tabla[i],TIMz,qb,x2new,GI,D1,eps_g,anglestrat,Gwavy,Gstrat,classeif);

HI=Hlbif2;

HS =H_gases (TSMz,GS,incre_z,DHS,TSWz);

```



```
m=pow((HS/(LAMW*DELTA)),0.5);

ef=(tanh(m*Lf))/(m*Lf);

MetodoMLDT(D2,D1,incre_z,A0t,ef,Af,LAMW,TIz1,TSz1,HS,HI,CPS_M,
MI1,MS1,TSz2,hfg,x1,&TSz2n,&TSWz,&TIWz,&x2new,&qnova);

}

niter=niter+1.0;

temp_i.tabla[i+1]=TIz2;

temp_s.tabla[i+1]=TSz2n;

TIbif=TIz2;

TSbif=TSz2n;

temp_iw.tabla[i]=TIWz;

temp_sw.tabla[i]=TSWz;

coefcons.tabla[i]=HS;

coefconi.tabla[i]=HIbif2;

titol_x.tabla[i+1]=x2new;

eps_g= FraccVol(pressat,TIMz,titol_x.tabla[i+1],GI);

fvv.tabla[i+1]=eps_g;

flowp2(pressat,TIMz,qnova,titol_x.tabla[i+1],GI,D1,eps_g,&Gwavy,&Gstrat,
&anglestrat,&classif);

flowpt.tabla[i+1]=classif;

DPIzbif=DeltaPI_bif(press_i.tabla[i],D1,TIz1,TSz2n,GI,x2new);

press_i.tabla[i+1]=press_i.tabla[i]-DPIzbif;

DPSz=DeltaPS(DHS,incre_z,TLz1,TSz1,TSz2n,GS,TSWz);

press_s.tabla[i+1]=press_s.tabla[i]-DPSz;

z_bifasic=z_bifasic+incre_z;

qtotal2=qtotal2+qnova;
```



```

i=i+1;

}

if(z_bifasic>=L){

fin_tubo=2.0;

}else if(!(z_bifasic>=L)) {

/********************************************/


/*INICIO PARTE ZONA RECALENTADA*****/


/********************************************/


z_reescalfat=z_bifasic;

titol_x.tabla[i]=1.0;

pressat=press_i.tabla[i];

while ((z_reescalfat<=L)) {

/*inicialización variables de la sección a estudiar*/

Tlz1=temp_i.tabla[i];

Tlz2=temp_i.tabla[i+1];

TSz1= temp_s.tabla[i];

TSz2= temp_s.tabla[i+1];

TIWz=temp_iw.tabla[i];

TSWz=temp_sw.tabla[i];

TSz2n=(Tlz1+TSz1)/2;

Tlz2n=TSz2n;

while((fabs(TSz2-TSz2n)+fabs(Tlz2-Tlz2n))>error){

/*entrada en el bucle para iterar las variables*/

Tlz2=Tlz2n;

TSz2=TSz2n;

TSMz=(TSz1+TSz2)/2.0;

```



```
TIMz=(Tlz1+Tlz2)/2.0;  
  
CPII_M=CPII(TIMz);  
CPS_M=CPS(TSMz);  
  
/*Calculo de los coeficientes de convección*/  
  
Hvapor=H_int_vapor(press_i.tabla[i],TIMz,GI,incre_z,D1,TIWz);  
  
HI=Hvapor;  
  
HS =H_gases (TSMz,GS,incre_z,DHS,TSWz);  
  
m=pow((HS/(LAMW*DELTA)),0.5);  
  
ef=(tanh(m*Lf))/(m*Lf);  
  
/*Metodo PRN*/  
  
MetodoPRN(D2,D1,incre_z,A0t,ef,Af,LAMW,Tlz1,TSz1,HS,HI,CPII_M,CPS_M,  
MI1,MS1,&TSz2n,&TLz2n,&TSWz,&q,&R,&N,&P,&TIWz,&RI,&RW,&RS,&UA,&Cc,&Ch,&D  
TI); /* Insertar funcio o accio del Metode PRN */  
  
niter=niter+1.0;  
  
}  
  
niter=niter+1.0;  
  
temp_i.tabla[i+1]=TLz2n;  
  
temp_s.tabla[i+1]=TSz2n;  
  
TLre=TLz2n;  
  
TSre=TSz2n;  
  
temp_iw.tabla[i]=TIWz;  
  
temp_sw.tabla[i]=TSWz;  
  
coefcons.tabla[i]=HS;  
  
coefconi.tabla[i]=HI;  
  
titol_x.tabla[i+1]=1.0;  
  
fvv.tabla[i+1]=1.0;  
  
flowpt.tabla[i+1]=7.0;
```



```
DPIz=DeltaPIvapor (press_i.tabla[i],D1,incre_z,TIz1,TSz1,TLz2n,GI,TIWz);  
press_i.tabla[i+1]=press_i.tabla[i]-DPIz;  
DPSz=DeltaPS(DHS,incre_z,TIz1,TSz1,TSz2n,GS,TSWz);  
press_s.tabla[i+1]=press_s.tabla[i]-DPSz;  
qtotal2=qtotal2+qnova;  
z_reescalfat=z_reescalfat+incre_z;  
i=i+1;  
}  
fin_tubo=5.0;  
} else assert (0);  
} else assert (0);  
}  
if (z_monof>L){  
z_monof =z_monof-incre_z;  
}else if(!(z_monof>L)){ z_monof =z_monof;  
}else assert(0);  
if (z_bifasic>L){  
z_bifasic =z_bifasic-incre_z;  
}else if(!(z_bifasic>L)){ z_bifasic =z_bifasic;  
}else assert(0);  
if (z_reescalfat>L){  
z_reescalfat =z_reescalfat-incre_z;  
}else if(!(z_reescalfat>L)){ z_reescalfat =z_reescalfat;  
}else assert(0);  
printf("\n\n");  
printf("correcto");printf("\n\n");
```



```
presentaresultats( n, L,z_monof,z_bifasic,z_reescalfat,TSmonof,Tlmonof,TSbif,Tlbif,
TSre,Tlre,x2new);

presentaresultats2(temp_i,temp_s, temp_iw,coefconi,fvv,titol_x,flowpt);

resultados(n,L,temp_i,temp_s, temp_iw,coefconi,fvv,titol_x,flowpt);

printf("\n\n");

printf("Los resultados han quedado grabados en c:resultados.dat****");printf("\n");

printf("*****FIN DEL PROGRAMA*****");printf("\n\n");

}

/********************************************/
```

```
/********************************************FIN DEL PROGRAMA*****/
```

```
/********************************************/
```

```
/*#####
#####
#####*/

```

```
*****FUNCIONES *****/
```

```
/*#####
#####
#####*/

```

```
void resultados( int n,float L,tuplanl a,tuplanl b, tuplanc d,tuplanc e,tuplanl f,tuplanl g,tuplanl h){
```

```
FILE *file1;
```

```
file1=fopen("c:\\resultados.dat","w");
```

```
if (file1==NULL){
```

```
    printf ("Error creant fitxer c:\\resultados.dat\\n");
```

```
    exit (1);
```

```
}
```



```

int contf;
contf=0;
float incre_z,z_total;
incre_z=L/n;
z_total=0.0;
fprintf(file1," long. tubo(m)\t\temp. agua(°C)\t\temp. gas(°C)\t\temp. pared(°C) \tfraction
vol \ttitulo\t\coef.conveccion\t tipo de flujo \n\n");
while (contf <n+1){
printf (file1," %.2f\t\tT_i[%d]= %.2f\t\tT_s[%d]= %.1f\t\tT_wall[%d]= %.2f \tf.vol[%d]= %.3f
\titulo[%d]= %.3f \t H[%d]= %.2f\t",
z_total,contf+1,a.tabla[contf]-273.15,contf+1,b.tabla[contf]-273.15,contf+1,d.tabla[contf]-
273.15,
contf+1,f.tabla[contf],contf+1,g.tabla[contf],contf+1,e.tabla[contf]);
if (h.tabla[contf]==0.0){
fprintf(file1,"liquido\n\n");
} else if (h.tabla[contf]==1.0) {
fprintf(file1,"bubbly \n\n");
} else if (h.tabla[contf]==2.0) {
fprintf(file1,"stratified \n\n");
} else if (h.tabla[contf]==3.0) {
fprintf(file1,"stratif. wavy \n\n");
} else if (h.tabla[contf]==4.0) {
fprintf(file1,"intermittent \n\n");
} else if (h.tabla[contf]==5.0) {
fprintf(file1,"annular \n\n");
} else if (h.tabla[contf]==6.0) {
fprintf(file1,"mist flow \n\n");
} else if (h.tabla[contf]==7.0) {
}
}

```



```
fprintf(file1,"vapor \n\n");
} else assert (0);

z_total=z_total+incre_z;
contf=contf+1;

}

fclose(file1);
}

void presentarResultats(int n,float L,float l1,float l2,float l3,float Tg1,float Tw1,float Tg2,
float Tw2,float Tg3,float Tw3,float x_tit){

float ancho;

float incre_z,peu;

incre_z=L/n;

ancho=70.0;

peu=ancho/n;

double xm,xb;

xm=floor((l1/incre_z)*peu) ;

xb=floor((l2/incre_z)*peu);

int c;

int x0,y0;

int j;

j=6;

x0=0;y0=0;

clrscr();

gotoxy(x0+10,y0+1);cprintf(" **** PROYECTO: 'SIMULACION DE UN EVAPORADOR'
****");

printf("\n\n\n");
```



```
gotoxy(x0+10,y0+4);
printf("*****RESULTADOS*****\n\n");
printf("pulse 1 para ver resultados o 0 para salir ");printf("\n");
c = LlegirEnter();
while( !(c==0)){
    clrscr();
    gotoxy(x0+3,y0+1);
    printf("*****RESULTADOS*****\n\n");
    gotoxy(x0+6,y0+7);
    printf("____");
    gotoxy(x0+6,y0+10);
    printf("-----");
    gotoxy(x0+6,y0+13);
    printf("-----");
    gotoxy(x0+6,y0+15);
    printf("____");
    if(l3>0.0){
        gotoxy(x0+1,y0+2);
        printf("z_m=fin del estado monofasico");
        gotoxy(x0+31,y0+2);
        printf("z_b=fin est bifasico");
        gotoxy(x0+55,y0+2);
        printf("z_r=fin est recalentado");
        gotoxy(x0+8,y0+11);
        printf("liq");
    }
}
```



```
gotoxy(xm+13,y0+11);
printf("bifasico");

gotoxy(xb+7,y0+11);
printf("vapor");

gotoxy(xm+6,y0+3);
printf("z_m=");printf("%.2f",l1);printf(" m");
gotoxy(xm+6,y0+17);
printf("Ti=");printf("%.2f",Tw1-273.15);printf(" C");
gotoxy(xm+6,y0+18);
printf("Ts=");printf("%.2f",Tg1-273.15); printf(" C");
gotoxy(xb,y0+4);
printf("z_b=");printf("%.2f",l2);printf(" m");
gotoxy(xb,y0+19);
printf("Ti=");printf("%.2f",Tw2-273.15);printf(" C");
gotoxy(xb,y0+20);
printf("Ts=");printf("%.2f",Tg2-273.15);printf(" C");
gotoxy(70,y0+5);
printf("z_r=");printf("%.2f",l3);printf(" m");
gotoxy(70,y0+21);
printf("Ti=");printf("%.2f",Tw3-273.15);printf(" C");
gotoxy(70,y0+22);
printf("Ts=");printf("%.2f",Tg3-273.15);printf(" C");
while (j<=16){
    gotoxy(xm+10,y0+j);printf("|");
    gotoxy(xb+4,y0+j);printf("|");
```



```
gotoxy(76,y0+j);printf("|\n");
j=j+2;
}
}else if(!(l3>0.0)){
}else assert (0);

if(l3==0.0&& !(l2==0.0 ){

gotoxy(x0+1,y0+2);
printf("z_m=fin estado monofasico");
gotoxy(x0+31,y0+2);
printf("z_b=fin est bifasico");
gotoxy(x0+8,y0+11);
printf("liq");
gotoxy(xm+15,y0+11);
printf("bifasico");
gotoxy(xm+6,y0+4);
printf("z_m=");printf("%.2f",l1);printf(" m");
gotoxy(xm+6,y0+17);
printf("Ti=");printf("%.2f",Tw1-273.15);printf(" C");
gotoxy(xm+6,y0+18);
printf("Ts=");printf("%.2f",Tg1-273.15);printf(" C");
gotoxy(70,y0+5);
printf("z_b=");printf("%.2f",l2);printf(" m");
gotoxy(xb,y0+19);
printf("Ti=");printf("%.2f",Tw2-273.15);printf(" C");
gotoxy(xb,y0+20);
```



```
printf("Ts=");printf("%.2f",Tg2-273.15);printf(" C");
gotoxy(xb,y0+21);
printf("titulo=");printf("%.3f",x_tit);
while (j<=16){
    gotoxy(xm+10,y0+j);printf("|");
    gotoxy(76,y0+j);printf("|");
    j=j+2;
}
}else if(!(l3==0.0&& !l2==0.0)){
}else assert (0);
if(l3==0.0 && l2==0.0 ){
    gotoxy(x0+1,y0+2);
    printf("z_m=longitud en estado monofasico");
    gotoxy(x0+36,y0+11);
    printf("liquido");
    gotoxy(70,y0+5);
    printf("z_m=");printf("%.2f",l1);printf(" m");
    gotoxy(xm+6,y0+17);
    printf("Ti=");printf("%.2f",Tw1-273.15);printf(" C");
    gotoxy(xm+6,y0+18);
    printf("Ts=");printf("%.2f",Tg1-273.15);printf(" C");
    while (j<=16){
        gotoxy(76,y0+j);printf("|");
        j=j+2;
    }
}else if(!(l3==0.0 && l2==0.0 )){
```



```
 }else assert (0);

 gotoxy(1,y0+8);printf("Gases=>");
 gotoxy(1,y0+14);printf("Gases=>");
 gotoxy(1,y0+11);printf("Agua=>");

 gotoxy(x0+3,24);
 printf("pulse 0 para pasar a RESULTADOS 2/2 ");
 c= LlegirEnter();

}

printf("fin de resultados ");printf("\n\n");
}

void presentarResultats2(tuplanl a,tuplanl b, tuplanc d,tuplanc e,tuplanl f,tuplanl g,tuplanl h){

int c;
int r1;
int i;
int x0,y0;
x0=5;y0=1;
i=0;

clrscr();

gotoxy(x0+5,y0);
printf("*****RESULTADOS 2/2*****\n\n");
printf("pulse 1 para ver resultados o 0 para salir ");printf("\n");
c = LlegirEnter();
while( !(c==0)) {
    clrscr();
```



```
gotoxy(x0+5,y0);

printf("*****RESULTADOS    2/*****\n\n");

printf("pulse '1' para ver temperaturas de liquido");printf("\n");

printf("pulse '2' para ver temperaturas gas");printf("\n");

printf("pulse '3' para ver temperaturas pared");printf("\n");

printf("pulse '4' para ver coef de conveccio interior");printf("\n");

printf("pulse '5' para ver fraccion volumetrica");printf("\n");

printf("pulse '6' para ver vector de titulos");printf("\n");

printf("pulse '7' para ver flowpattern");printf("\n");

r1= LlegirEnter();

if (r1==1){

while( i<a.ntablal){

printf("%.2f",a.tabla[i]-273.15);

printf("\t ");

i=i+1; }

}else if (r1==2){

while( i<b.ntablal){

printf("%.2f",b.tabla[i]-273.15);

printf("\t ");

i=i+1; }

}else if (r1==3){

while( i<d.ntablac){

printf("%.2f",d.tabla[i]-273.15);

printf("\t ");

i=i+1; }

}else if (r1==4){
```



```
while( i<e.ntablac){
    printf("%.2f",e.tabla[i]);
    printf("\t");
    i=i+1;
}else if (r1==5){
    while( i<f.ntablaf){
        printf("%.4f",f.tabla[i]);
        printf("\t");
        i=i+1;
    }else if (r1==6){
        while( i<g.ntablaf){
            printf("%.4f",g.tabla[i]);
            printf("\t");
            i=i+1;
        }else if (r1==7){
            while( i<h.ntablaf){
                printf("%.4f",h.tabla[i]);
                printf("\t");
                i=i+1;
            }else assert(0);
            printf("\n\n");
            printf("pulse 1 para ver MAS resultados o 0 para salir ");printf("\n\n");
            c = LlegirEnter();
            i=0;
        }
        printf("fin de resultados ");printf("\n\n");
```



```
}

float LlegirReal (void)

{

int ret;

float f;

ret=scanf("%f", &f);

assert( ret==1);

return f;

}

int LlegirEnter (void)

{

int i,ret;

ret=scanf("%d", &i);

assert( ret==1);

return i;

}

/*#####
##*/
*****FUNCIONES DE LAS PROPIEDADES*****/

/*#####
##*/
*****FUNCIONES DE LAS PROPIEDADES DE LOS GASES DE
ESCAPE*****/

/** CPS devuelve el valor del calor específico, LAMS el valor de la conductividad termica,
RHOS retorna el valor de la densidad y VISCS el valor de la viscosidad dinamica.

PRS es la función que retorna el numero de Prandtl*****/

float CPS (float T)
```



```
{  
    float aux;  
  
    aux=1219.7862-0.74579*T+0.00139635*T*T-5.3179758e-7*T*T*T;  
  
    return aux;  
}  
  
float LAMS (float T)  
{  
    float aux;  
  
    aux=0.0001278019*pow(T,0.9153355);  
  
    return aux;  
}  
  
float RHOS (float T)  
{  
    float aux,aux2;  
  
    aux2=1/T;  
  
    aux=321.48191*pow(3369.3273,aux2)*pow(T,-0.98473204);  
  
    return aux;  
}  
  
float VISCS (float T)  
{  
    float aux;  
  
    aux=9.3888038e-7+5.99889569e-8*T-2.7503556e-11*T*T+8.1703725e-  
15*T*T*T;  
  
    return aux;  
}
```



```
float PRS (float T)

{
    float caes,vi,cond;
    caes=CPS(T);
    cond=LAMS(T);
    vi=VISCS(T);
    return vi*caes/cond;
}

/*FUNCIONES DE LAS PROPIEDADES DEL AGUA EN ESTADO
LIQUIDO***** */

/** CPIL devuelve el valor del calor especifico, LAMIL el valor de la conductividad termica,
RHOIL retorna el valor de la densidad y VISCL el valor de la viscosidad dinamica.
PRIL es la fucion que retorna el numero de Prandtl*****/

double CPIL(float T)

{
    return (2820+11.82*T-0.03502*T*T+3.599e-5*T*T*T);
}

double LAMIL(float T)

{
    return (-0.3835+0.00525*T-0.000006265*T*T);
}

float RHOIL (float T)

{
    return (741.966+1.9613*T-0.00371211*T*T);
}

double VISCL(float T)
```



```

{

    return (pow(10,(-13.73+(1830/T)+0.0197*T-0.0000147*T*T)));
}

double PRIL (float T)

{

    return( ((pow(10,(-13.73+(1830/T)+0.0197*T-0.0000147*T*T)))*
(2820+11.82*T-0.03502*T*T+3.599e-5*T*T*T))/(-0.3835+0.00525*T-
0.000006265*T*T));
}

/*FUNCIONES DE LAS PROPIEDADES DEL AGUA EN ESTADO
VAPOR******/

```

/** LAMIV devuelve el valor de la conductividad termica,RHOIV retorna el valor de la densidad

y VISCIVel valor de la viscosidad dinamica.PRIV es la fucion que retorna el numero de Prandtl*****/

```

double LAMIV(float T)

{
    return (-0.007145+0.000084*T);

}

float RHOIV(float T,float P)

{
    return ((18.015*P)/(8.314*T));

}

double VISCIV(float T,float P)

{
    return (-0.000002927+0.0000000407*T-(P/2810e6));
}

```



```
}
```

```
float PRIV (float T,float P)
{
    float caes,vi,cond;
    caes=1870;
    cond=LAMIV(T);
    vi=VISCIW(T,P);
    return vi*caes/cond;
}
```

```
/*#####
#####*/
*****FUNCIONES      COEFICIENTES      DE      TRANSFERENCIA      DE
CALOR*****/
```

```
/*#####
#####*/
***** "H_int_monof" retorna el valor del coeficiente de convección para el agua en
estado líquido*****/
```

```
float H_int_monof (float TIM, float GI, float L, float D1,float TIW){
    /*declaración de variables locales*/
    float n;
    float HI;
    double CFI,PRIL_M,NUI,VISCI_W,PHI_I,VISCIL_M,LAMIL_M,REIL;
    PRIL_M=PRIL(TIM);
    VISCIL_M=VISCIL(TIM);
    VISCI_W = VISCIL(TIW);
    REIL=GI*D1/VISCIL_M;
```



```

LAMIL_M=LAMIL(TIM);

CFI=1/((1.58*log(REIL)-3.28)*(1.58*log(REIL)-3.28));

PHI_I= VISCIL_M/VISCI_W;

    if (VISCI_W < VISCIL_M) {

        n=0.11;

    } else if (!(VISCI_W < VISCIL_M )) {

        n=0.25;

    } else assert(0);

NUI=(CFI*(REIL-1000)*PRIL_M*(1+(pow((D1/L),0.666667)))*pow(PHI_I,n))/

(2+ (17.96*pow(CFI,0.5)*(pow(PRIL_M,0.666667)-1)));

HI=(NUI*LAMIL_M)/D1;

return ( HI );

}

/******** "H_gases" retorna el valor del coeficiente de conveccion para los gases de
combustión****/

float H_gases (float TSM, float GS, float L, float DHS,float TSW){

/*declaración de variables locales*/

float n;

float HS;

double CFS,PRS_M,NUS,PHI_S,VISCS_M,LAMS_M,RES;

PRS_M=PRS(TSM);

LAMS_M=LAMS(TSM);

VISCS_M=VISCS(TSM);

RES=GS*DHS/VISCS_M;

CFS=1/((1.58*log(RES)-3.28)*(1.58*log(RES)-3.28));

PHI_S=TSM/TSW;

if (TSW>TSM) {

```



```
n=0.47;  
 } else if (!(TSW>TSM)){  
     n=0.36;  
 } else assert(0);  
  
 NUS=    (CFS*(RES-1000)*PRS_M*(1+(pow((DHS/L),0.666667)))*pow(PHI_S,n))/  
 (2+ (17.96*pow(CFS,0.5)*(pow(PRS_M,0.666667)-1)));  
 HS=(NUS*LAMS_M)/DHS;  
 return ( HS );  
}  
  
***** "H_int_vapor" retorna el valor del coeficiente de conveccion para el agua en  
estado vapor****/  
  
float H_int_vapor (float PIM,float TIM, float GI, float L, float D1,float TIW){  
/*declaración de variables locales*/  
float n;  
float HI;  
double CFI,PRIV_M,NUI,PHI_I,VISCIV_M,LAMIV_M,REIV;  
PRIV_M=PRIV(TIM,PIM);  
VISCIV_M=VISCIV(TIM,PIM);  
REIV=GI*D1/VISCIV_M;  
LAMIV_M=LAMIV(TIM);  
CFI=1/((1.58*log(REIV)-3.28)*(1.58*log(REIV)-3.28));  
PHI_I=TIM/TIW;  
if (TIW>TIM) {  
    n=0.47;  
 } else if (!(TIW>TIM)){  
    n=0.36;
```



```

} else assert(0);

NUI= (CFI*(REIV-1000)*PRIV_M*(1+(pow((D1/L),0.666667)))*pow(PHI_I,n))/ (2+
(17.96*pow(CFI,0.5)*(pow(PRI_M,0.666667)-1)));

HI=(NUI*LAMIV_M)/D1;

return ( HI );

}

***** "H_bif_Kand" retorna el valor del coeficiente de conveccion para el agua en estado
bifasico

segun la formulacion de Kandlikar (coeficiente promedio, no usa mapas de flujo)****/

float H_bif_Kand(float PIM,float TIM,float q,float x2new,float GI, float D1){

float H_bif;

double PRIL_M,NUI,VISCIL_M,LAMIL_M,REIL;

float alfanb,alfacb;

float c5,Ffl,Bo,Co,Frl,Hls;

float hfg;

float g;

g=9.81;

hfg=enthalpyvap(TIM);

float RHOIL_M,RHOIV_M;

RHOIL_M=RHOIL(TIM);

RHOIV_M=RHOIV(TIM,PIM);

Bo=q/(GI*hfg*1000);

Co=(pow((1-x2new)/x2new,0.8)*pow(RHOIV_M/RHOIL_M,0.5));

Frl=pow(GI/RHOIL_M,2)/(g*D1);

```



```
PRIL_M=PRIL(TIM);
VISCIL_M=VISCIL(TIM);
REIL=GI*D1/VISCIL_M;
LAMIL_M=LAMIL(TIM);
Ffl=1.0;

NUI=0.023*pow(REIL,0.8)*pow(PRL_M,0.4);

Hls=(NUI*LAMIL_M)/D1;

if (FrI>0.04) {
    c5=0;
} else if (!(FrI>0.04)) {
    c5=0.3;
} else assert(0);

if (Co<0.65) {
    alfabcb=0.6683*pow(Co,-0.2)*pow(1-
x2new,0.8)*Hls*pow(25*FrI,c5)+1058*
pow(Bo,0.7)*pow(1-x2new,0.8)*Hls*Ffl;
    H_bif=alfacb;
} else if (!(Co<0.65)) {
    alfanb=1.136*pow(Co,-0.9)*pow(1-
x2new,0.8)*Hls*pow(25*FrI,c5)+667.2*
pow(Bo,0.7)*pow(1-x2new,0.8)*Hls*Ffl;
    H_bif=alfanb;
} else assert(0);

return (H_bif);
}

***** "H_bif_Katt" retorna el valor del coeficiente de convección para el agua en estado
bifásico
```



```
segun la formulacion de Kattan,Thome and Favrat (usa mapas de
flujo)*****
float H_bif_Katt(float PIM,float TIM,float q,float x2new,float GI, float D1,float epsg,
float anglestrat,float Gwavy,float Gstrat, float classif){
double alphatp;
double PRIL_M,LAMIL_M,VISCIL_M;
double VISCIV_M,CPV_M,LAMIV_M;
double alphav,alphawet,alphanb,alphacb;
float angledry;
float delta;
angledry=anglestrat*((Gwavy-GI)/(Gwavy-Gstrat));
LAMIL_M=LAMIL(TIM);
PRIL_M=PRIL(TIM);
VISCIL_M=VISCIL(TIM);
VISCIV_M=VISCIV(TIM,PIM);
LAMIV_M=LAMIV(TIM);
CPV_M=1870;
if (angledry>=0.0) {

} else if (angledry<0.0) {
    angledry=0.0;
} else assert (0);
if (classif==6.0) {
    epsg=epsg-0.05;
} else if (!(classif==6.0) ) {
    } else assert (0);
delta=(PI*D1*(1-epsg))/(2*(2*PI-angledry));
```



```

float Pr;
float Pc;
Pc=22088.85; /*valor medio para la presion critica del agua*/
Pr=PIM/Pc; /*Pr es la presion reducida*/
float Mmol;
Mmol=18.02; /*M es el peso molecular del agua*/
/*alphanb es el coeficiente de conveccion de la parte ebullición nucleada*/
alphanb=55*pow(Pr,0.12)*pow(-log10(Pr),-0.55)*pow(Mmol,-0.5)*pow(q,0.67);
/*alphacb es el coeficiente de conveccion de la ebullición convectiva*/
alphacb=0.0133*pow(PRIL_M,0.4)*(LAMIL_M/pow(delta,0.8))*pow((4*GI*(1-
x2new)*delta)/((1-epsg)*VISCIL_M),0.69);
alphawet =pow((pow(alphanb,3)+pow(alphacb,3)),0.33);
/*alphav es el coeficiente de conveccion del estado vapor*/
alphav=0.023*pow((GI*x2new*D1)/(epsg*VISCIV_M),0.8)*pow
(CPV_M*VISCIV_M/LAMIV_M,0.4)*(LAMIV_M/D1);
alphatp=(angledry*alphav+(2*PI-angledry)*alphawet)/(2*PI);
return alphatp;
}

#####
#####*/
*****FUNCIONES DE LAS PÉRDIDAS DE PRESIÓN******/
#####
#####*/
float DeltaPS (float DHS,float L,float TI1,float TS1,float TS2new,float GS,float TSW){
    float DPS,PHI_S,RES,p,RHOS_M,Cf,DTS,TSM;
    DTS=TS1-TI1;
}

```



```

TSM=(TS1+TS2new)/2.0;

PHI_S=VISCS(TSM)/VISCS(TSW);

if (DTS>=0.0) {

    p=0.38;

} else if (DTS<0.0) {

    p=0.52;

} else assert (0);

RHOS_M=RHOS(TSM);

RES=GS*DHS/VISCS(TSM);

Cf=1.0/pow((1.58*log(RES)-3.28),2);

DPS=2.0*Cf*pow(PHI_S,p)*L*GS*GS/(DHS*RHOS_M);

return (DPS/1000);

}

float DeltaPI (float DI,float L,float TI1,float TS1,float TI2new,float GI,float TIW){

    float DPI,PHI_I,REI,p,RHOI_M,Cf,DTI,TIM;

    DTI=TS1-TI1;

    TIM=(TI1+TI2new)/2.0;

    PHI_I=VISCIL(TIM)/VISCIL(TIW);

    if (DTI>=0.0) {

        p=-0.33;

    } else if (DTI<0.0) {

        p=-0.24;

    } else assert (0);

    RHOI_M=RHOIL(TIM);

    REI=GI*DI/VISCIL(TIM);

```



```
Cf=1.0/pow((1.58*log(REI)-3.28),2);  
DPI=2.0*Cf*pow(PHI_I,p)*L*GI*GI/(DI*RHOI_M);  
return (DPI/1000);  
}  
  
float DeltaPIvapor (float PIM,float DI,float L,float TI1,float TS1,float TI2new,  
float GI,float TIW){  
    float DPlv,PHI_I,REI,p,RHOI_M,Cf,DTI,TIM;  
    DTI=TS1-TI1;  
    TIM=(TI1+TI2new)/2.0;  
    PHI_I=VISCIV(TIM,PIM)/VISCIV(TIW,PIM);  
    if (DTI>=0.0) {  
        p=0.52;  
    } else if (DTI<0.0) {  
        p=0.38;  
    } else assert (0);  
    RHOI_M=RHOIV(TIM,PIM);  
    REI=GI*DI/VISCIV(TIM,PIM);  
    Cf=1.0/pow((1.58*log(REI)-3.28),2);  
    DPlv=2.0*Cf*pow(PHI_I,p)*L*GI*GI/(DI*RHOI_M);  
    return (DPlv/1000);  
}  
float DeltaPI_bif(float PIM,float DI,float TI1,float TI2new,float GI,float x2new){  
    float PHI,Rel,Reg,RHOI_H,RHOI_L,RHOI_V,TIM,VISCI_L,VISCI_V;  
    float g,sigma,Cfg,Cfl;  
    float Fr,E,F,H,We;
```





```
*****MAPA DE FLUJO SEGUN YEHUDA TAITEL *****/  
******/  
float flowp(float PIM,float TIM,float Aint,float GI,float MI1,float D1,tuplanl h,int i){  
float eps_g;  
float classif;  
float dc,dcd,dcb;  
float Ug,UI,Uls,Ugs,Um,Q,QI,Qg;  
float Al,Ag;  
double k,fm,RHOIL_M,RHOIV_M,VISCIL_M;  
float Cm,n;  
float eps_dg,g,sigma;  
eps_g=h.tabla[i];  
g=9.81;  
sigma=0.0595;  
classif=0.0;  
RHOIV_M=RHOIV(TIM,PIM);  
  
RHOIL_M=RHOIL(TIM);  
VISCIL_M=VISCIL(TIM);  
Q=MI1/RHOIL_M;  
Al=Aint*(1-eps_g);  
Ag=Aint*eps_g;  
UI=Q/Aint;  
Ug=GI/RHOIV_M;  
Cm=0.046;  
n=-0.2;
```



```

QI=UI*AI;
Qg=Ug*Ag;
UIs=QI/Aint;
Ugs=Qg/Aint;
Um=UIs+Ugs;

eps_dg=Ugs/Um;
fm=Cm*pow((4*Aint*Um/(PI*D1*VISCIL_M/RHOIL_M)),n);
k=2*fm*pow(Um,3.0)/D1;
dc=(0.725+4.15*pow(eps_dg,0.5))*pow(sigma/RHOIL_M,3/5)*pow(k,-0.4);

cd=2*pow(0.4*sigma/((RHOIL_M-RHOIV_M)*g),0.5);
dcb=0.375*(RHOIL_M/(RHOIL_M-RHOIV_M))*fm*pow(Um,2)/g;
/*stratified flow****/
float deltaAlhl;
float fl;
float UgKH;
float UIKH;
deltaAlhl= (h.tabla[i]-h.tabla[i-1])*PI*(D1/4);
UgKH=pow(((RHOIL_M-RHOIV_M)*g*Aint*eps_g)/(RHOIV_M*deltaAlhl),0.5);
fl=Cm*pow((4*Aint*(1-eps_g)*UI/(PI*D1*(1-eps_g)*VISCIL_M/RHOIL_M)),n);
UIKH=g*D1*(1-eps_g)/fl;
float UI2;
UI2= pow(UI,2.0);
/*annular flow*/
float Rmax;

```



```
Rmax=0.48;  
  
float bloccrit;  
  
bloccrit=(1.0-eps_g)/Rmax;  
  
if(dc<dcd && dc<dcb && (eps_g)<0.52) {  
    classif=1.0; /*bubble*/  
  
} else if (!(dc<dcd && dc<dcb && (eps_g)<0.52)) {  
  
    if (Ug<UgKH && pow(UI,2.0)<UIKH) {  
  
        classif=2.0; /*stratified*/  
  
    } else if (!(Ug<UgKH && UI2<UIKH)) {  
  
        if (bloccrit>0.5) {  
  
            classif=3.0; /*annular*/  
  
        } else if (!(bloccrit>0.5)) {  
  
            classif=4.0; /*intermittent*/  
  
        } else assert(0);  
  
    } else assert(0);  
  
} else assert(0);  
  
return classif;  
}  
  
*****MAPA DE FLUJO SEGUN KATTAN-THOME-FAVRAT*****  
*****  
  
void flowp2 (float PIM,float TIM,float q,float x,float GI,float D1,float eps_g,float *const  
Gwavy,float *const Gstrat,  
  
float *const anglestrat,float *const classif){  
  
float eps_l;  
  
eps_l=1.0-eps_g;  
  
float hfg;  
  
hfg=enthalpyvap(TIM);
```



```

double compangstrat;
double Ald,Avd;
double hld,Pid;
double WeFrl;
double qcrit,F1q,F2q;
double Xia,Gmist,Gbubbly;
double epsilon;
*anglestrat=2*PI-2*((PI*eps_I)-(1/200)*eps_I*eps_g*(1-
2*eps_I)*(1+4*(pow(eps_I,2)+pow(eps_g,2)))+
(1.676*(1.0-2*eps_I+pow(eps_I,0.333)-pow(eps_g,0.333))));

compangstrat=(2*PI)- (*anglestrat);
Ald=(0.125)*( compangstrat-sin(compangstrat));
Avd=(PI/4.0)-Ald;
hld=0.5*(1-cos(compangstrat/2.0));
Pid=sin(compangstrat/2.0);
double RHOIL_M,RHOIV_M,VISCIV_M,VISCIL_M;
float g,sigma;
g=9.81;
sigma=0.0595;
*classif=0.0;
RHOIV_M=RHOIV(TIM,PIM);
VISCIV_M=VISCIV(TIM,PIM);
RHOIL_M=RHOIL(TIM);
VISCIL_M=VISCIL(TIM);
WeFrl=g*pow(D1,2)*RHOIL_M/sigma;
qcrit=0.131*pow(RHOIV_M,0.5)*hfg*1000*pow(g*(RHOIL_M-RHOIV_M)*sigma,0.25);
F1q=646.0*pow((q/(2*qcrit)),2)+64.8*q/(2*qcrit);

```



```

F2q=18.8*q/(2*qcrit)+1.023;

*Gwavy=pow((16*pow(Avd,3.0)*g*D1*RHOIL_M*RHOIV_M /(pow(x*Pl,2)-
pow(1.0-pow(2*hld-1.0,2.0),0.5)))*(0.04*pow(Pl/hld,2)*pow(1-x,-F1q)*
pow(WeFrI,-F2q)+1.0),0.5)+50-75*exp(-pow(pow(x,2)-0.97,2)/(x*(1-x)));

Xia=pow((0.2914*pow(RHOIV_M/RHOIL_M,-0.34)*pow(VISCIL_M/VISCIV_M,-
0.14))+1.0,-1.0);

epsilon=pow(1.138+2*log10(Pl/(1.5*Ald)), -2.0);

Gmist=pow(7680*pow(Avd,2.0)*g*D1*RHOIL_M*RHOIV_M/(pow(x*Pl,2.0)*epsilon*WeFrI),
0.5);

*Gstrat=pow(pow(226.3*Avd/x,2.0)*Ald*RHOIV_M*(RHOIL_M-
RHOIV_M)*g*VISCIL_M/((1-x)*pow(Pl,3.0)),0.333)+20*x;

Gbubbly=pow(256*Avd*pow(Ald/Pl,2.0)*pow(D1,1.25)*RHOIL_M*(RHOIL_M-
RHOIV_M)*g/(0.314*pow(1.0-x,1.75)*Pid*pow(VISCIL_M,0.25)),0.34);

/* classifiquem el bifasic: bubbly=1.0; stratified=2.0; stratified wavy=3.0;
intermittent =4.0; annular=5.0; mist flow=6.0*/
if (GI<*Gstrat) {
    *classif=1.0; /*bubbly*/
    } else if (!(GI<*Gstrat)) {
        if (GI<Gbubbly) {
            *classif=2.0; /*stratified*/
            } else if (!(GI<Gbubbly)) {
                if (GI<*Gwavy) {
                    *classif=3.0; /*stratified wavy*/
                    } else if (!(GI<*Gwavy)) {
                        if (GI<Gmist && x<Xia) {
                            *classif=4.0; /*intermittent*/
                            } else if (GI<Gmist && x>Xia) {

```



```

*classif =5.0; /*annular*/
} else if (GI>Gmist) {
    *classif =6.0; /*mist flow*/
} else assert(0);
} else assert(0);
} else assert(0);
} else assert(0);
}

*****OTRAS FUNCIONES*****
*****OTRAS FUNCIONES*****

```

```

float Tempsat(float P)
{
    return((3816.44/(-log((P*760)/101.3)+18.3036))+46.13 );
}

float FraccioVol(float PIM,float TIM,float x2new,float GI){
    double epsilon;
    double g,sigma,xg,xl,RHOIV_M,RHOIL_M;
    xg=x2new;
    xl=1.0-xg;
    RHOIV_M=RHOIV(TIM,PIM);
    RHOIL_M=RHOIL(TIM);
    g=9.81;
    sigma=0.0595;
    epsilon=(xg/RHOIV_M)*pow((1+0.12*xl)*((xg/RHOIV_M)+(xl/RHOIL_M))+((1.18*xl*pow(g*sigma*(RHOIL_M-RHOIV_M),0.25))/(GI*pow(RHOIL_M,0.5))),-1.0);
}

```



```
return epsilon;
```

```
}
```

```
float enthalpyvap(float T){
```

```
float TC;
```

```
float y;
```

```
TC=T-273.15;
```

```
y=1.5351*TC+2513.4;
```

```
return(y);
```

```
}
```

```
bool bifasico (float P,float T){
```

```
float Taux;
```

```
Taux=Tempsat(P);
```

```
return( T>Taux);
```

```
}
```

```
bool estado_vapor (float x,float VI1){
```

```
float xaux;
```

```
if (VI1<0.6) {
```

```
    xaux=0.95;
```

```
} else if (VI1>=0.6 && VI1<0.7){
```

```
    xaux=0.97;
```

```
} else if (VI1>=0.7){
```

```
    xaux=0.99;
```

```
} else assert (0);
```

```
return(x>xaux);
```

```
}
```



```

void MetodoMLDT (float D0, float DI, float L,float A0t,float ef,float Af, float LAMW, float TI1,
float TS1, float HS,
float HI,float CPS, float MI, float MS,float TS2,float hfg,float x1,
float *const TS2new,float *const TSWnew,float *const TIWnew,float *const x2new,
float *const qnova){

float TSM,RI,RS,RW,UA;
float increTI,increTF,Tmldt;
    RI= 1/(HI*DI*L*PI);
    RS=1/(HS*(A0t+ef*Af));
    RW=log(D0/DI)/(2*PI*LAMW*L);
    UA= 1/(RI+RS+RW);
increTI=TS1-TI1;
increTF=TS2-TI1;
Tmldt=((increTI-increTF)/(log(increTI/increTF)));
*qnova=UA*Tmldt;
*TS2new=TS1-(*qnova/(MS*CPS));
*x2new=(*qnova/(1000*hfg))+(x1*MI)/MI;
TSM=(TS1+*TS2new)/2.0;
*TIWnew=TSM-(*qnova)*(RS+RW);
*TSWnew=TSM-(*qnova)*(RS);
}

void MetodoPRN (float D0, float DI, float L,float A0t,float ef,float Af, float LAMW, float TI1,
float
TS1, float HS, float HI, float CPI, float CPS, float MI, float MS, float *const TS2new,
float *const TI2new, float * const TSWnew,float *const q,float *const R,float *const
N,float *const P,float *const TIWnew,float *const RI,float *const

```



```
RW,float *const RS,float *const UA,float *const Cc,float *const Ch,float *const DTI)

{

float TSM ;

*RI= 1/(HI*DI*L*PI);

*RS=1/(HS*(A0t+ef*Af));

*RW=log(D0/DI)/(2*PI*LAMW*L);

*UA= 1/(*RI+*RS+*RW);

*DTI=TS1-TI1;

if (*DTI>=0) {

    *Cc=CPI*MI;

    *Ch=CPS*MS;

    *N=*UA/(*Cc);

    *R= *Cc/(*Ch);

    *P= (1.0-exp(-(*N)*(*R+1.0)))/(*R+1.0);

} else if (*DTI<0){

    *Cc=CPS*MS;

    *Ch=CPI*MI;

    *N=*UA/(*Ch);

    *R= *Ch/(*Cc);

    *P= (1.0-exp(-(*N)*(*R+1.0)))/(*R+1.0);

}

*TI2new=TI1+(TS1-TI1)*(*P);

*TS2new=TS1-(*TI2new-TI1)*(*R);

TSM=(TS1+*TS2new)/2.0;

*q=MS*CPS*(TS1-*TS2new);

*TIWnew=TSM-(*q)*(*RS+*RW);
```



*TSWnew=TSM-(*q)*(*RS);

}

