PRACTICA No 4

Objetivo general: Que el alumno aplique sus conocimientos y en especial los de estructuras de decisión if else ya sea en su forma anidada o en cascada para generar dos programa donde se observe la utilidad de estas ultimas, además de los ciclos de repetición generados con el comando while.

Desarrollo:

El primer programa llamado divisor de voltaje se realizara mediante la siguiente metodología; con la siguiente formula:

$$Vsal = Vent\left[\frac{R_1 * R_2}{R_1 + R_2}\right]$$

Las condiciones son las siguientes:

R1 y R2 deben estar entre los 10 KΩ y 1 MΩ
Vsal varia respecto a la temperatura, pero solo se tomara la muestra cuando la temperatura este entre los 20 °C y 27 °C
Solo se tomaran muestras los días pares de la semana
Por ejemplo
1.- Domingo
2.- Lunes

Por lo tanto tenemos que despejar la formula para obtener el voltaje de entrada por lo que siendo así:

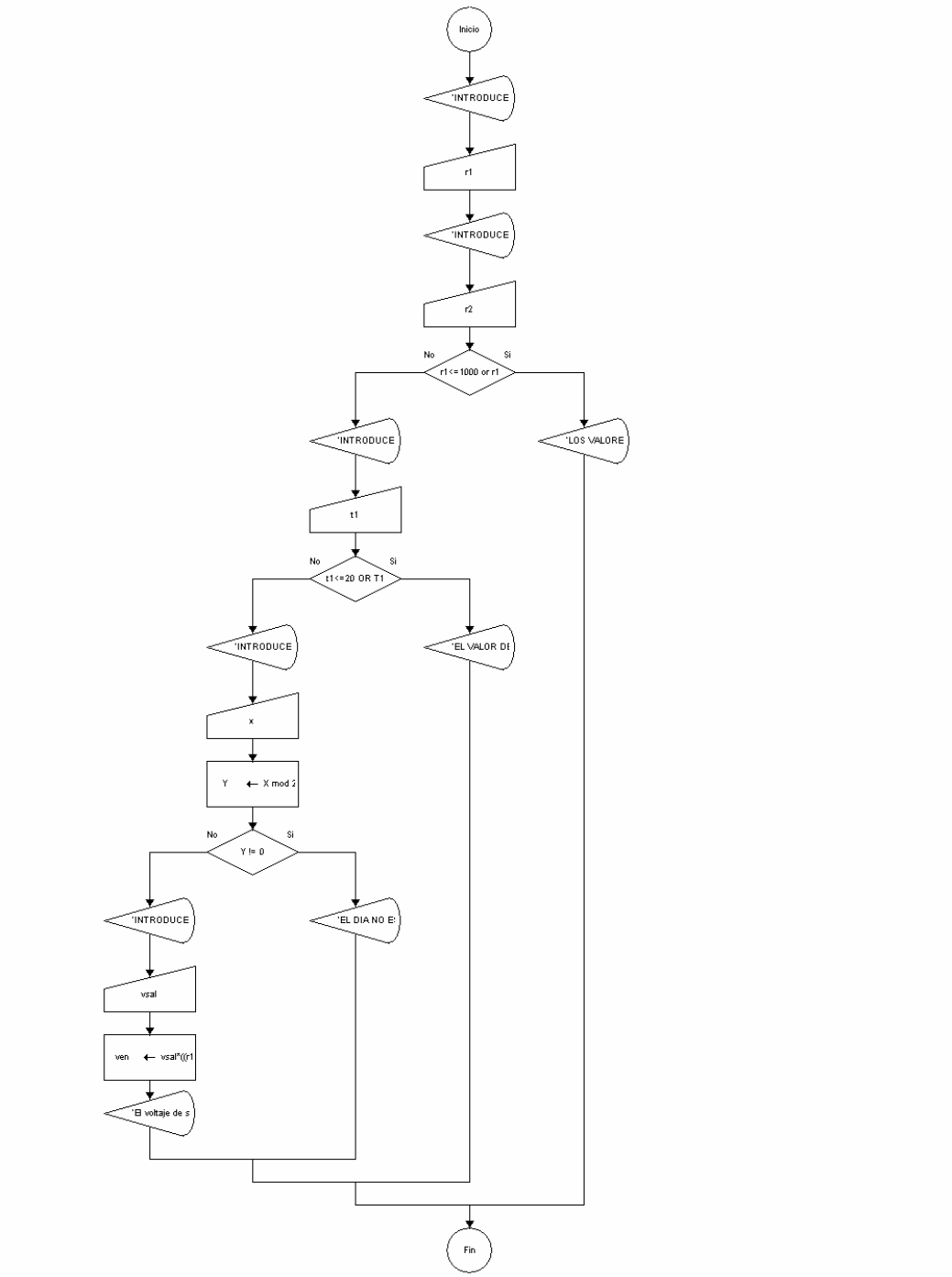
$$Vent = Vsal * \left[\frac{R_1 + R_2}{R_1 * R_2}\right]$$

Ahora si podemos realizar el programa quedando de la siguiente manera el codigo fuente:

```
/*
FECHA: 12-SEP-06
PROGRAMA # 13
OBJETIVO: LOGRAR DESARROLLAR UN PROGRAMA
CON LAS CONDICIONES ESTABLECIDAS USANDO
PARA ELLO ESTRUCTURAS DE DECISIÓN QUE PERMITAN
MAS DE DOS BIFURCACIONES RESPECTO A LAS VARIABLES USADAS*/
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main ()
{
int x;
float r1,r2,t1,ven,vsal,y;
clrscr ();
printf ("introduce el valor de r1\n");
scanf ("%f",& r1);
printf ("introduce el valor de r2\n");
scanf ("%f",& r2);
if (r1<=1000||r2>=1000000 || r2<=1000 || r2>=100000 )
```

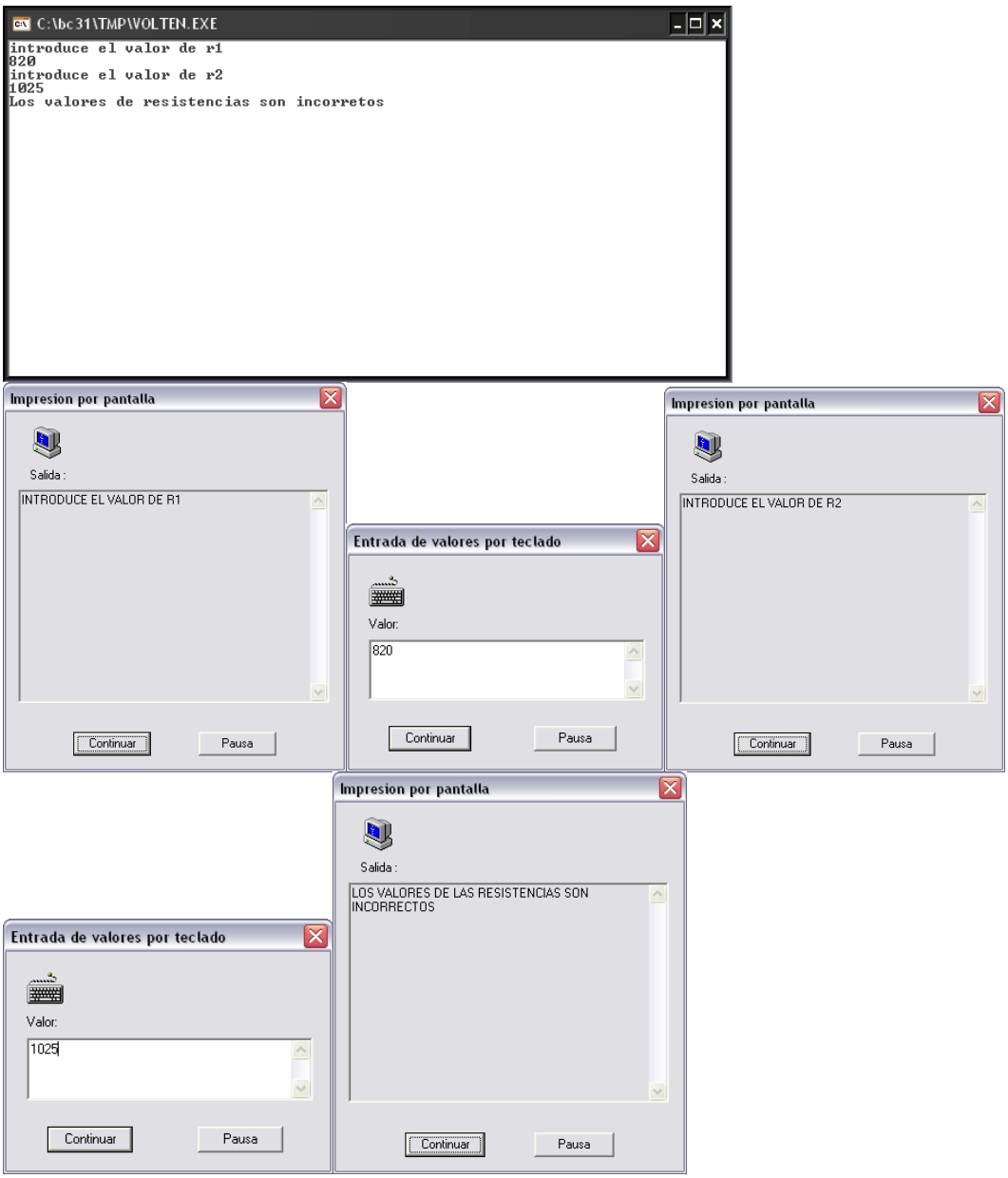
```
printf ("Los valores de resistencias son incorretos\n");
else
   {
   printf ("introduce el valor de la temperatura\n");
   scanf ("%f",& t1);
   if (t1<=20 || t1>=27)
     printf ("La temperatura no es ideal\n");
     else
            {
            printf ("introduce el numero del dia segun corresponda\n");
            printf("domingo=1 lunes=2 martes=3 miercoles=4 jueves=5 viernes=6 sabado=7\n");
            scanf ("%d",& x);
            y=x % 2;
            if (y != 0)
            printf ("El dia no es correcto\n");
            else
               {
               printf ("introduce el voltaje de salida\n");
               scanf ("%f",& vsal);
               ven=vsal*((r1+r2)/(r1*r2));
               printf ("El voltaje de entrada es:%f", ven);
               }
            }
   }
getch ();
}
/*CONCLUSIONES: EN ESTE PROGRAMA USAMOS
UNA ESTRUCTURA DE DECISIÓN IF ELSE DEL
TIPO ANIDADO PERO CADA PROGRAMADOR
TIENE SU ESTILO Y ESTE PROGRAMA SE PUEDE HACER
DE DISTINTAS FORMAS QUE REALIZAN LA MISMA FUNCION*/
```

El diagrama de flujo es el siguiente:

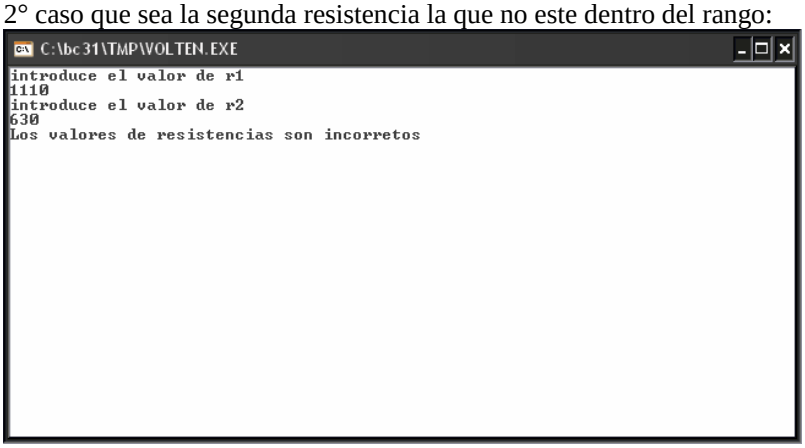


Para comprobar que realmente funciona debemos hacer las pruebas de escritorio siendo estas las primeras que las resistencias estén fuera del rango que se esta pidiendo por lo que aquí presento las dos posibilidades:

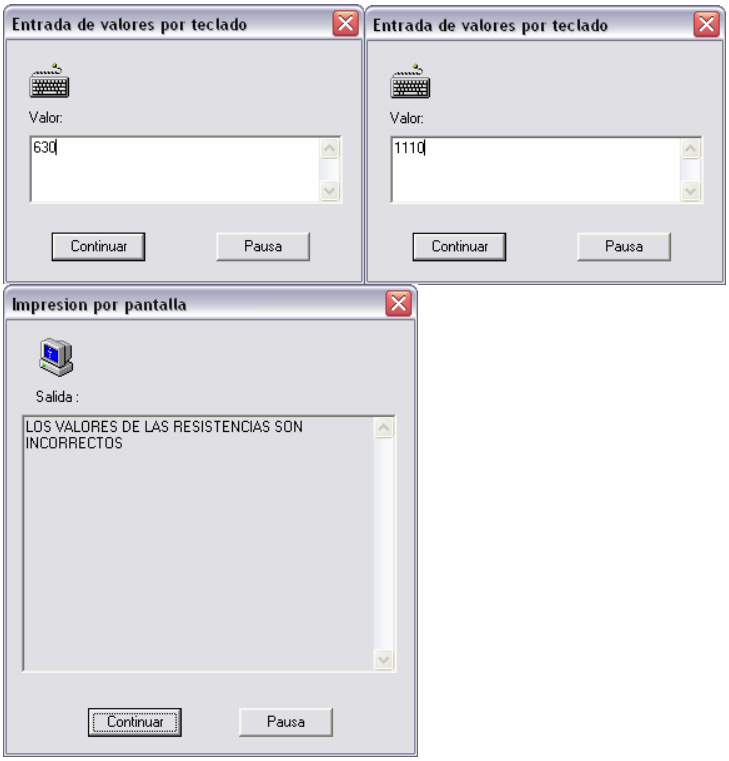
1° caso que sea la primera resistencia la que no este dentro del rango



2° caso que sea la segunda resistencia la que no este dentro del rango:

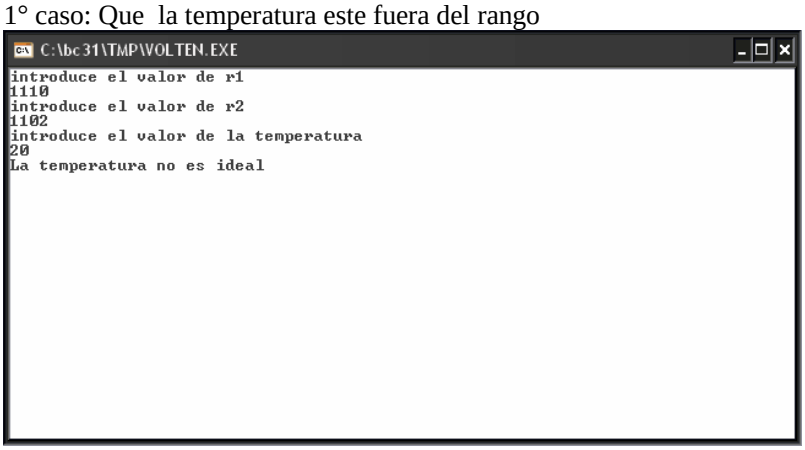


En este caso omitire los cuadros de dialogo que piden las resistencias y solo pondre los valores y el resultado:

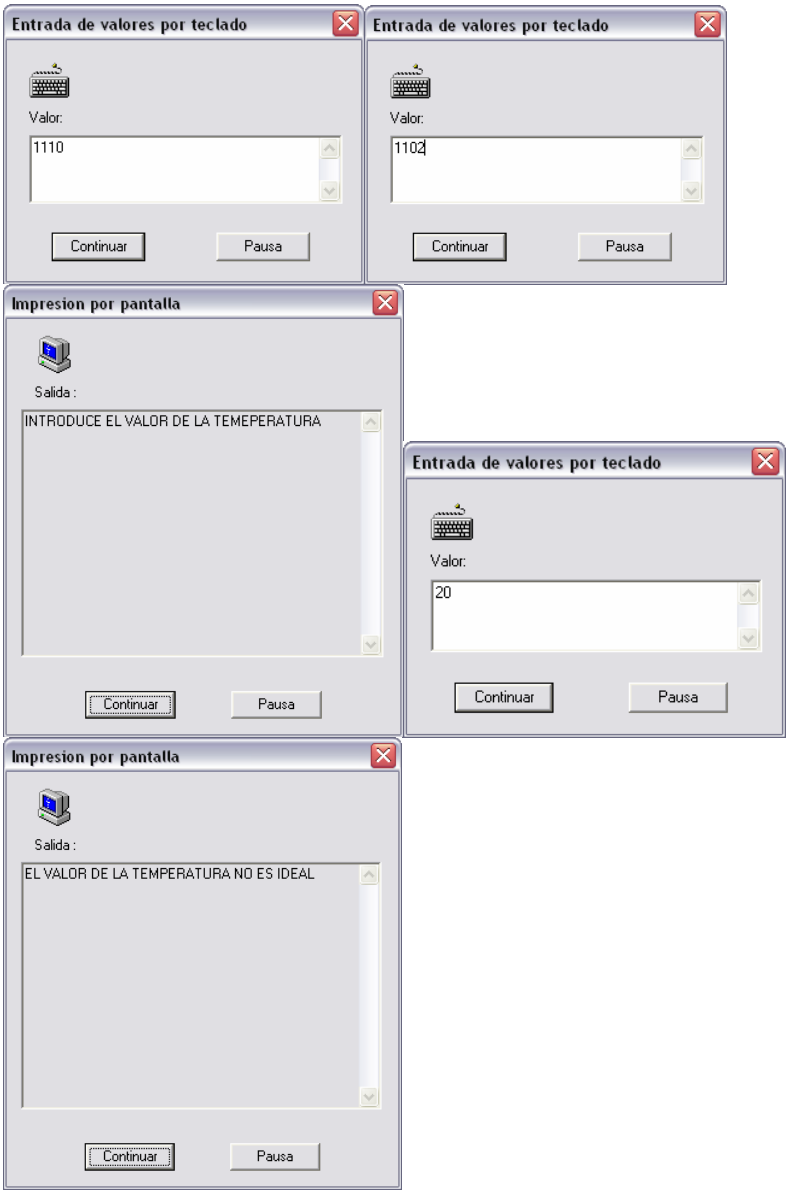


Ahora debemos comprobar que solo cuando ponemos una temperatura dentro del rango funciona:

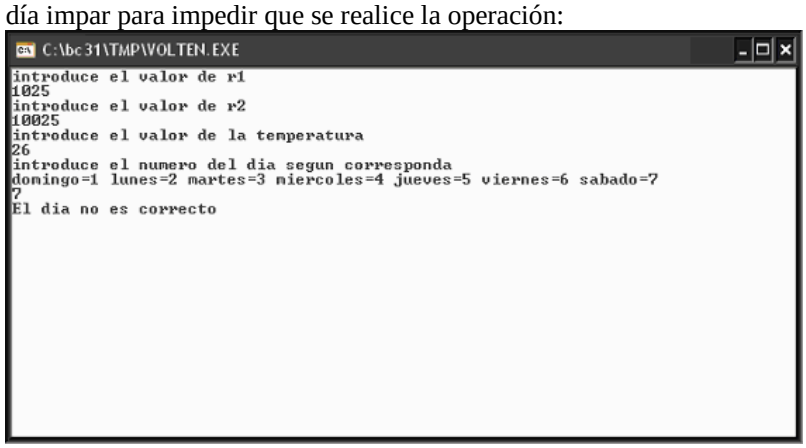
1° caso: Que la temperatura este fuera del rango



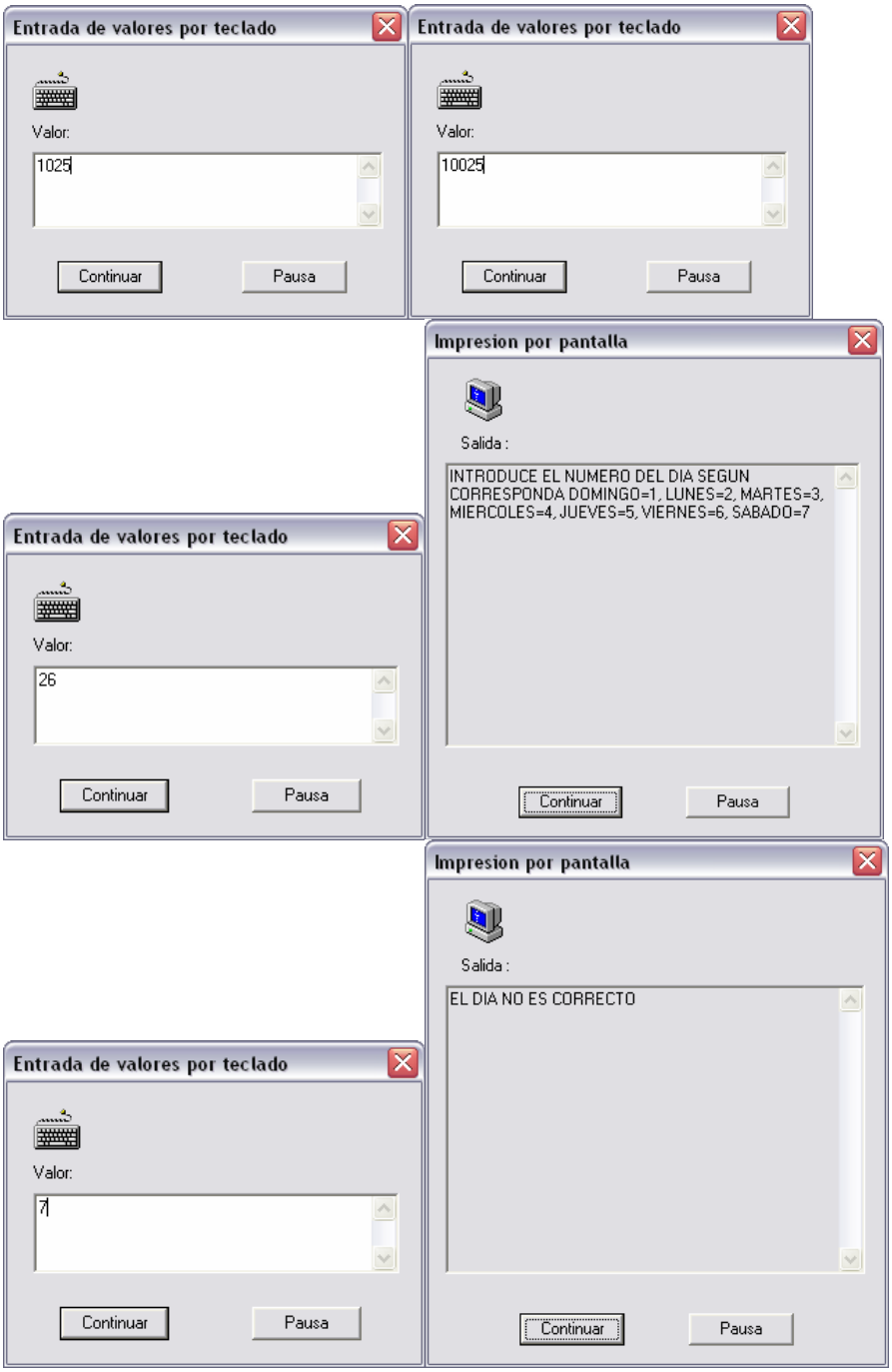
De nuevo para la corrida del dfd omitire los cuadros de dialogo que piden los valores de las resistencias poniendo solo estos



El segundo caso donde la temperatura es ideal lo mezclare con la última condición que sea un día impar para impedir que se realice la operación:



Ahora omitiré los cuadros de dialogo que piden primero el valor de R1 luego el que pide el valor de R2 y luego el de la temperatura solo poniendo en el mismo orden el de los valores que se introducen:

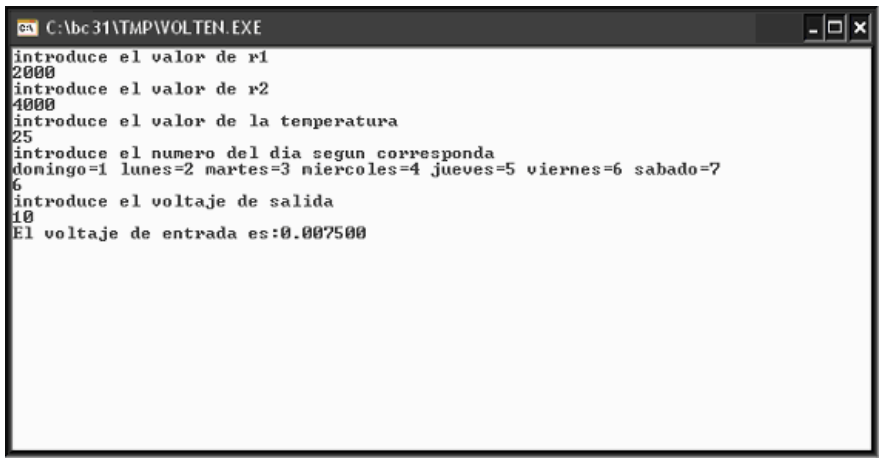


Por ultimo tenemos la prueba sobre todos los valores en rangos correctos y que el programa realice bien la operación por lo que daremos volres de 2 kΩ a R1 y de 4k Ω a R2 y un voltaje de salida de 10V.

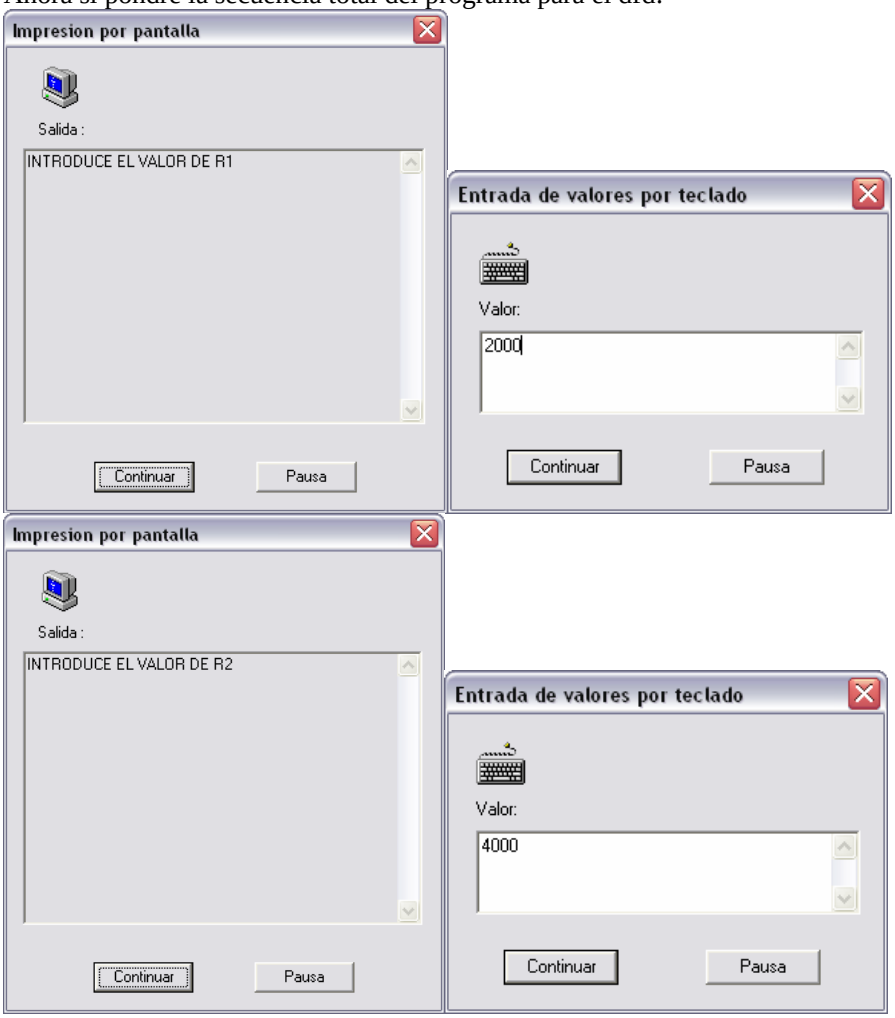
Por lo que la operación queda de la siguiente forma:

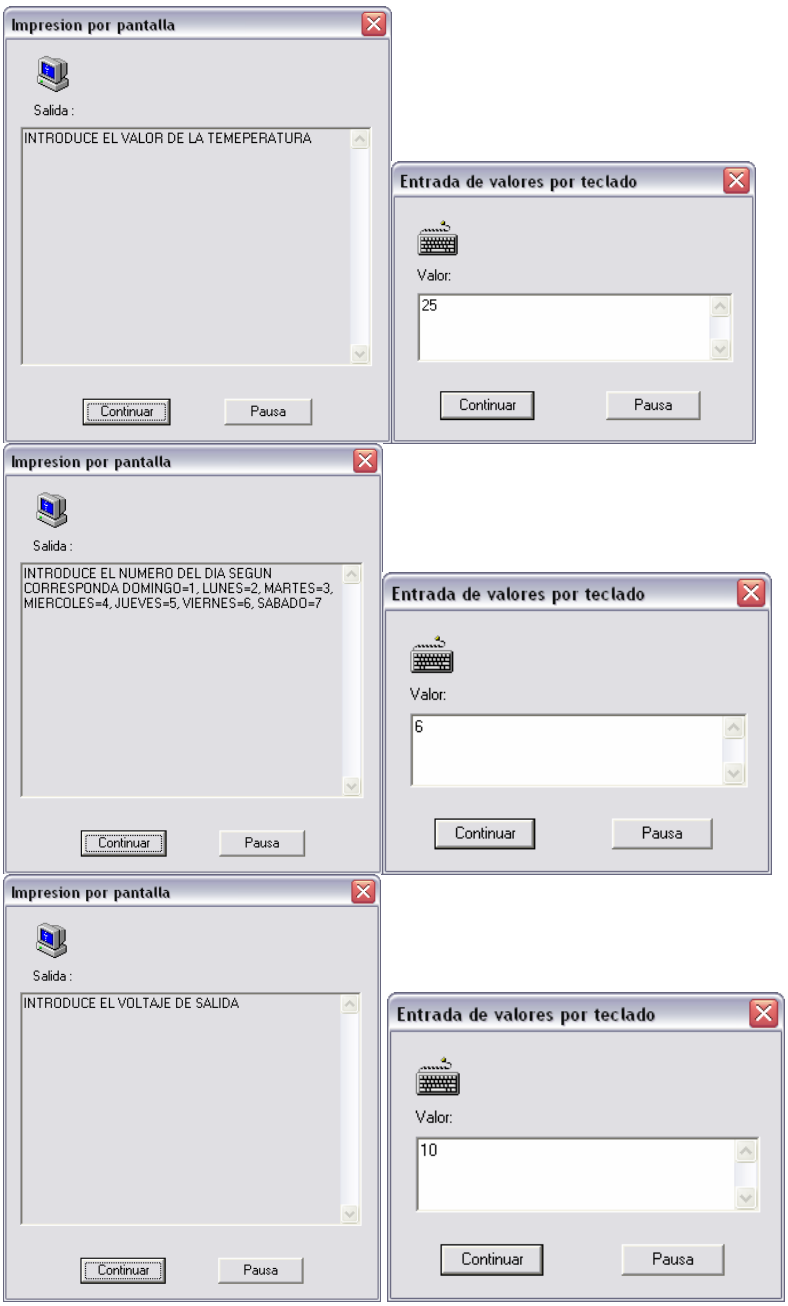
$$Vent = 10\left[\frac{2000+4000}{2000*4000}\right] = \quad Vent = 10\left[\frac{6000}{8000000}\right] = \qquad Vent = 10*.0075$$

Vent = .075



Ahora si pondre la secuencia total del programa para el dfd:




Impresion por pantalla
Salida :
INTRODUCE EL VALOR DE LA TEMEPERATURA
Continuar
Pausa
Entrada de valores por teclado
Valor:
25
Continuar
Pausa
Impresion por pantalla
Salida :
INTRODUCE EL NUMERO DEL DIA SEGUN CORRESPONDA DOMINGO=1, LUNES=2, MARTES=3, MIERCOLES=4, JUEVES=5, VIERNES=6, SABADO=7
Continuar
Pausa
Entrada de valores por teclado
Valor:
6
Continuar
Pausa
Impresion por pantalla
Salida :
INTRODUCE EL VOLTAJE DE SALIDA
Continuar
Pausa
Entrada de valores por teclado
Valor:
10
Continuar
Pausa

EL SEGUNDO PROGRAMA es también la resolución de una ecuación con algunas características por lo que ahora presento las condiciones y como tal no hay metodología a seguir mas que encontrar el valor de f(v):

La formula es explicita y solo hay que obtener f(v)

$$f(v) = Pv^3 - (Pb - RT)v^2 + av - ab$$

R y b son constantes definidas por el usuario (iguales).

P= $1*10^9$

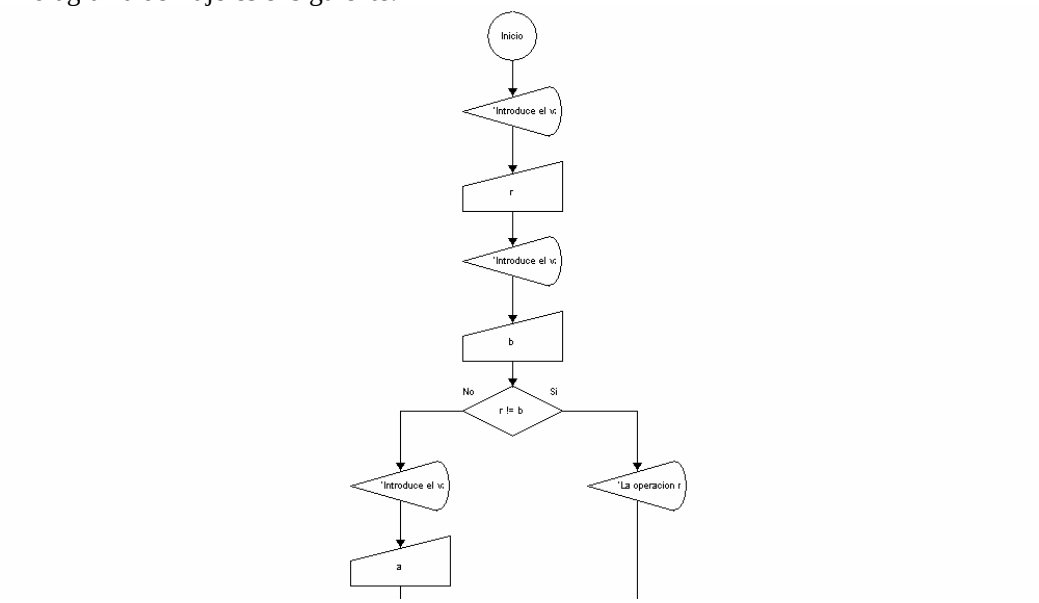
V= Debe ser diferente de cero para evaluar la condición.

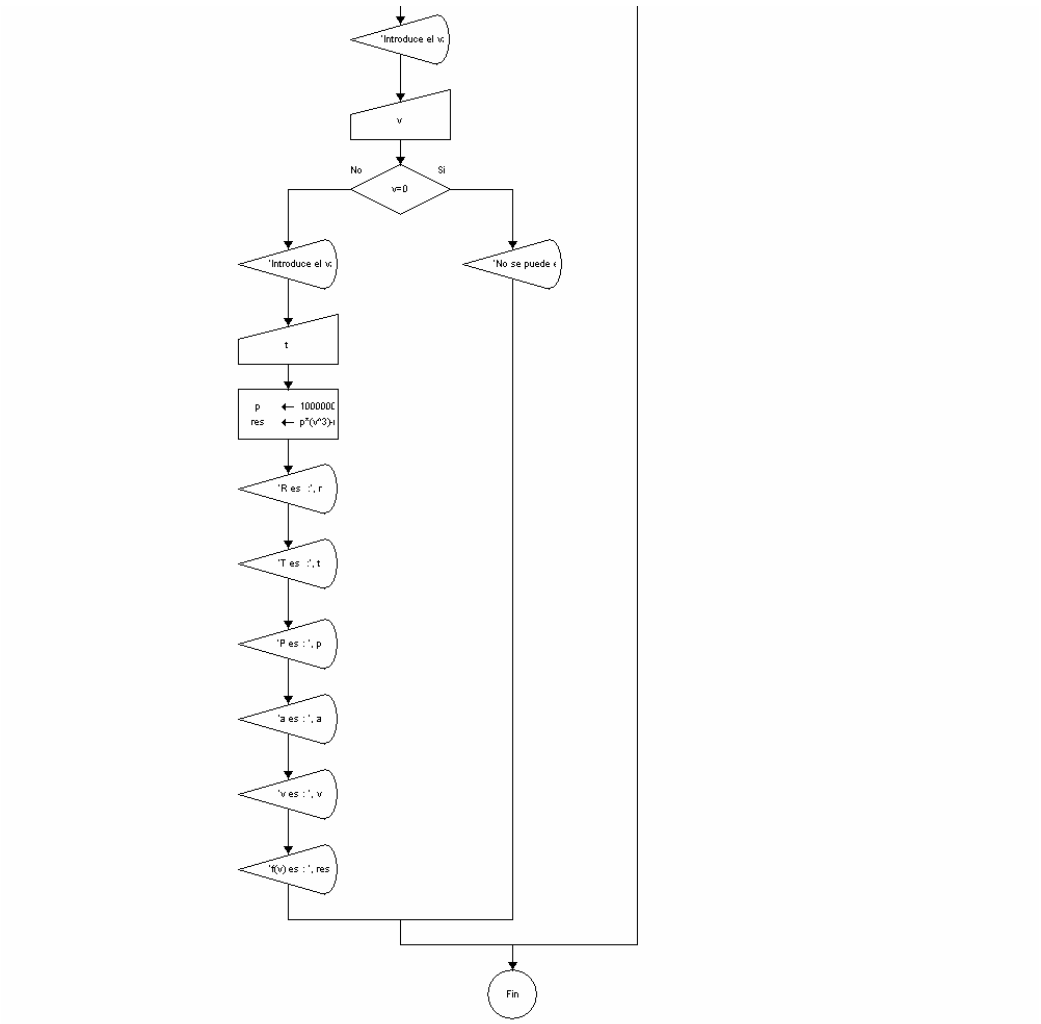
Por lo tanto el programa queda de la siguiente forma:

```
/*
FECHA: 12-SEP-06
PROGRAMA # 14
OBJETIVO: REALIZAR UN PROGRAMA USANDO BIFURCACIONES
CON LOS OPERADORES DE DECISIÓN YA SEA EN CASCADA O
ANIDADOS PARA LOGRAR OBTENER EL RESULTADO DE LA
ECUACIÓN DE VANDER WALLS, ADEMAS DE APRENDER A
MOSTRAR EL RESULTADO EN FORMA DE TABLAS*/
#include <stdio.h>
#include <conio.h>
#include <math.h>
main ()
{
const float p=1000000000;
float r,b,t,a,v,res;
clrscr ();
printf ("introduce el valor de R\n");
scanf ("%f", & r);
printf ("introduce el valor de b que debe ser igual a R\n");
scanf ("%f", & b);
if (r != b)
printf ("No se puede realizar la operacion por que R y b no son iguales\n");
else
   {
   printf ("introduce el valor de a\n");
   scanf ("%f", & a);
   printf ("introduce el valor de v\n");
   scanf ("%f", & v);
   if (v ==0)
```

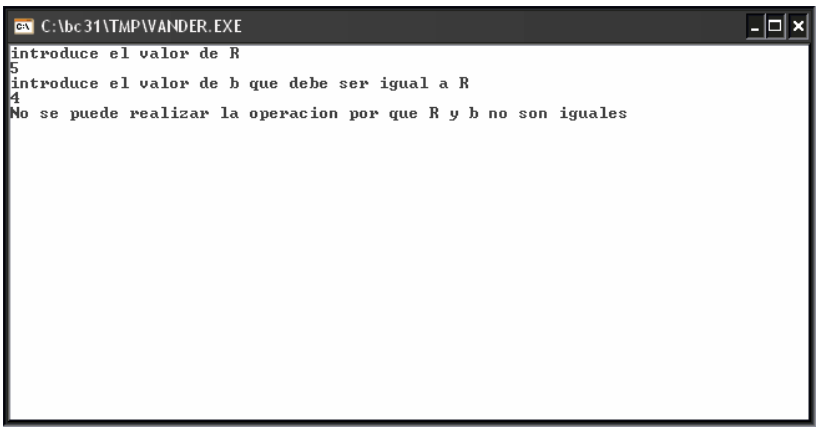
```
   printf ("No se puede evaluar la condicion\n");
   else
         {
         printf ("introduce el valor de T\n");
         scanf ("%f", & t);
         res=(p*pow (v,3))-(((p*b)+(r*t))*pow(v,2))+((a*v)-(a*b));
         printf ("R\tT\tP\t\ta\tv\tf(v)\n");
         printf ("%.2f\t", r);
         printf ("%.2f\t", t);
         printf ("%.2f\t", p);
         printf ("%.2f\t", a);
         printf ("%.2f\t", v);
         printf ("%.2f\n", res);
         }
   }
getch ();
}
/*CONCLUSIONES: APRENDIMOS QUE \T SIRVE PARA DAR ESPACIOS
LATERALES Y ESO NOS SIRVE PARA HACER LA TABLA ADEMAS DE
QUE PODEMOS SEÑALAR CUANTOS DECIMALES DESPUÉS DEL
PUNTO PUEDE TENER NUESTRO RESULTADO PONIENDO
ANTES DE LA INDICACION F DE PUNTO FLOTANTE EL NUMERO
DE DECIMALES COMO: .2%F, ADEMAS DE APLICAR LOS CONOCIMIENTOS SOBRE
LAS ESTRUCTURAS DE DECISIÓN*/
```

El diagrama de flujo es el siguiente:

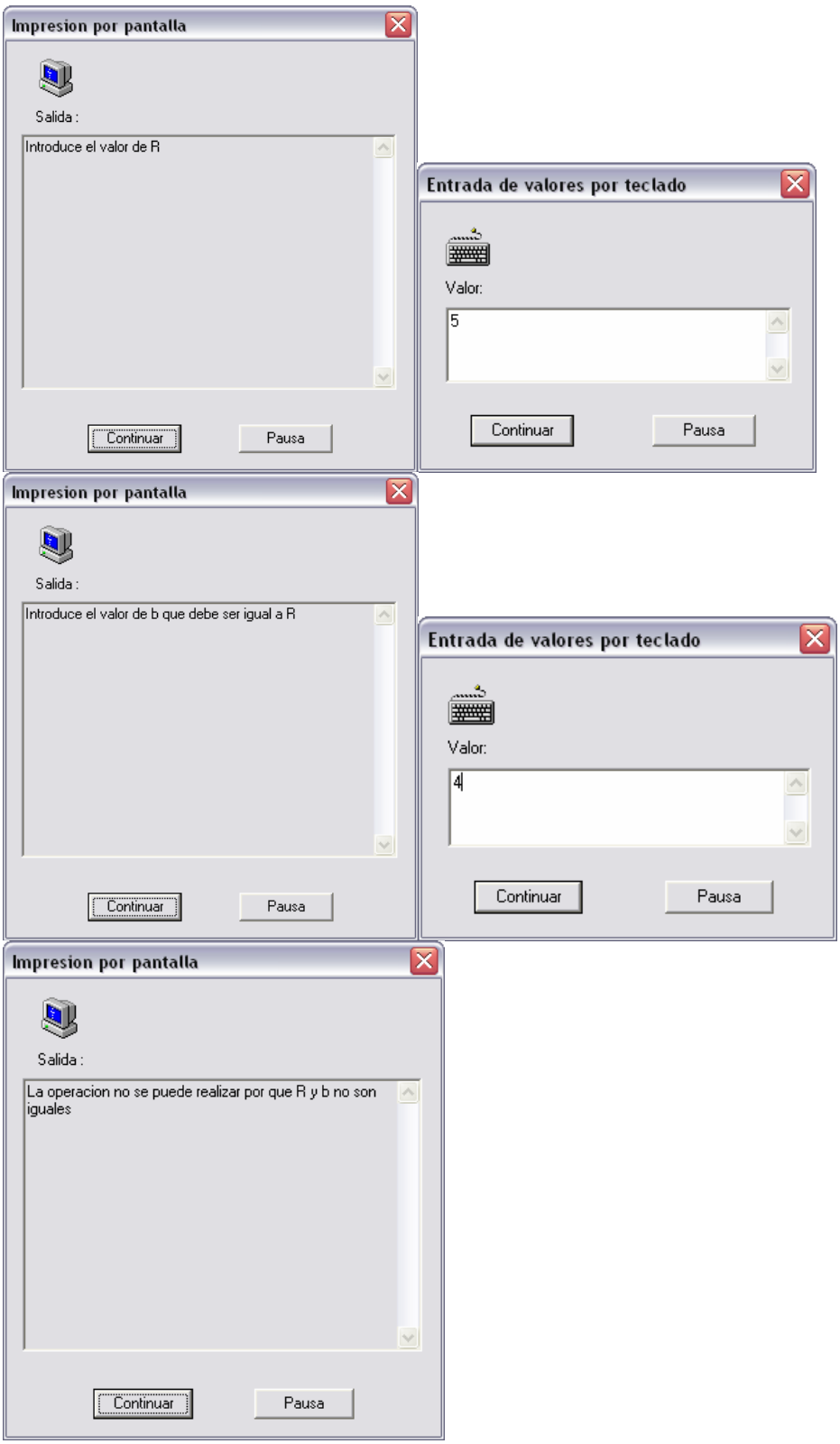




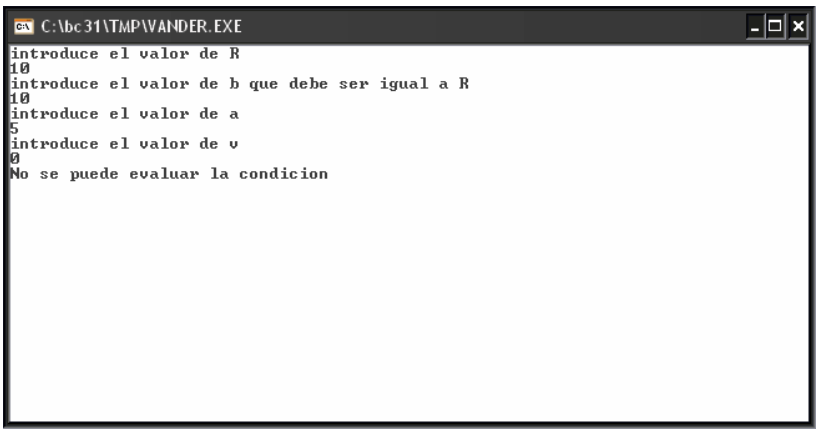
Las pruebas de escritorio están dadas a cumplir las condiciones y de obtener un resultado correcto por lo que la primera prueba es que si pones dos distintos numeros para R y para b te lo indique. Por lo que la corrida se ve asi:



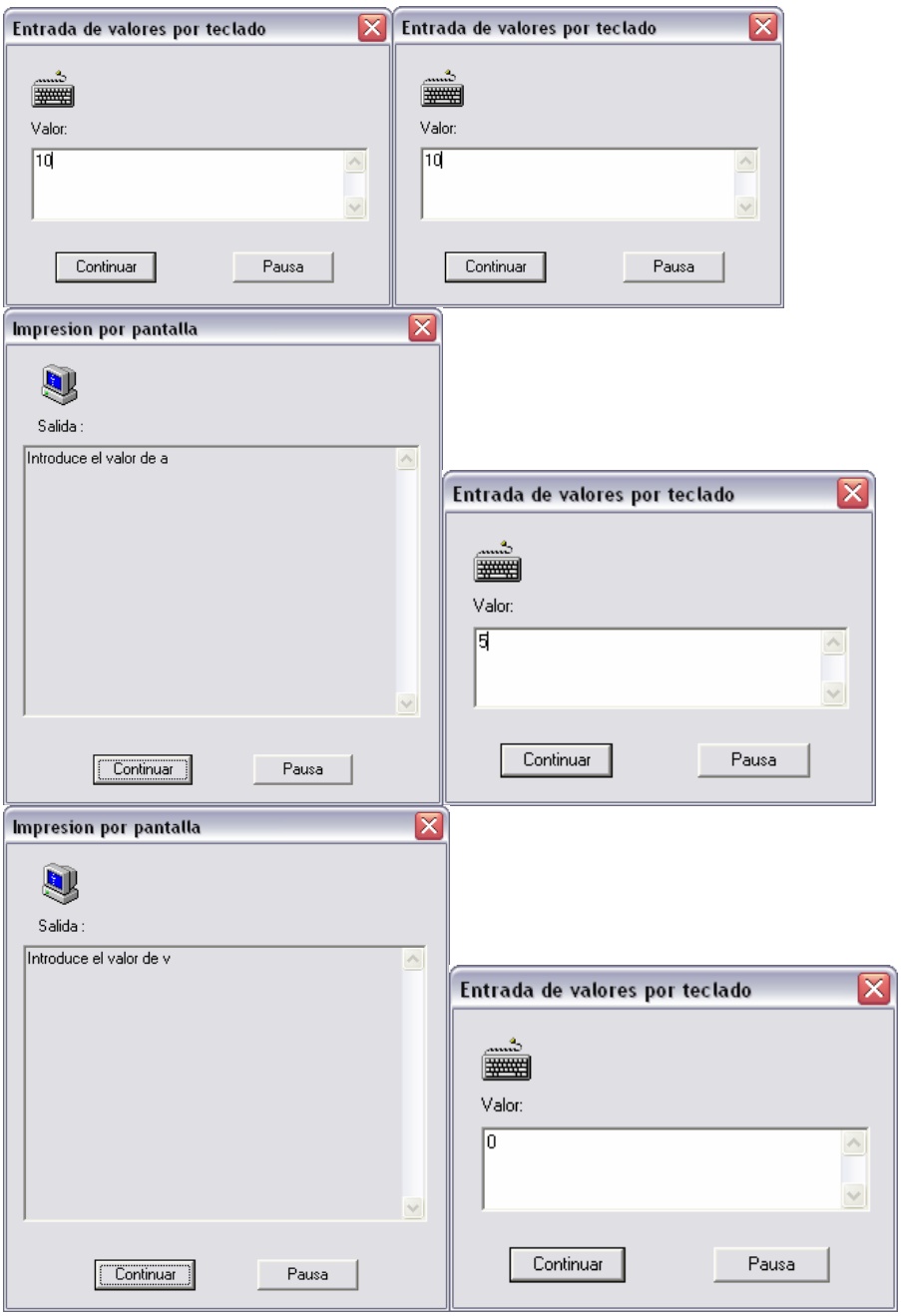
La corrida de diagrama de flujo en dfd es el siguiente:



El cumplimiento de la segunda condición es que si v es cero entonces no sea imposible resolver la ecuación por lo que la corrida se ve así:



Para la corrida en dfd omitiré los cuadros de dialogo que piden el valor de R y de v solo pondré sus valores:


Entrada de valores por teclado
Valor:
10
Continuar
Pausa
Entrada de valores por teclado
Valor:
10
Continuar
Pausa
Impresion por pantalla
Salida :
Introduce el valor de a
Continuar
Pausa
Entrada de valores por teclado
Valor:
5
Continuar
Pausa
Impresion por pantalla
Salida :
Introduce el valor de v
Continuar
Pausa
Entrada de valores por teclado
Valor:
0
Continuar
Pausa

La última prueba es que pasadas las dos anteriores resuelva de manera correcta la ecuación con los datos proporcionados y los muestre en forma de tabla:
Por lo que tomaremos como datos R y b iguales a 5, v =2, T= 0, a= 4
Por lo que sustituyendo quedaría:

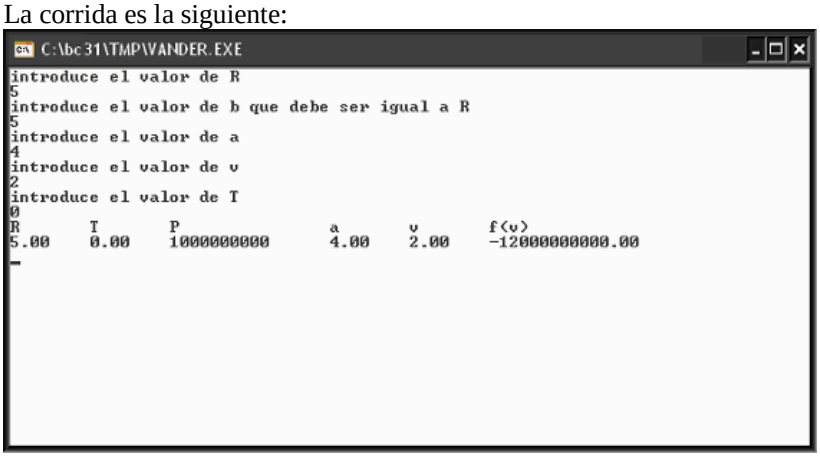
$$f(v) = ((1*10^9)(2)^3) - (((1*10^9)(5)) + ((5)(0))*(2)^2) + ((4*2) - (4*5))$$

$$f(v) = (8*10^9) - ((5*10^9 + 0)*4) + (8 - 20)$$

$$f(v) = 8*10^9 - 2*10^{10} - 12$$

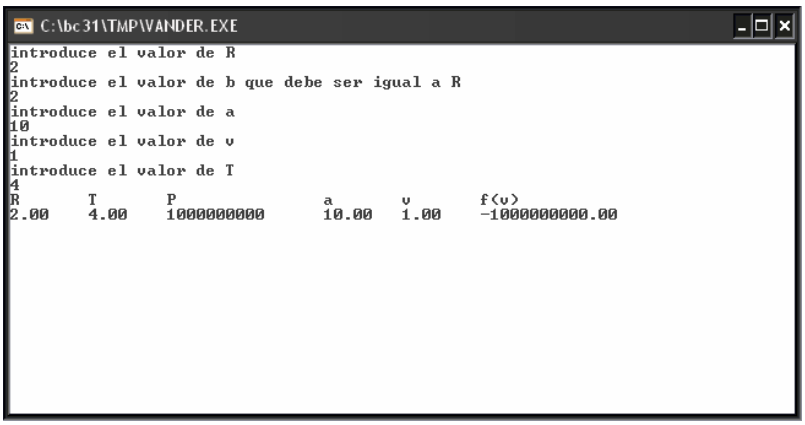
$$f(v) = -1.20*10^{10}$$

La corrida es la siguiente:

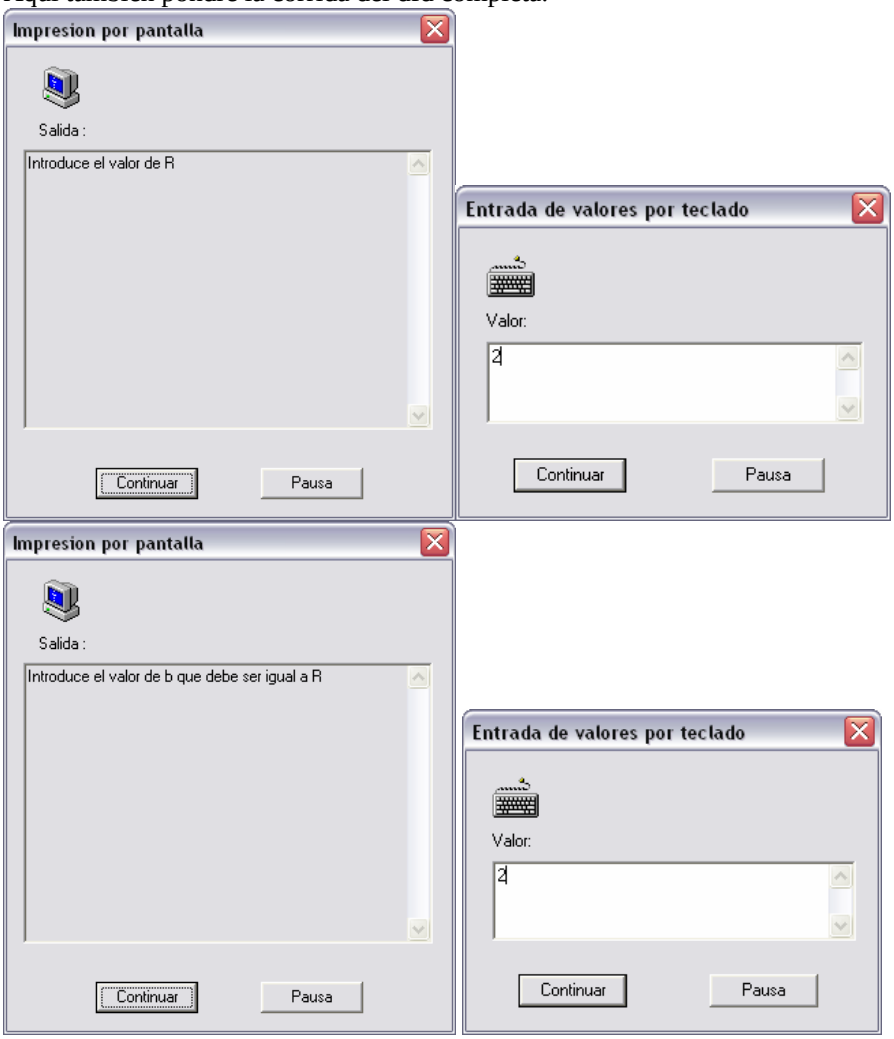


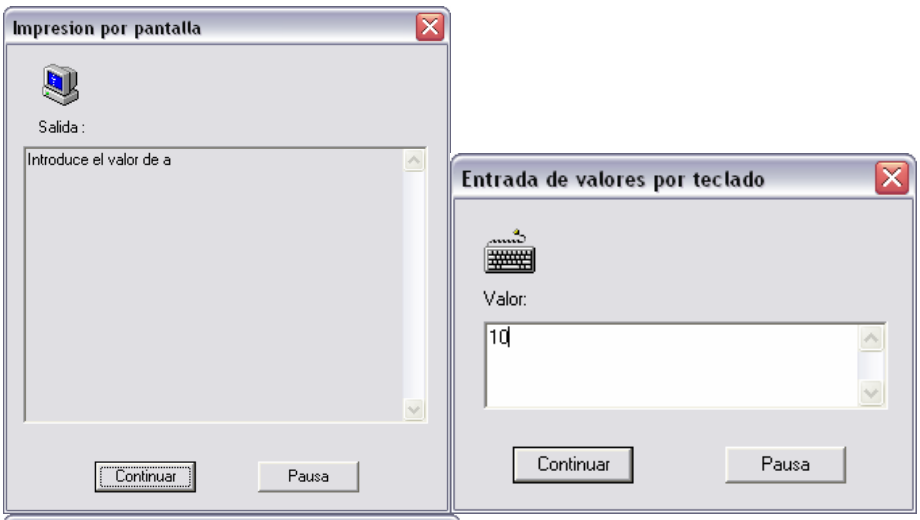
Una segunda prueba con valores de R y b iguales a 2, v=1, T=4, a=10

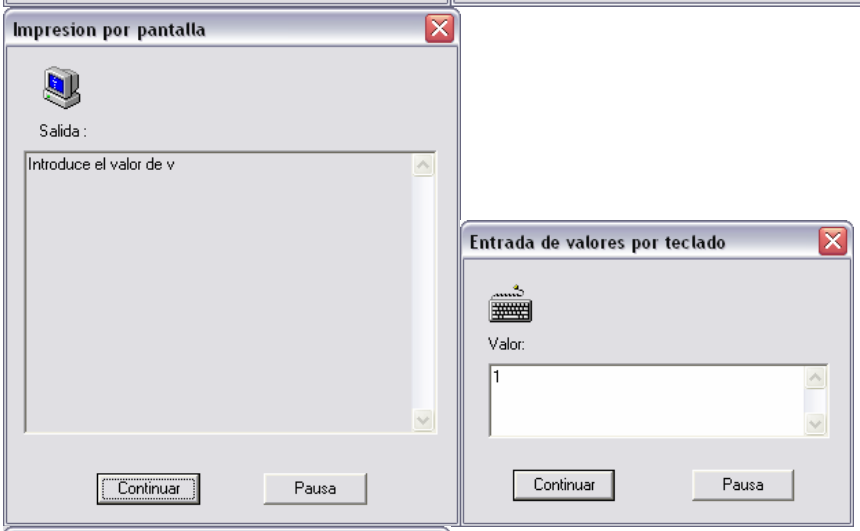
$$f(v) = ((1*10^9)(1)^3) - (((1*10^9)(2)) + ((2)(4))*(1)^2) + ((10*1) - (10*2))$$
$$f(v) = 1*10^9 - ((2*10^9) + 8)*1) - (10 - 20)$$
$$f(v) = 1*10^9 - 2000000008 - 12$$
$$f(v) = -1000000020 \sim -1*10^9$$

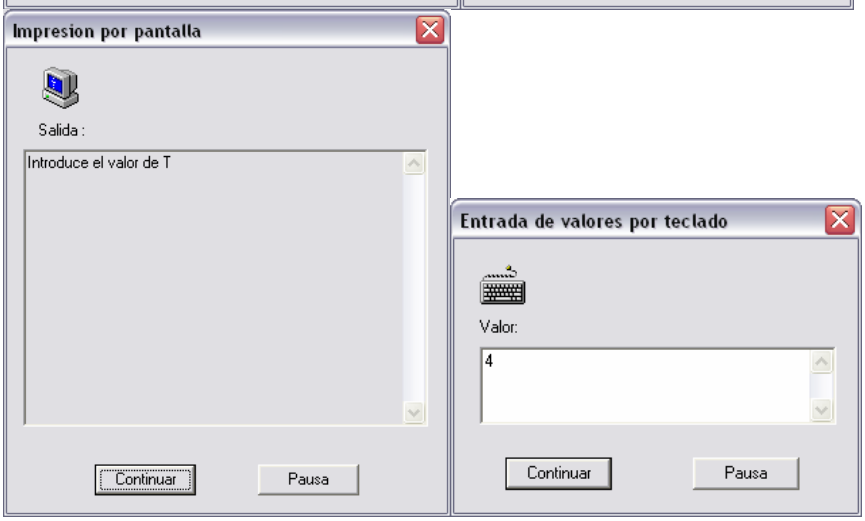


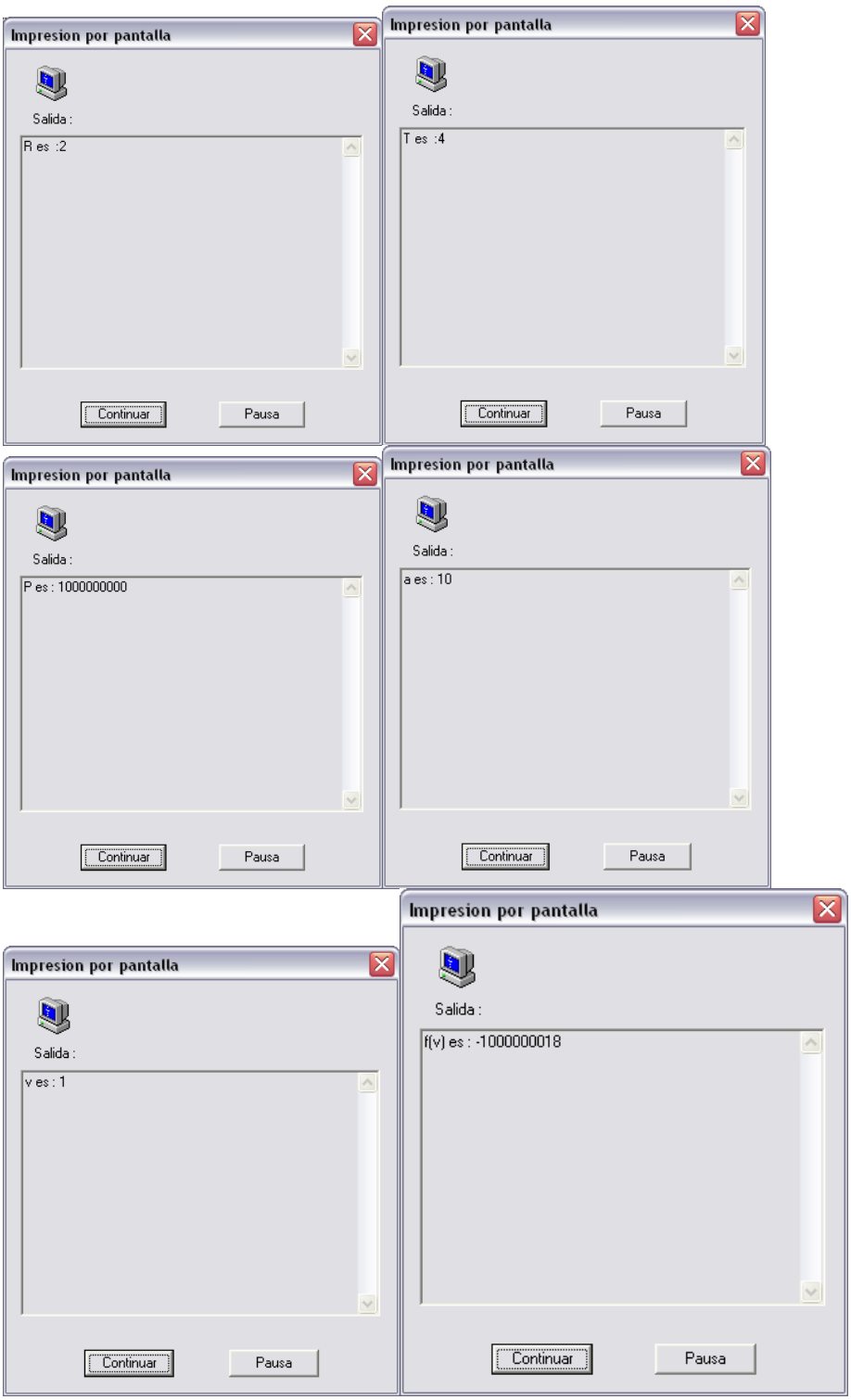
Aquí también pondré la corrida del dfd completa:




Impresion por pantalla
Salida :
Introduce el valor de a
Continuar
Pausa
Entrada de valores por teclado
Valor:
10
Continuar
Pausa


Impresion por pantalla
Salida :
Introduce el valor de v
Continuar
Pausa
Entrada de valores por teclado
Valor:
1
Continuar
Pausa


Impresion por pantalla
Salida :
Introduce el valor de T
Continuar
Pausa
Entrada de valores por teclado
Valor:
4
Continuar
Pausa



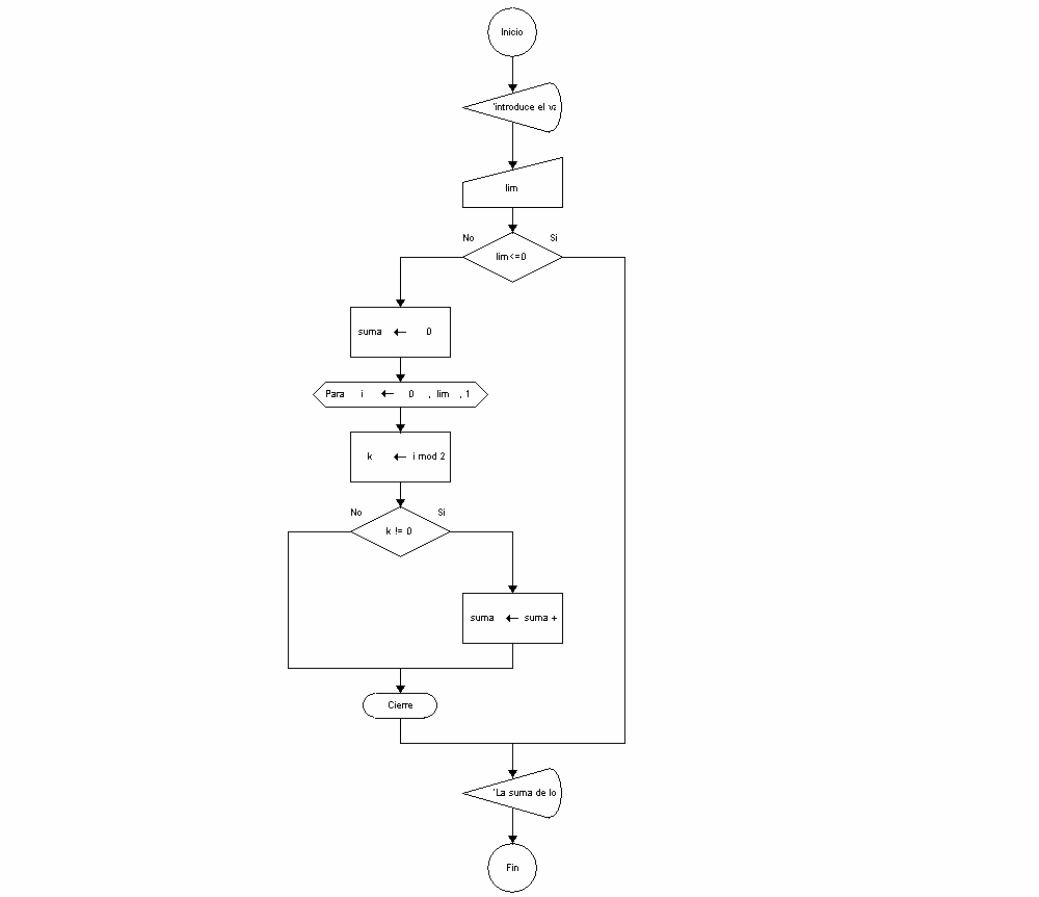
El tercer programa de esta practica fue el hacer que el usuario diera un numero como limite y a este numero se le sacaba la suma de todos los números impares a partir de el hacia el 0

Por lo que en forma *de metodología* analizamos que debíamos meter dos variables especiales en forma de contador la primera iria sumando de 1 en 1 hasta llegar al limite el segundo solo iria sumando los valores de el primer contador que después corroborado que eran impares para saber si son impares estos números introducimos en la condición que si el valor del primer contador

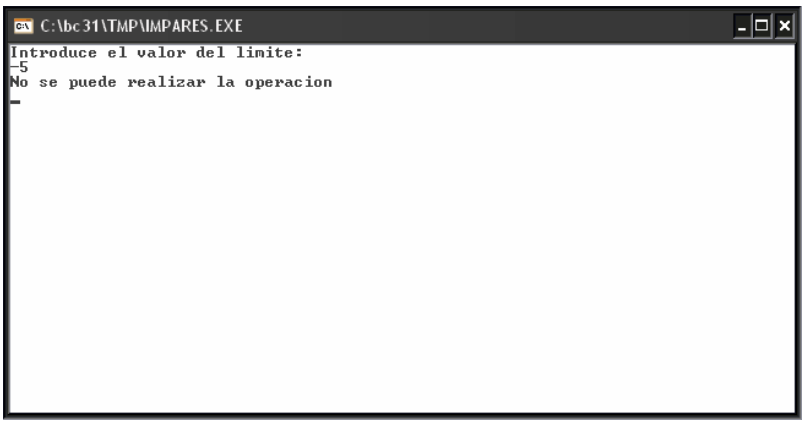
modulo de 2 es diferente de cero entonces hablamos de un numero impar o lo que es lo mismo si el valor del primer contador modulo de dos es igual a 1.
Por lo tanto el código fuente del programa quedo de la siguiente manera:

```
/*
FECHA: 12-SEP-06
PROGRAMA # 15
OBJETIVO: QUE EL ALUMNO APRENDA A USAR EL CICLO DE
REPETICIÓN WHILE Y A REALIZAR UN CONTADOR DENTRO DE
TURBO C PARA QUE  PUEDA AGRANDAR SUS POSIBILIDADES DE
REALIZACIÓN DE PROGRAMAS.*/
#include <stdio.h>
#include <math.h>
#include <conio.h>
void main ()
{
int i=0, limite, suma=0;
clrscr ();
printf ("Introduce el valor del limite:\n");
scanf ("%d", & limite);
if (limite<=0)
printf ("No se puede realizar la operacion\n");
else
{
while (i<=limite)
    {
    if (i%2 !=0)
    {
    suma +=i;
    }
    i++;
    }
printf ("La suma es:%d\n",suma);
}
getch ();
}
/*CONCLUSIONES: APRENDIMOS A USAR EL CICLO DE REPETICION WHILW QUE NOS PERMITE MEDIANTE UNA CONDICION REALIZAR UN CICLO N VECES MIENTRAS SE CUMPLA LA CONDICION FORMANDO ASI UN BUCLE QUE NOS SIRVE DE MUCHO YA QUE DENTRO DE ESTE BUCLE PODEMOS REALIZAR DIVERSAS OPERACIONES QUE SON REPETITIVAS AUNQUE LAS VARIABLES QUE CONTENGAN PUEDEN IR CAMBIANDO*/
```

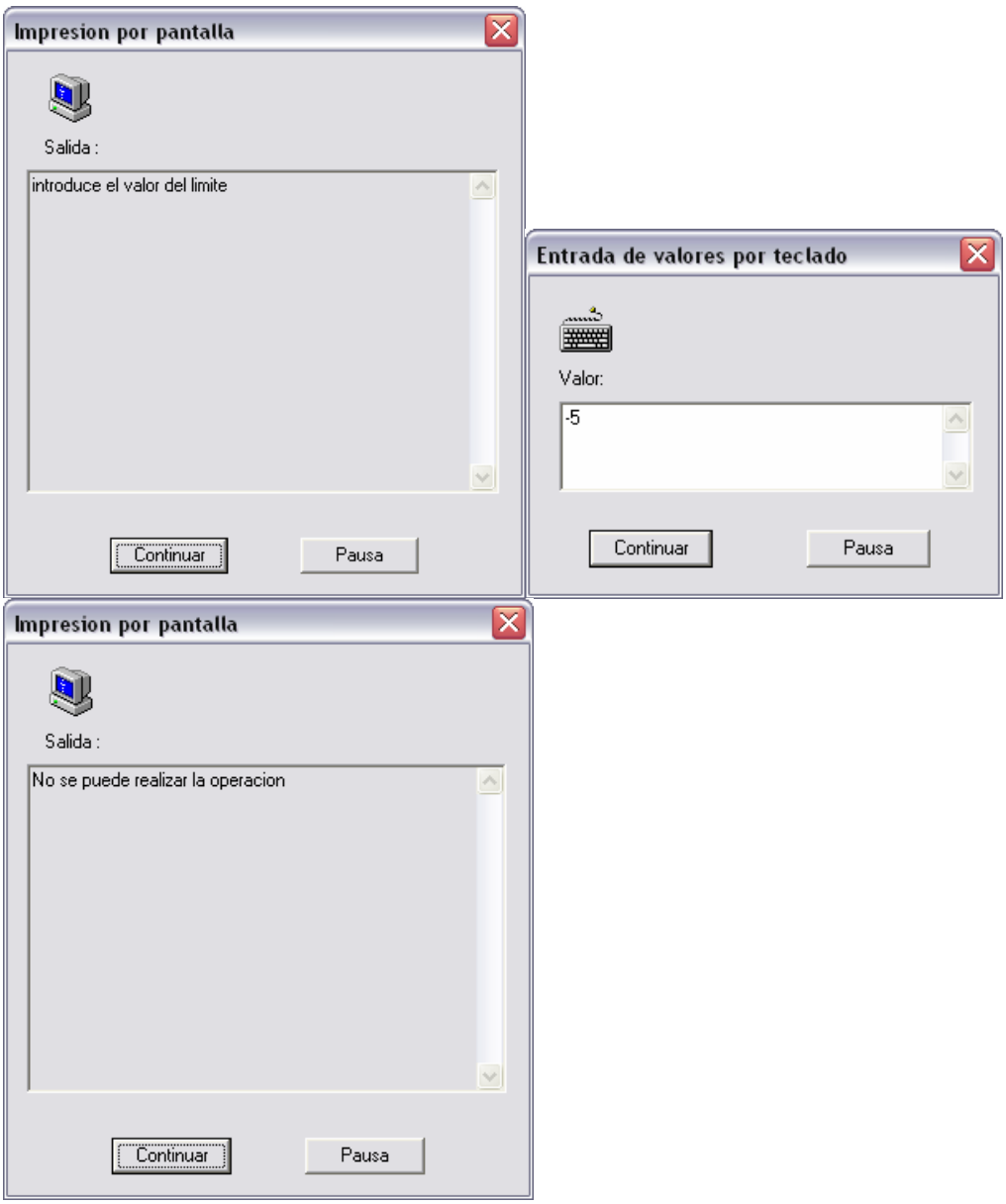
El diagrama de flujo se hizo a base del ‘ciclo para’ que no es mas que un contador de variable ‘i’ que empieza en 0 y llega hasta el limite dado por el usuario, dentro de este ciclo para introducimos la condición para saber si el numero es primo si lo es se le suma a la variable llamada ‘suma’ el valor del contador i si no se inicia el ciclo y el valor de i se incrementa en 1 y nuevamente se evalúa la condición para saber si el numero es primo. Cabe mencionar que la variable suma debe ser asignado a un valor de 0 antes de que inicie el ciclo lo anterior a este ciclo es la condición de que el limite no sea menor o igual a cero.



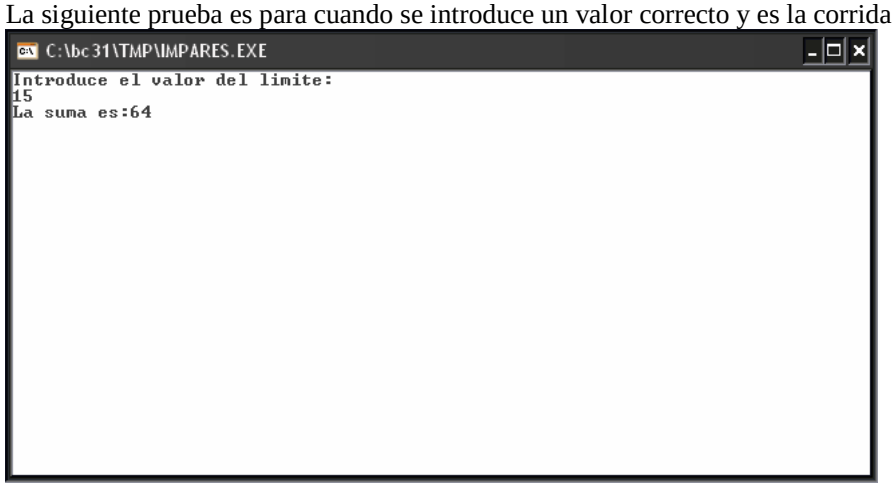
La corrida del programa siendo que el nuecero dado sea menor o igual a 0 debe ser la siguiente:



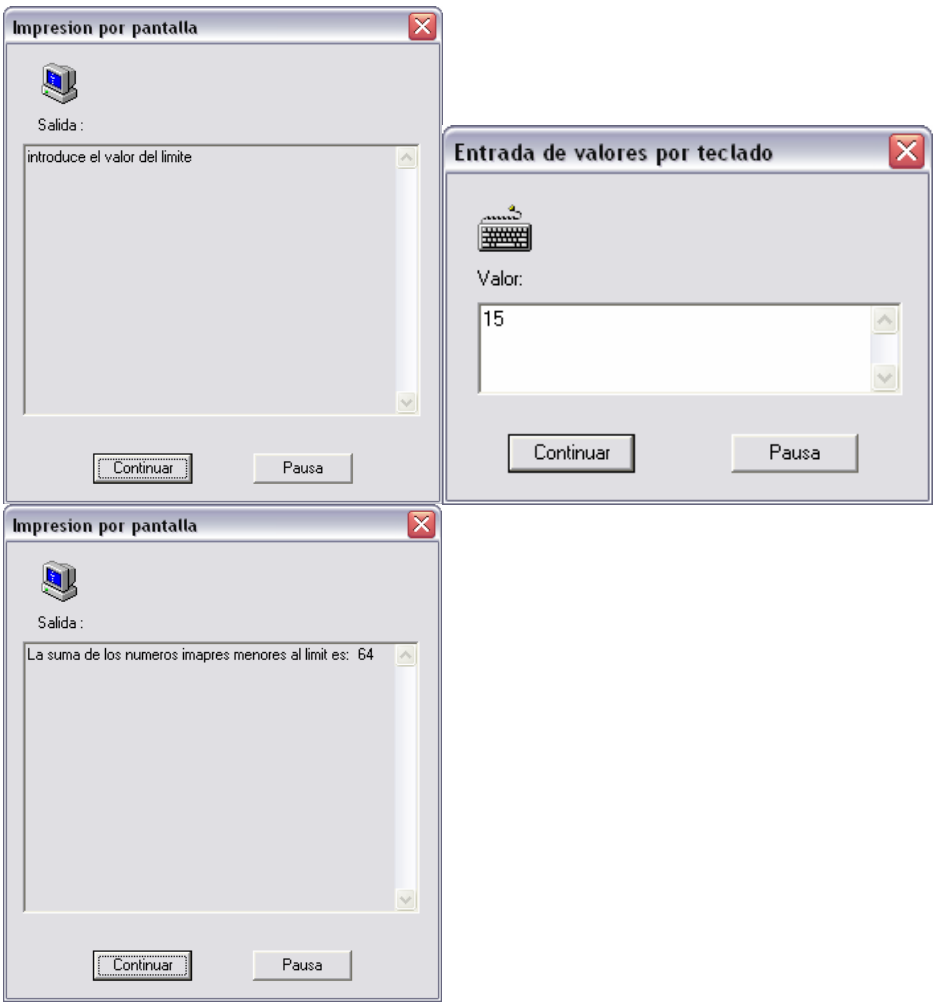
La corrida en dfd es la siguiente:



La siguiente prueba es para cuando se introduce un valor correcto y es la corrida se ve asi:



la corrida en dfd se ve asi:



Y que da comprobado ya que: 1+3+5+7+9+11+13+15 = 64

El ultimo programa es el de encontrar el factorial de un numero que es introducido por el usuario. Por lo que a manera de *metodología* tenemos que el factorial del número no debe rebasar los 10000 y si lo hace se debe pedir al usuario que introduzca nuevamente el numero para obtener su factorial.

Por lo tanto tenemos que hacer dos ciclos de repetición el primero debe tener como condición que si la variable factor que es el resultado es mayor a 10000 se ejecute o que también se ejecute cuando la variable factor sea 1 y la variable i que será nuestro contador sea también 1 para que pueda entrar al ciclo la primera vez que se ejecuta el programa; el segundo ciclo de repetición mientras debe tener como condición que el numero que introdujo el usuario no sea rebasado y dentro de este ciclo multiplicaremos la variable factor por el valor de i que en un principio será 1 pero ira aumentando de uno en uno hasta alcanzar el valor del nuecero por lo que se saldrá de este ciclo y si el numero es menor a 10000 (condición del primer ciclo mientras) se saldrá de ambos ciclos e imprimirá el valor de factor, de lo contrario entrar de nuevo en los dos ciclos mientras. Para que los valores de i y factor vuelvan a 1 antes de entrar al segundo ciclo mientras debemos dividirlos entre su mismo valor para que de el numero 1.
Entonces el programa quedo de la siguiente forma:

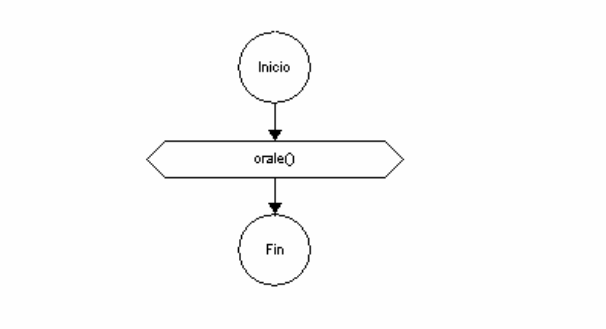
/*
FECHA: 12-SEP-06

PROGRAMA # 16
OBJETIVO: QUE EL ALUMNO APRENDA A USAR EL CICLO
DE REPETICION WHILE Y DESARROLLAR UN PROGRAMA
COMPLEJO QUE REQUIERA DE RAZONAMIENTO Y EL PODER
INTRODUCIR UN CICLO QUE REGRESE AL PRINCIPIO SI
NO SE DA UNA DE LAS CONDICIONES .*/

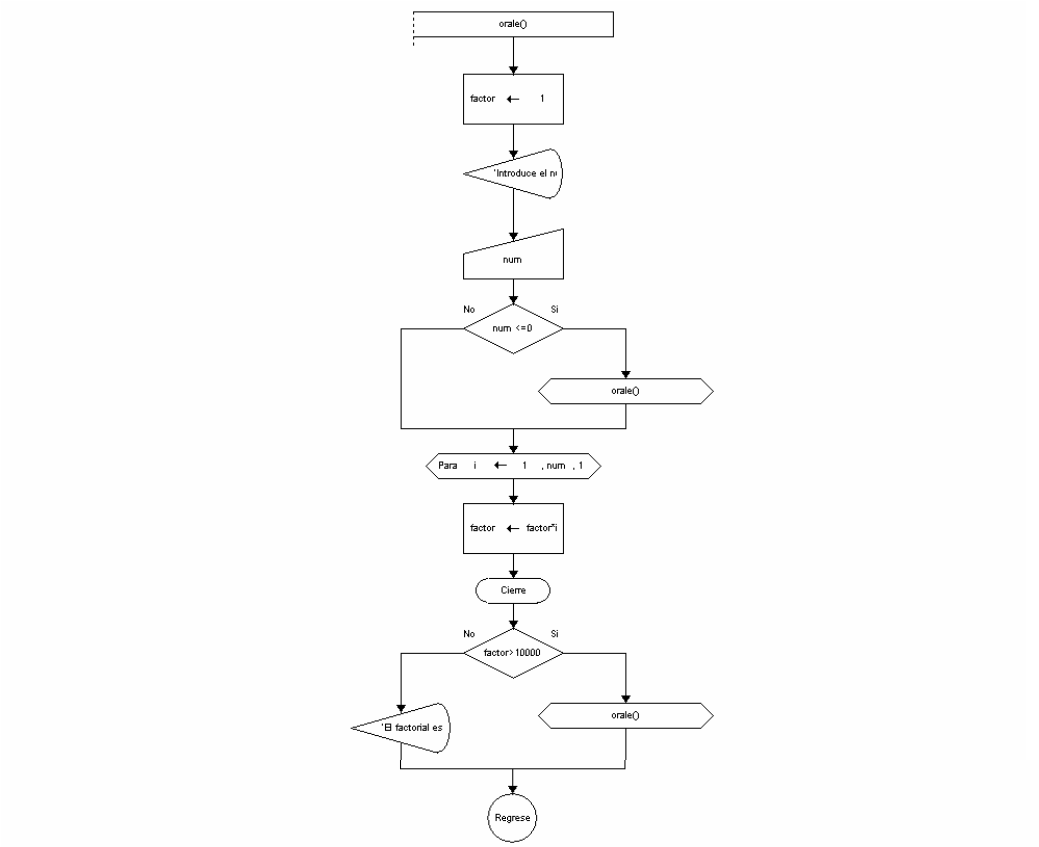
```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main ()
{
float  factor=1.0,num;
int i=1;
clrscr ();
while (factor > 10000 || (factor==1 && i==1))
    {
    printf ("Introduce el numero al que se quiere obtener el factorial\n");
    scanf ("%f",& num);
    i=i/i;
    factor=factor/factor;
    while (i<=num)
            {
            factor*=i;
            i++;
            }
    }
    printf ("El factorial es:%.2f", factor);
getch ();
}
```

/* CONCLUSIONES: EN ESTE PROGRAMA PUDE NOTAR QUE SE
PUEDEN INTRODUCIR PARA EL CICLO DE REPETICION WHILE
VARIAS CONDICIONES PARA QUE ENTRE AL CICLO ADEMAS DE
QUE TAMBIEN SE VIO CLARAMENTE QUE SE PUEDEN REALIZAR CICLOS DE REPETICION ANIDADOS*/

El diagrama de flujo esta hecho a base de un subprograma y un ciclo para y funciona de la siguiente forma:
Al inicio se le da una llamada al subprograma orale por lo que entra en este subprogramaza dentro de el lo que hace es dar el valor de 1 a la variable factor ya que cada que entre a este subprograma pierde los datos por lo que si necesitamos entrar de nuevo debemos poner de nuevo las variables a 1 luego se pida se introduzca el valor del numero al que queremos obtener el factorial este se relaciona a la variable num, luego se entra a un ciclo de repetición para que tiene como variable a i que inicia en 1 y termina en el valor de num avanzando de uno en uno, luego ponemos que la variable factor es igual a factor multiplicado por i por lo que cada vez que se entre al ciclo de repetición para se multiplicara por i que ira aumentando en 1 su valor hasta que se llegue al valor de num y se sale del ciclo, luego se evalua si el valor de factor es mas grande que 10000 de ser cierto se inicia de nuevo el subprograma orale pero si es falso se imprime el valor de factor y se sale del subprograma y se regresa al programa principal donde termina la operación.
Por lo que el diagrama de flujo esta de la siguiente forma:



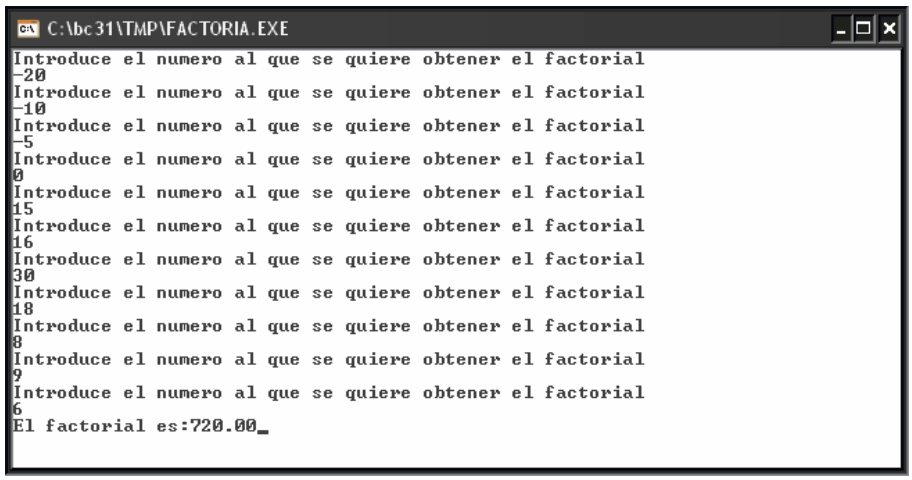
Este es el subprograma ‘orale’



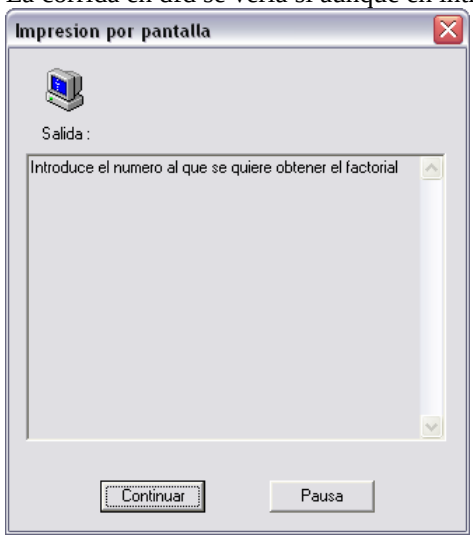
La primera corrida corresponde a que se de un ciclo de repetición por que los números que se han introducido son incorrectos o el factorial del numero rebasa los 10000 siendo que todo numero introducido arriba de 7 su factorial será mayor a 10000 por que tan solo el factorial de 8 ya alcanza el valor de 40320 no con ello quiero decir que la condición de mi programa sea que el numero que introduce el usuario sea menor a 8.

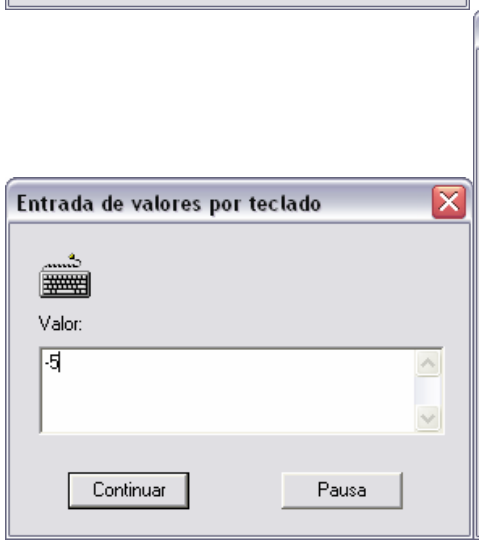
Nótese que estamos introduciendo primero valores negativos luego pasmaos por el cero y luego por números mayores a 8 que su factorial por inferencia decimos que son mayores 10000 pero nuestro programa no nota que estos números son mayores a 7, si no que el factorial es mayor a 10000 como lo condicionamos y por lo tanto siempre pide un nuevo valor para obtener el factorial hasta que se pone un numero el cual su factorial esta entre 1 y 10000.

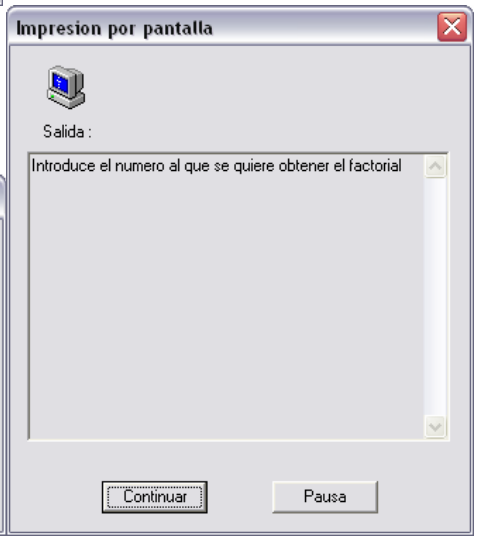
Por lo que la corrida es la siguiente:

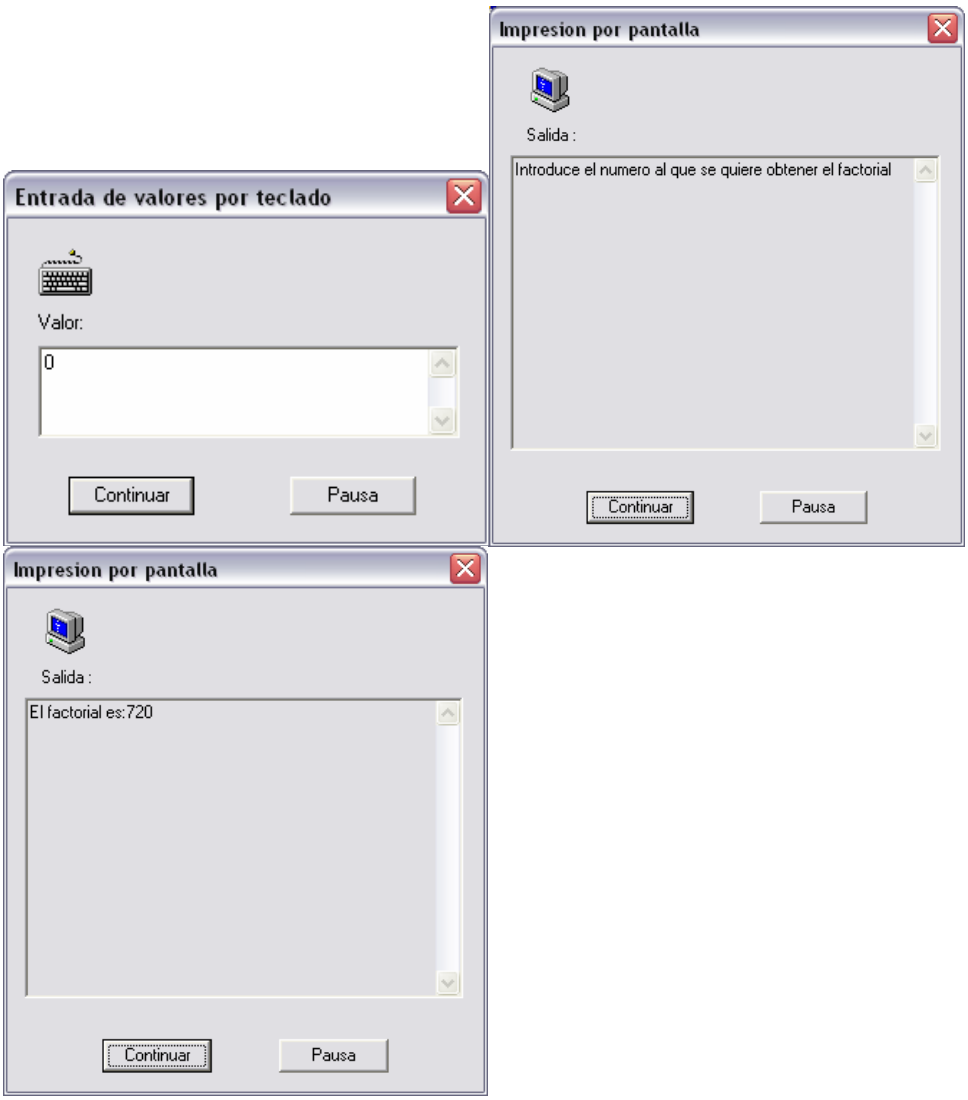


La corrida en dfd se vería si aunque eh introducido menos valores.

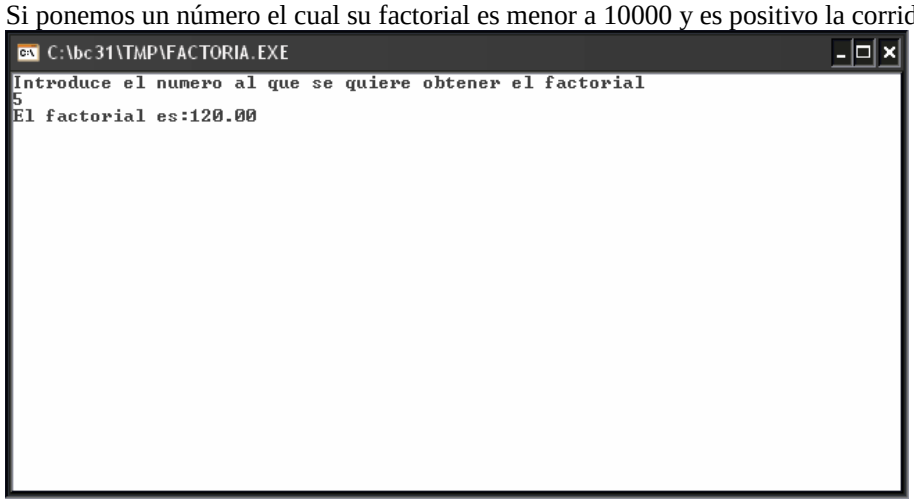




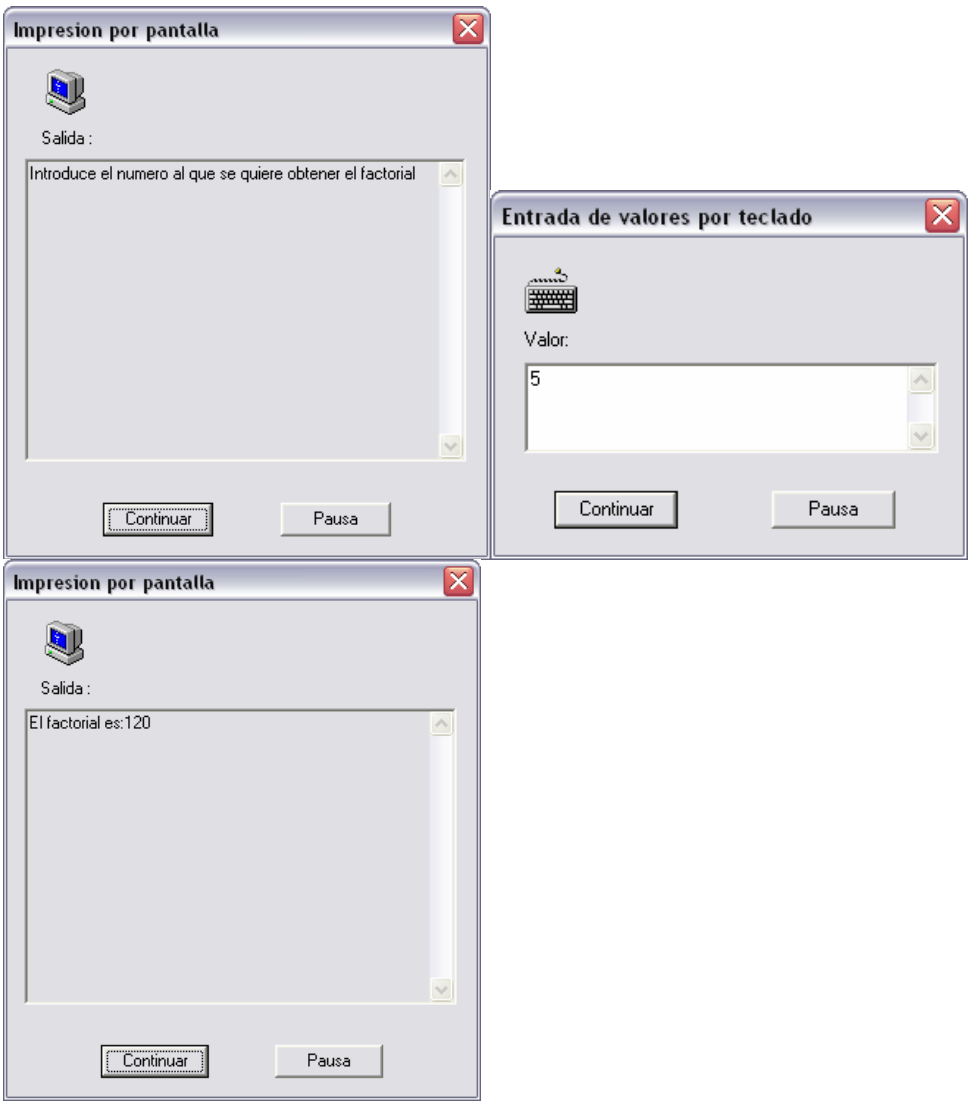




Si ponemos un número el cual su factorial es menor a 10000 y es positivo la corrida se vera así:



La corrida en dfd se ve de la siguiente forma:



Conclusiones generales:

En esta practica aprendimos a usar las estructuras de decisión en cascada y anidados además de aprender a realizar una tabla, quitarle decimales a un resultado de punto flotante además de ver la utilidad tan grande que se puede dar al restringir y delimitar algunas operaciones además de una mejor comunicación con el usuario, también observamos ahora si la utilidad del operando modulo ya que nos da como resultado el residuo de una división que en este caso pudimos aplicar. En los últimos dos programas aprendimos a usar el ciclo de repetición while y observamos que incluso podemos poner varias condiciones para que se realice el ciclo de repetición y anidar este tipo de ciclos de repetición.