Proyecto final

FECHA: 20/05/2022

ALUMNOS:
EDUARDO DE JESÚS QUINTERO MEZA. CÓDIGO: 218551683
LEVI ALEXANDER PEREZ ELIZONDO. CÓDIGO: 218701391
CÓDIGO:

MATERIA: PROGRAMACIÓN ESTRUCTURADA. CLAVE: IL352

HORARIO: MIÉRCOLES Y VIERNES, DE 13:00 A 15:00 HRS.

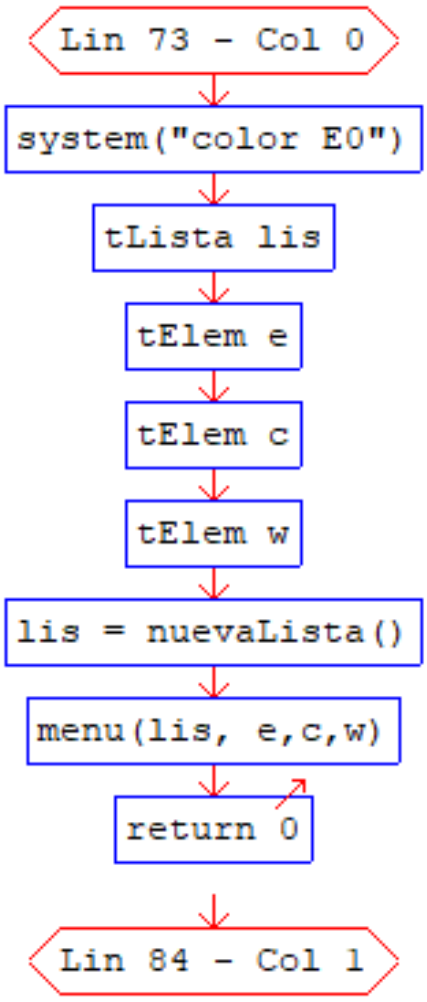
Planteamiento:

Planteamiento del problema: Con este proyecto se pretende solucionar la problemática al momento de revisar tu horario y ver algo confuso y difícil de entender por medio de una aplicación en la que lo pueda organizar sin pedir sus datos institucionales para evitar desconfianza ya que muchas aplicaciones que se encuentran disponibles actualmente piden datos de su cuenta institucional y no son aplicaciones oficiales.

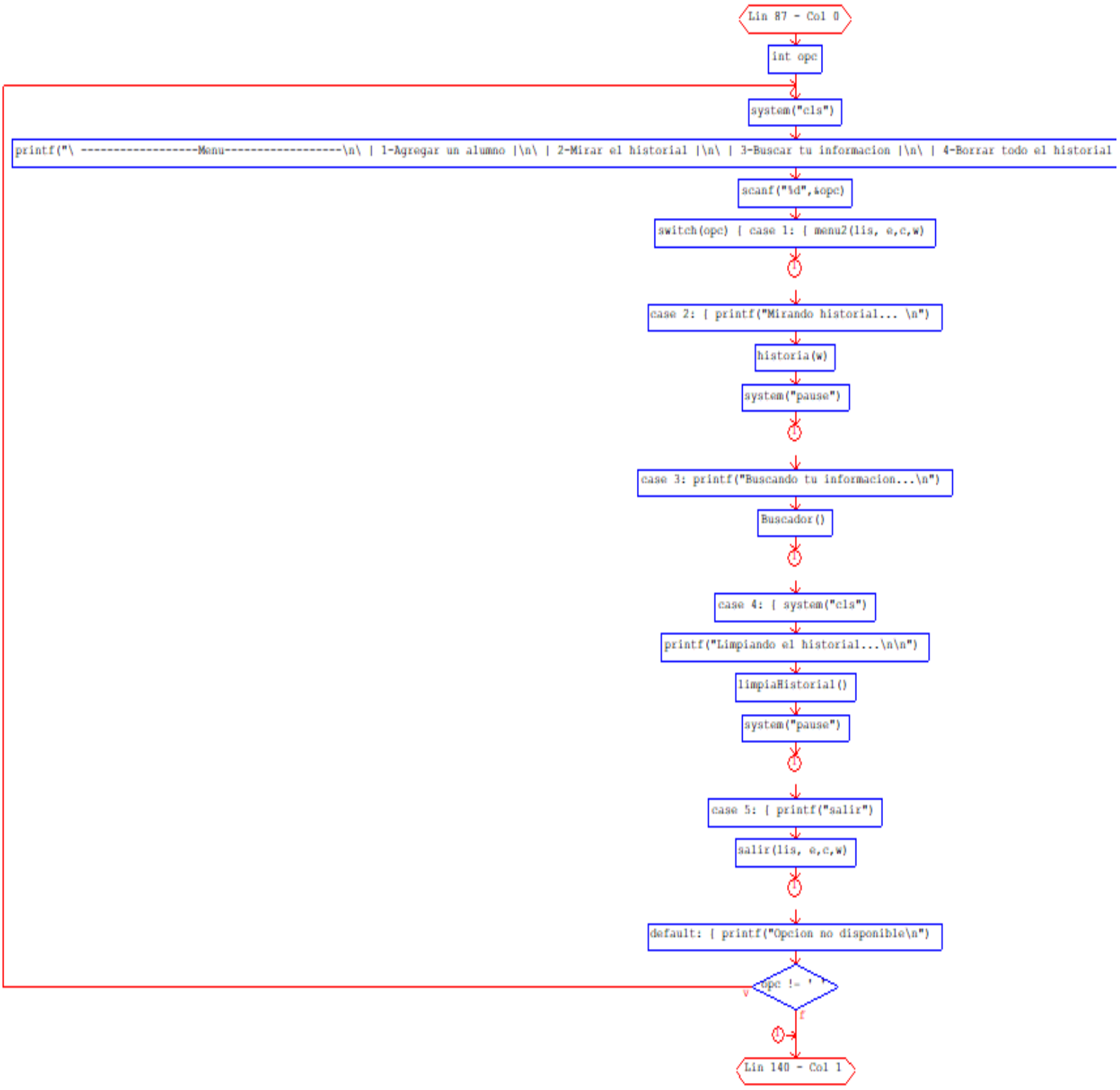
El programa va a consistir en el diseño de un menú en el cual el usuario puede seleccionar una opción las cuales van referentes al registro, búsqueda, eliminación, vista y ordenamiento de las cosas que ingrese, los primeros datos que ingrese son Nombre, materia, día, código, nombre, y edad. Posteriormente el programa va a solicitar en orden el nombre de la materia, el día de la semana, la hora de inicio y la hora de salida, siguiendo este proceso hasta terminar acomodando todo en una tabla.
Finalmente cuando se acabe el registro y se seleccione ver los datos el usuario pueda encontrar de forma ordenada por medio de una tabla su horario registrado.

Sección 1: Diagrama de flujo

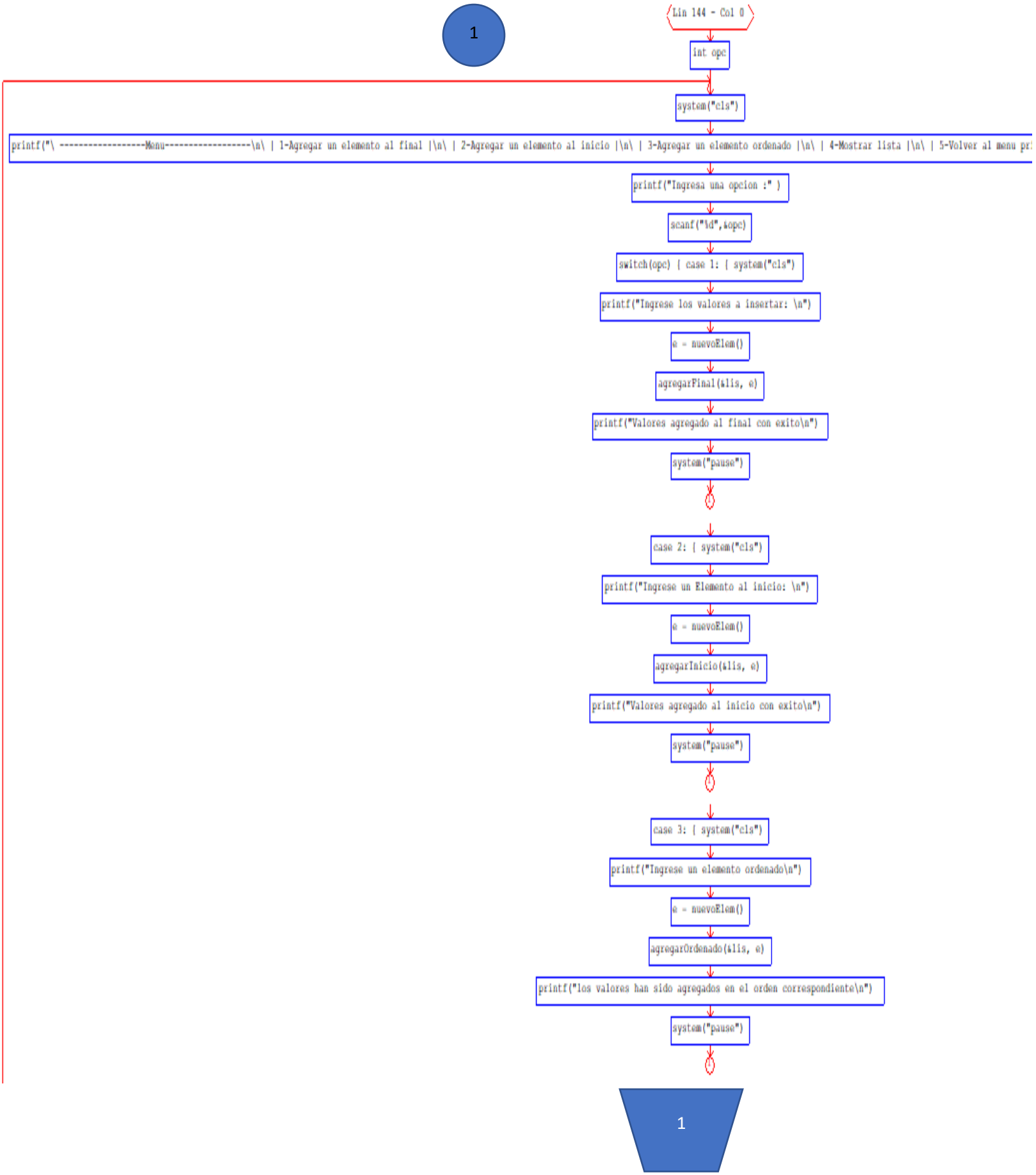
Función Main



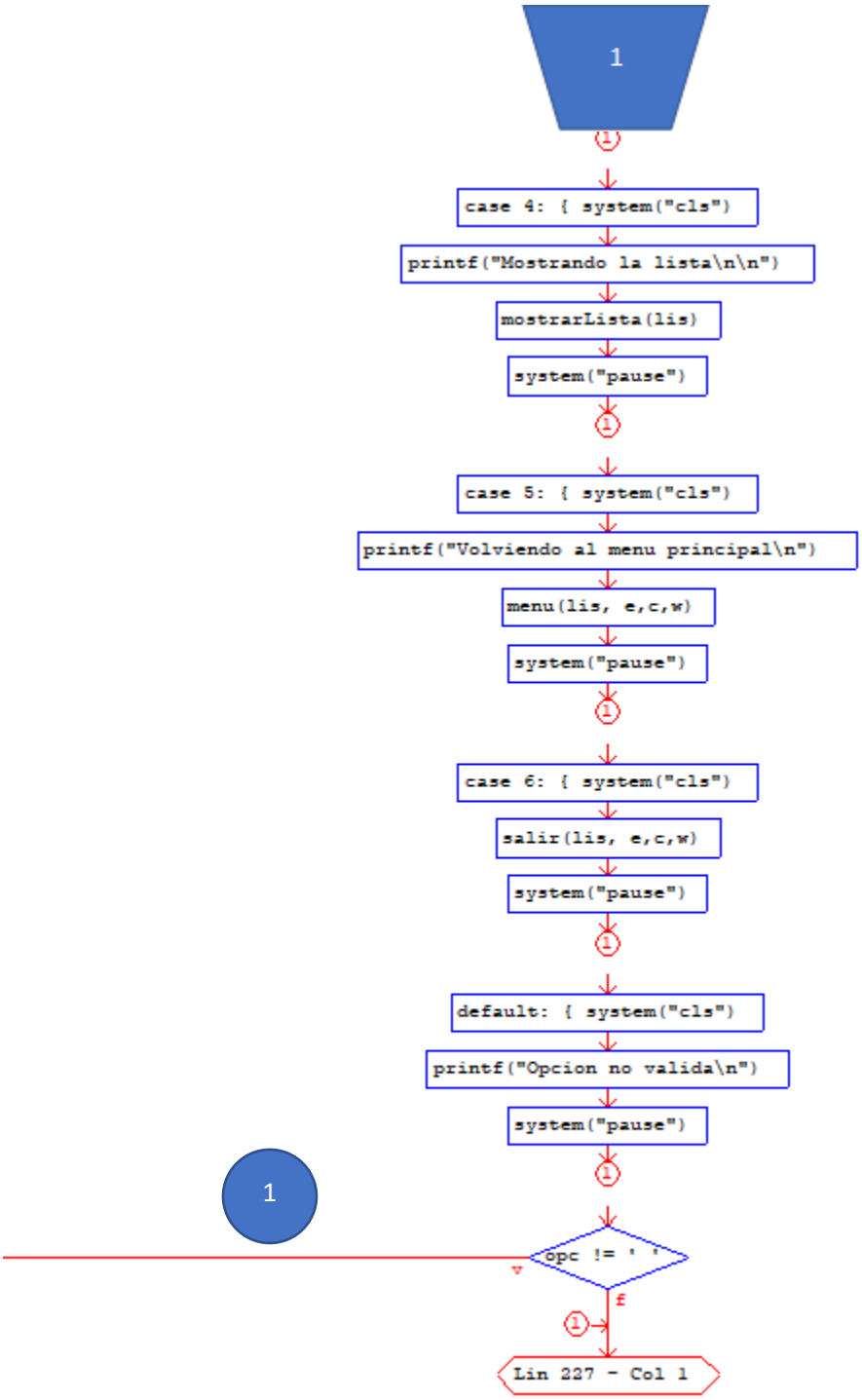
Menu principal



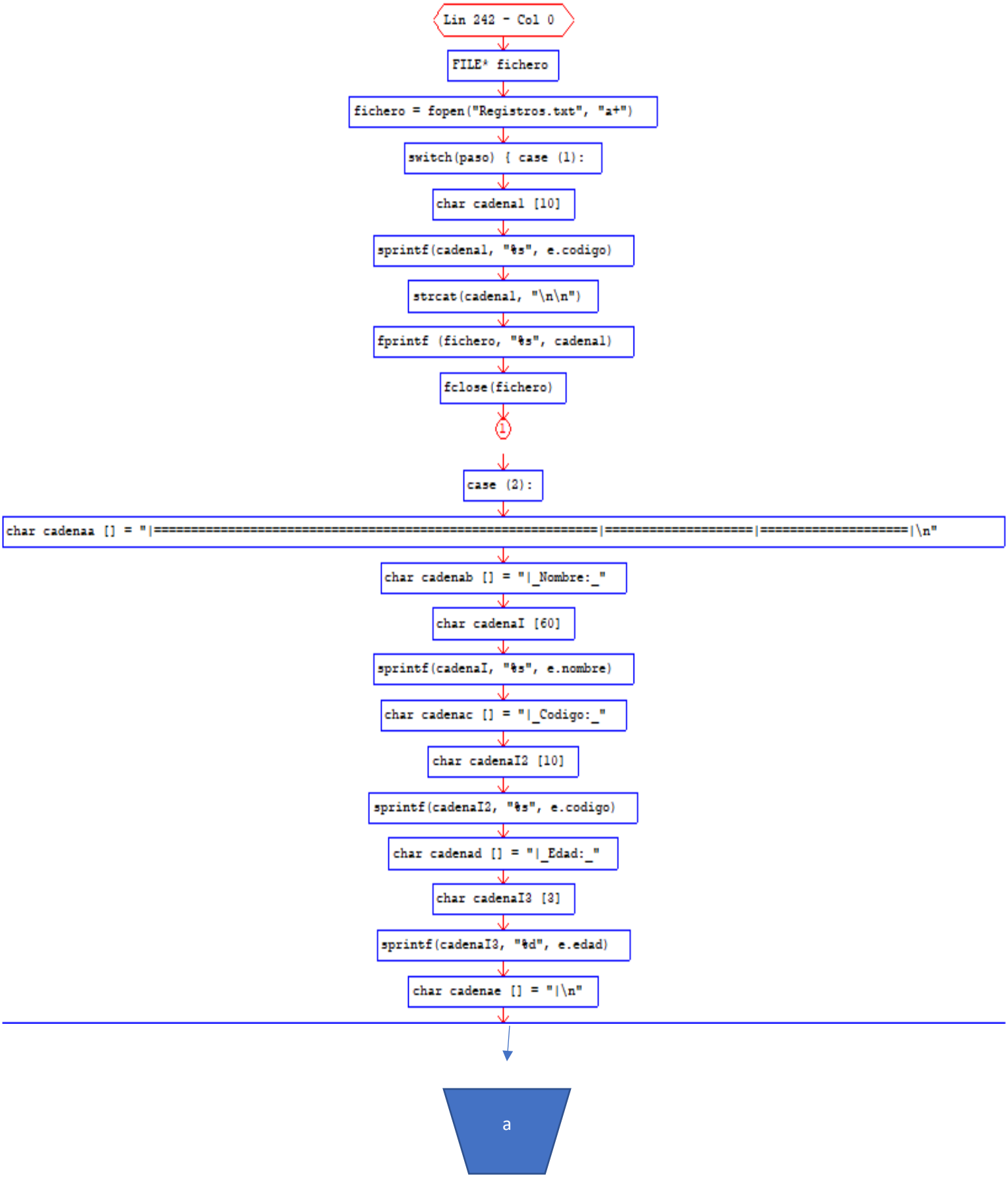
Menu 2


1
Lin 144 - Col 0
int opc
system("cls")
printf("\ ----------------Menu----------------\n\ | 1-Agregar un elemento al final |\n\ | 2-Agregar un elemento al inicio |\n\ | 3-Agregar un elemento ordenado |\n\ | 4-Mostrar lista |\n\ | 5-Volver al menu pr
printf("Ingresa una opcion :")
scanf("%d",&opc)
switch(opc) { case 1: { system("cls")
printf("Ingrese los valores a insertar: \n")
e = nuevoElem()
agregarFinal(&lis, e)
printf("Valores agregado al final con exito\n")
system("pause")
case 2: { system("cls")
printf("Ingrese un Elemento al inicio: \n")
e = nuevoElem()
agregarInicio(&lis, e)
printf("Valores agregado al inicio con exito\n")
system("pause")
case 3: { system("cls")
printf("Ingrese un elemento ordenado\n")
e = nuevoElem()
agregarOrdenado(&lis, e)
printf("los valores han sido agregados en el orden correspondiente\n")
system("pause")
1

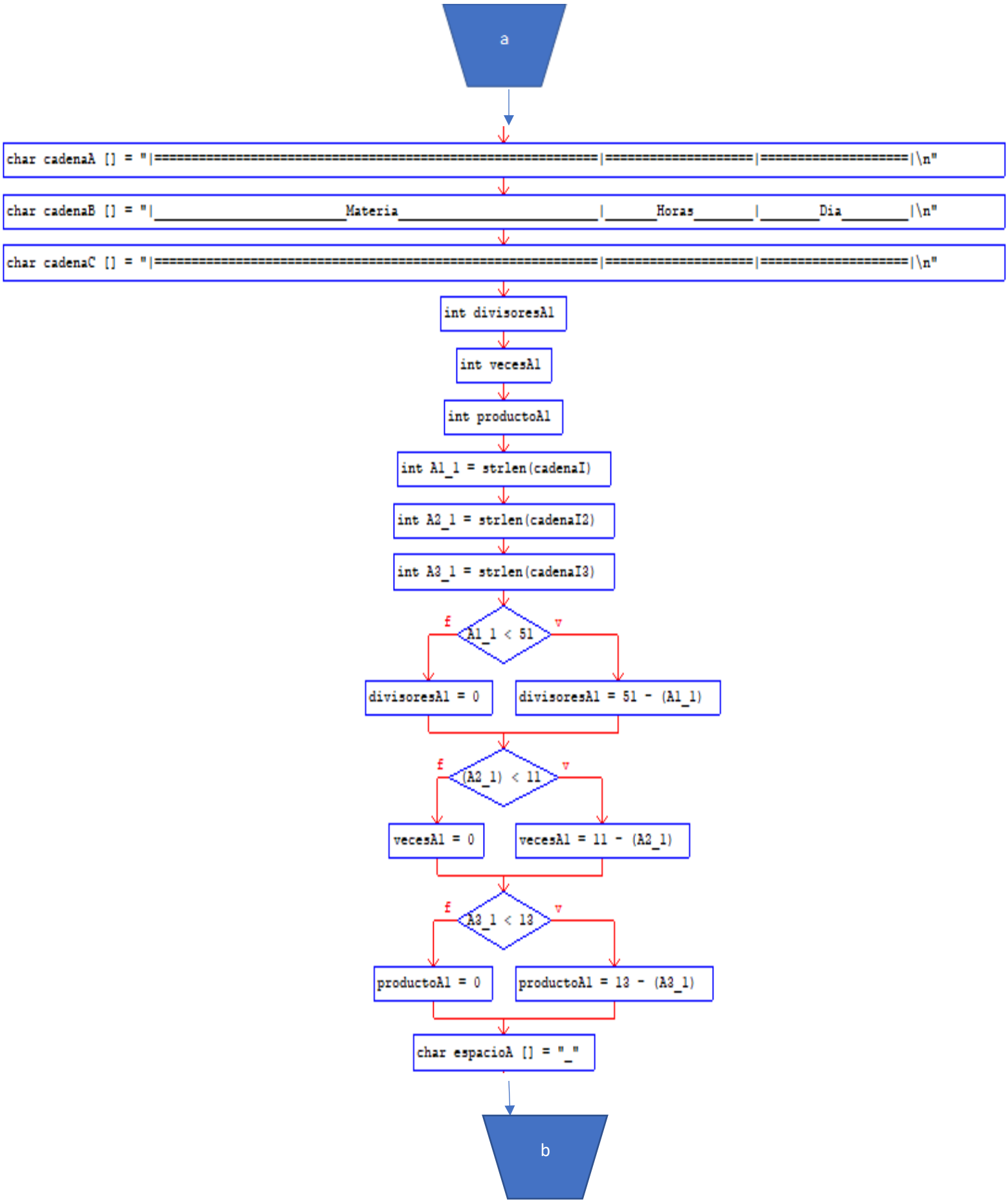
5

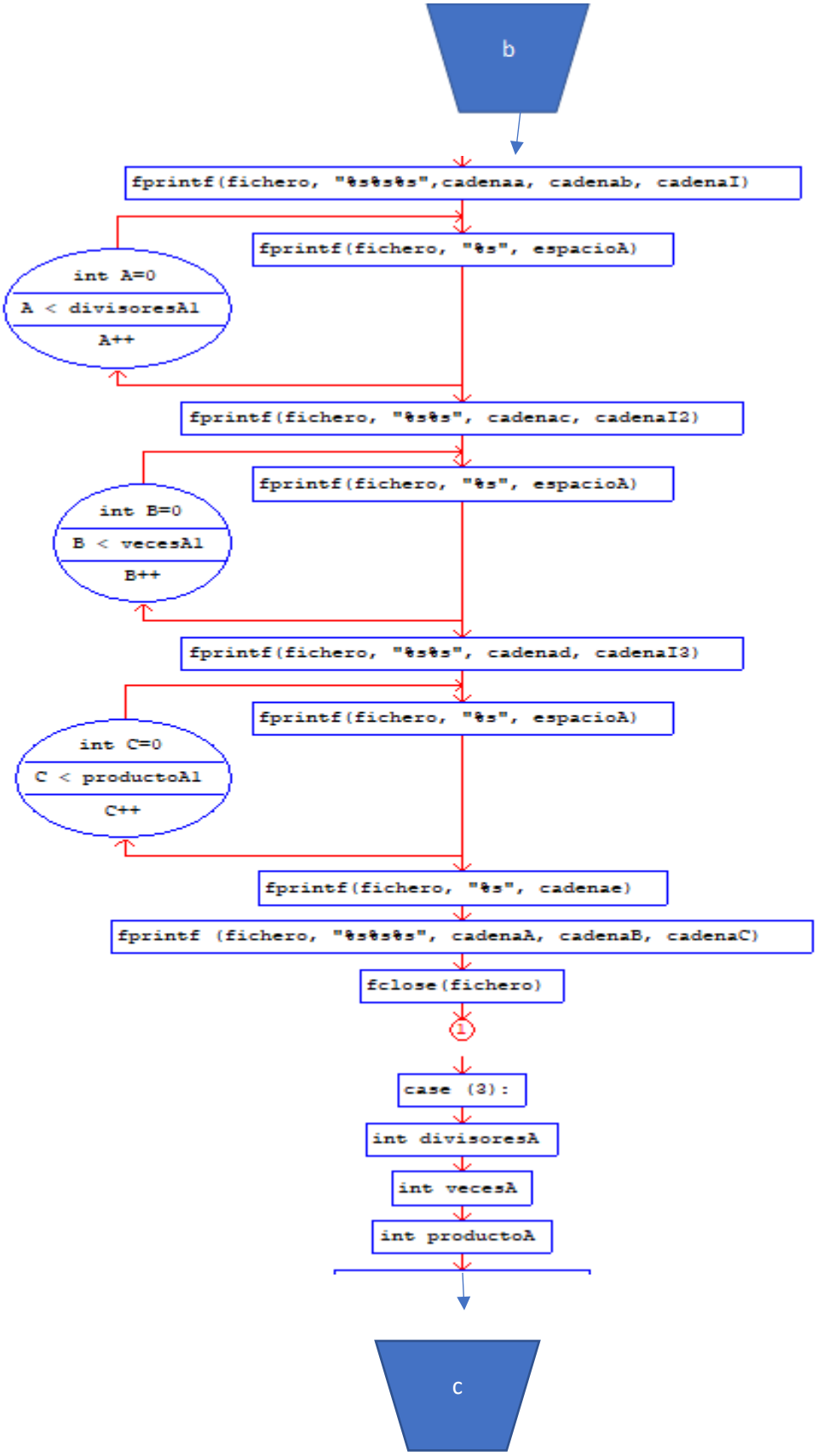

1
case 4: { system("cls")
printf("Mostrando la lista\n\n")
mostrarLista(lis)
system("pause")
case 5: { system("cls")
printf("Volviendo al menu principal\n")
menu(lis, e,c,w)
system("pause")
case 6: { system("cls")
salir(lis, e,c,w)
system("pause")
default: { system("cls")
printf("Opcion no valida\n")
system("pause")
1
opc != ' '
Lin 227 - Col 1

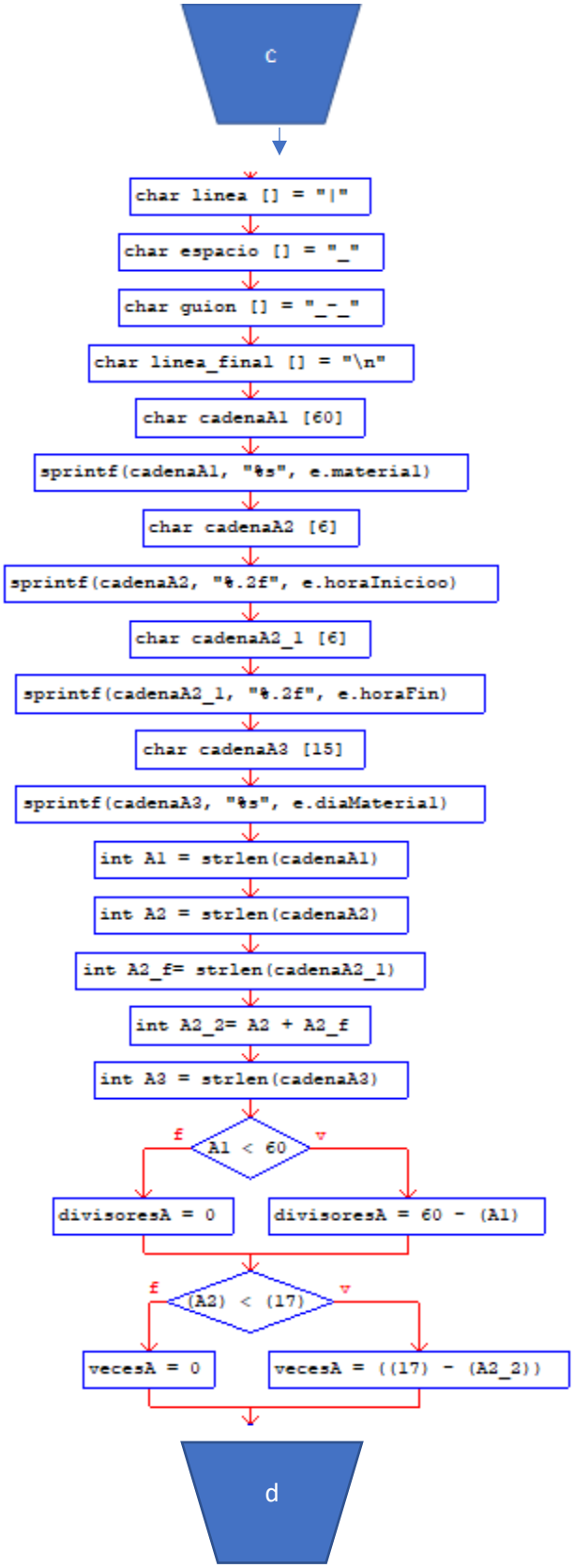
Guardar funcion

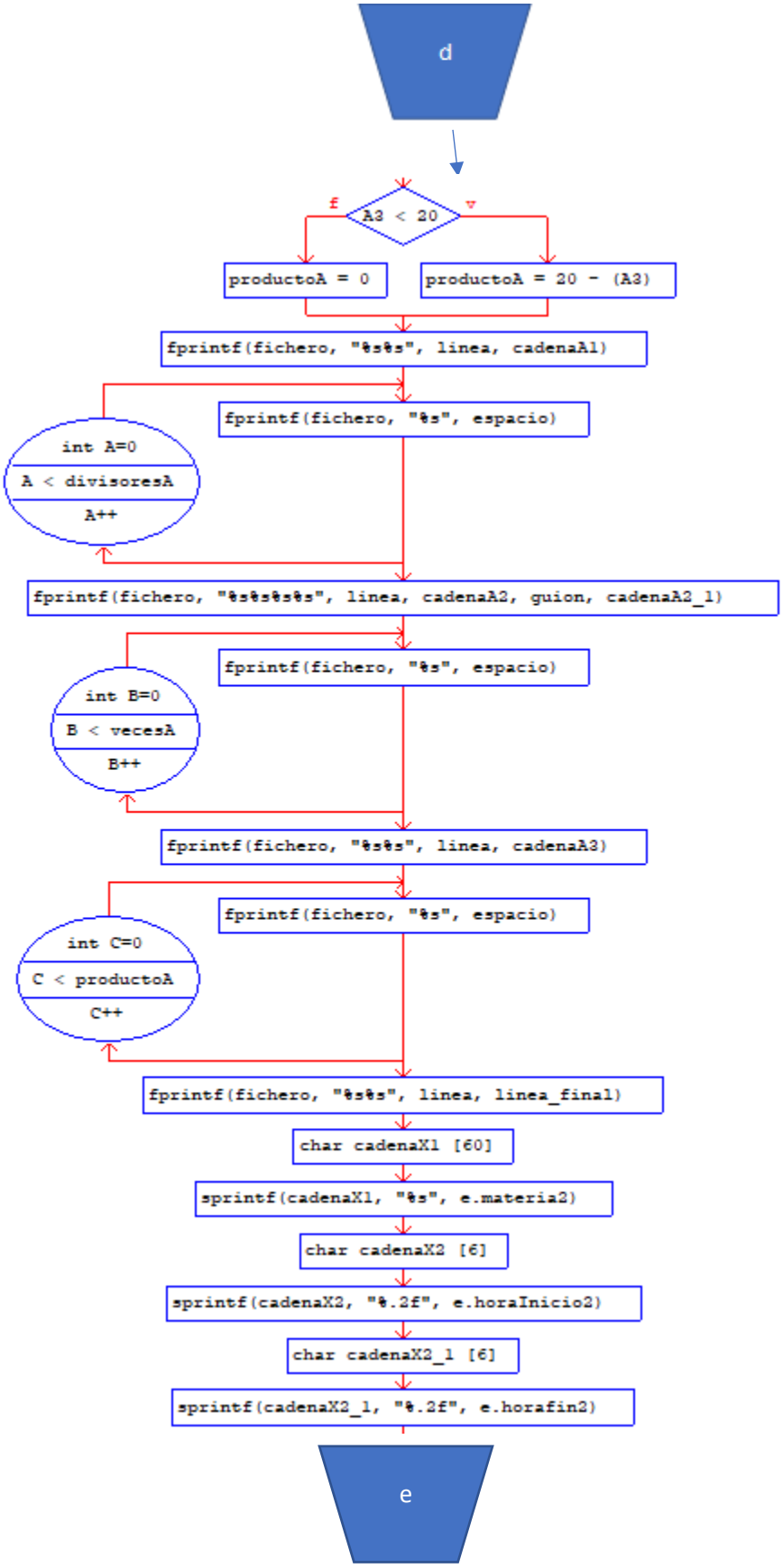


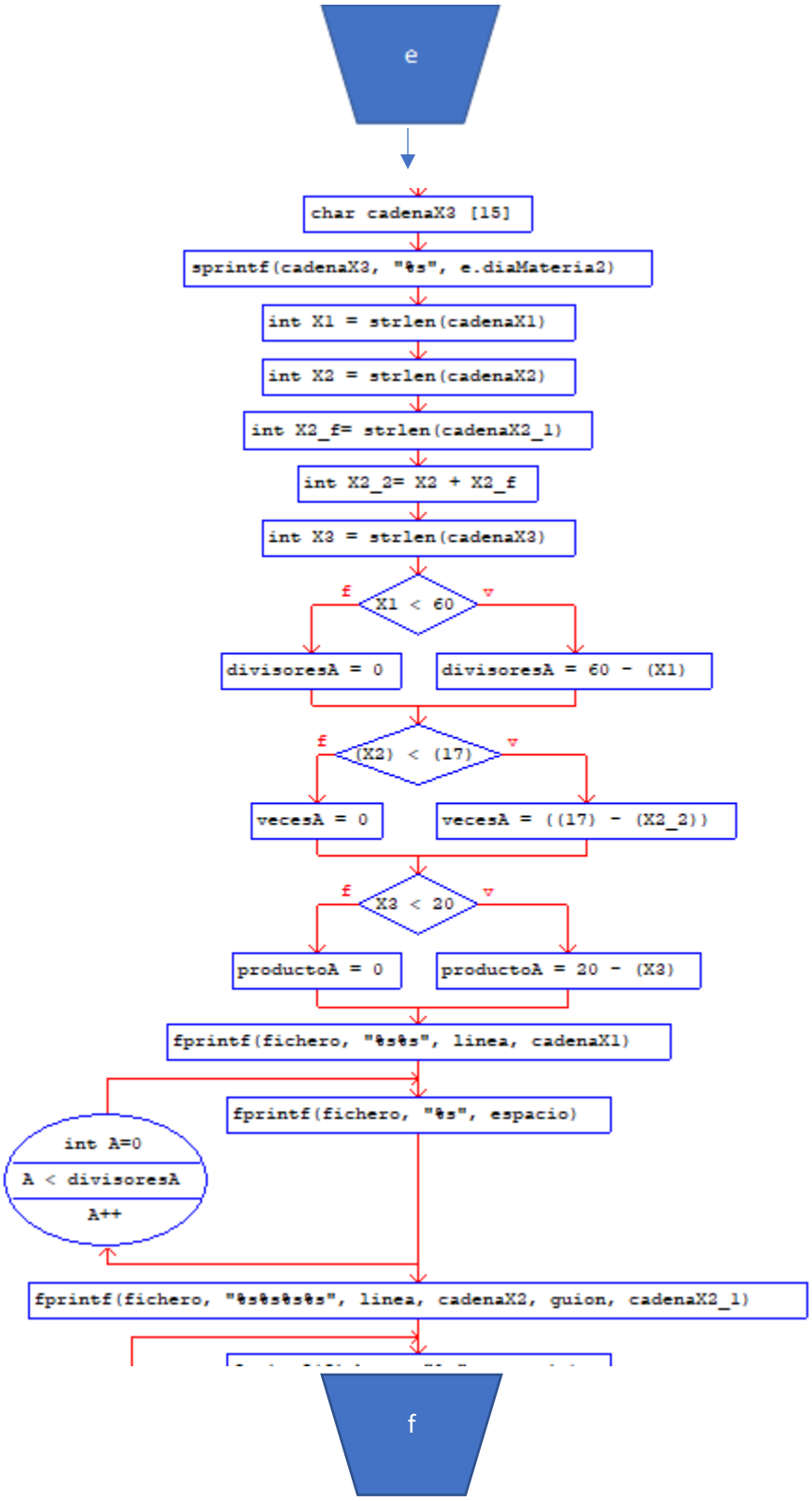
7

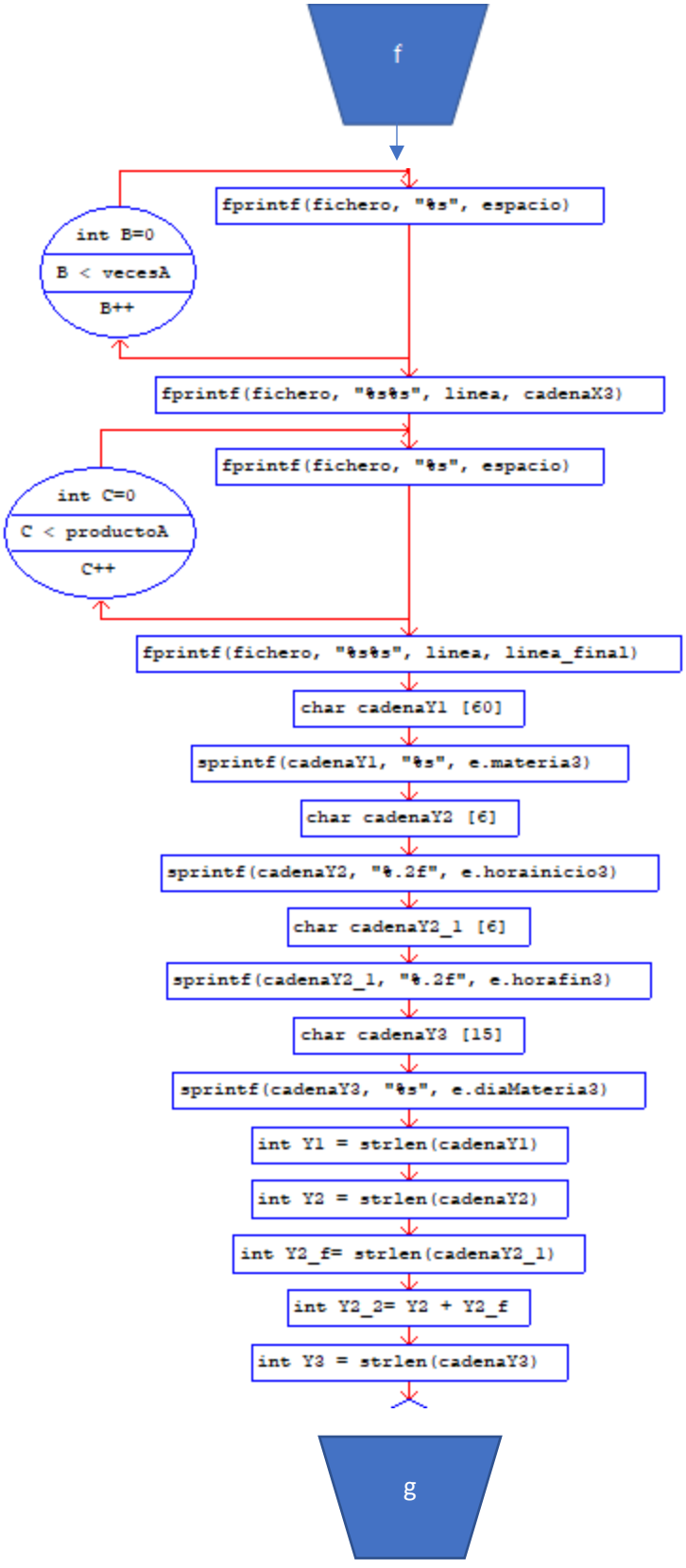

a
char cadenaA [] = "|==|====================|====================|\n"
char cadenaB [] = "|_____________________Materia_____________________|______Horas_____|_____Dia_____|\n"
char cadenaC [] = "|==|====================|====================|\n"
int divisoresA1
int vecesA1
int productoA1
int A1_1 = strlen(cadenaI)
int A2_1 = strlen(cadenaI2)
int A3_1 = strlen(cadenaI3)
f
A1_1 < 51
v
divisoresA1 = 0
divisoresA1 = 51 - (A1_1)
f
(A2_1) < 11
v
vecesA1 = 0
vecesA1 = 11 - (A2_1)
f
A3_1 < 13
v
productoA1 = 0
productoA1 = 13 - (A3_1)
char espacioA [] = "_"
b

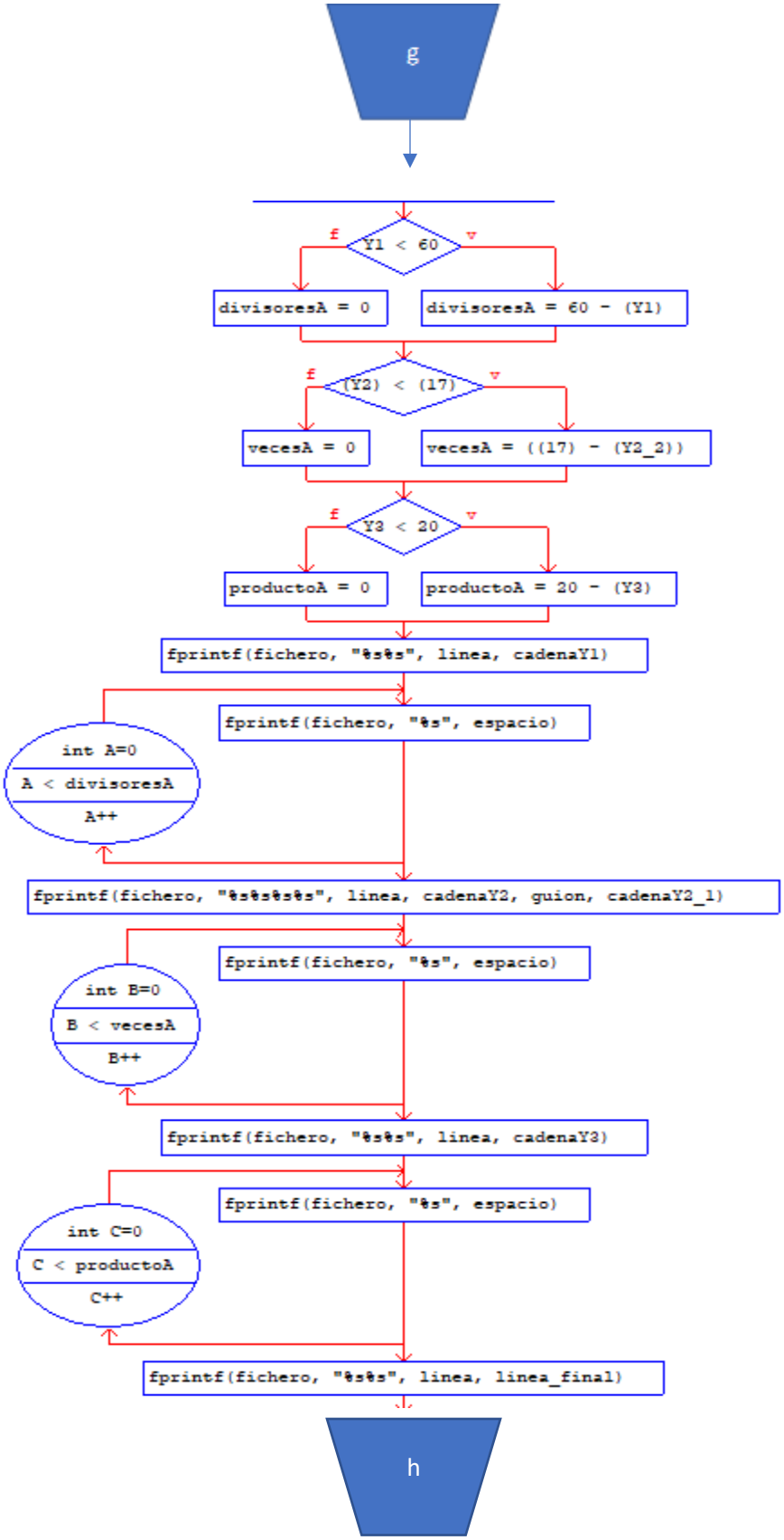

b
fprintf(fichero, "%s%s%s",cadenaa, cadenab, cadenaI)
fprintf(fichero, "%s", espacioA)
int A=0
A < divisoresAl
A++
fprintf(fichero, "%s%s", cadenac, cadenaI2)
fprintf(fichero, "%s", espacioA)
int B=0
B < vecesAl
B++
fprintf(fichero, "%s%s", cadenad, cadenaI3)
fprintf(fichero, "%s", espacioA)
int C=0
C < productoAl
C++
fprintf(fichero, "%s", cadenae)
fprintf (fichero, "%s%s%s", cadenaA, cadenaB, cadenaC)
fclose(fichero)
1
case (3):
int divisoresA
int vecesA
int productoA
c

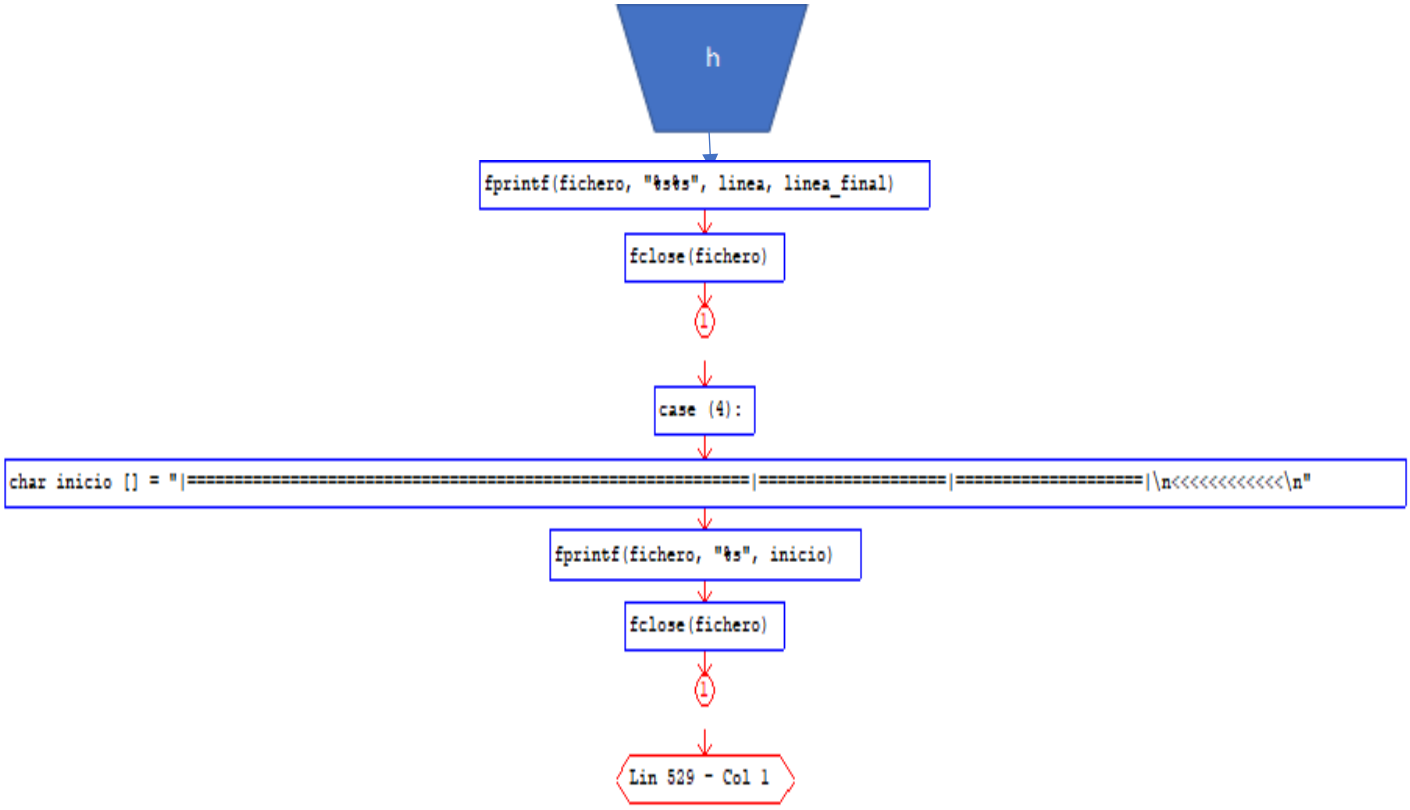

c
char linea [] = "|"
char espacio [] = "_"
char guion [] = "_-_"
char linea_final [] = "\n"
char cadenaA1 [60]
sprintf(cadenaA1, "%s", e.material)
char cadenaA2 [6]
sprintf(cadenaA2, "%.2f", e.horaInicioo)
char cadenaA2_1 [6]
sprintf(cadenaA2_1, "%.2f", e.horaFin)
char cadenaA3 [15]
sprintf(cadenaA3, "%s", e.diaMaterial)
int A1 = strlen(cadenaA1)
int A2 = strlen(cadenaA2)
int A2_f= strlen(cadenaA2_1)
int A2_2= A2 + A2_f
int A3 = strlen(cadenaA3)
f
A1 < 60
v
divisoresA = 0
divisoresA = 60 - (A1)
f
(A2) < (17)
v
vecesA = 0
vecesA = ((17) - (A2_2))
d


d
A3 < 20
f
v
productoA = 0
productoA = 20 - (A3)
fprintf(fichero, "%s%s", linea, cadenaA1)
fprintf(fichero, "%s", espacio)
int A=0
A < divisoresA
A++
fprintf(fichero, "%s%s%s%s", linea, cadenaA2, guion, cadenaA2_1)
fprintf(fichero, "%s", espacio)
int B=0
B < vecesA
B++
fprintf(fichero, "%s%s", linea, cadenaA3)
fprintf(fichero, "%s", espacio)
int C=0
C < productoA
C++
fprintf(fichero, "%s%s", linea, linea_final)
char cadenaX1 [60]
sprintf(cadenaX1, "%s", e.materia2)
char cadenaX2 [6]
sprintf(cadenaX2, "%.2f", e.horaInicio2)
char cadenaX2_1 [6]
sprintf(cadenaX2_1, "%.2f", e.horafin2)
e

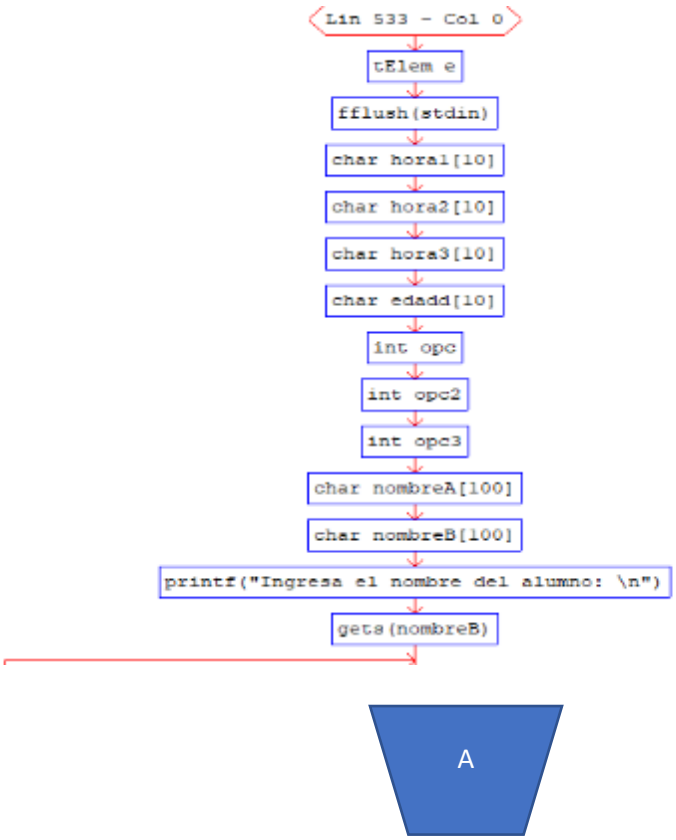

e
char cadenaX3 [15]
sprintf(cadenaX3, "%s", e.diaMateria2)
int X1 = strlen(cadenaX1)
int X2 = strlen(cadenaX2)
int X2_f= strlen(cadenaX2_1)
int X2_2= X2 + X2_f
int X3 = strlen(cadenaX3)
f
X1 < 60
v
divisoresA = 0
divisoresA = 60 - (X1)
f
(X2) < (17)
v
vecesA = 0
vecesA = ((17) - (X2_2))
f
X3 < 20
v
productoA = 0
productoA = 20 - (X3)
fprintf(fichero, "%s%s", linea, cadenaX1)
fprintf(fichero, "%s", espacio)
int A=0
A < divisoresA
A++
fprintf(fichero, "%s%s%s%s", linea, cadenaX2, guion, cadenaX2_1)
f

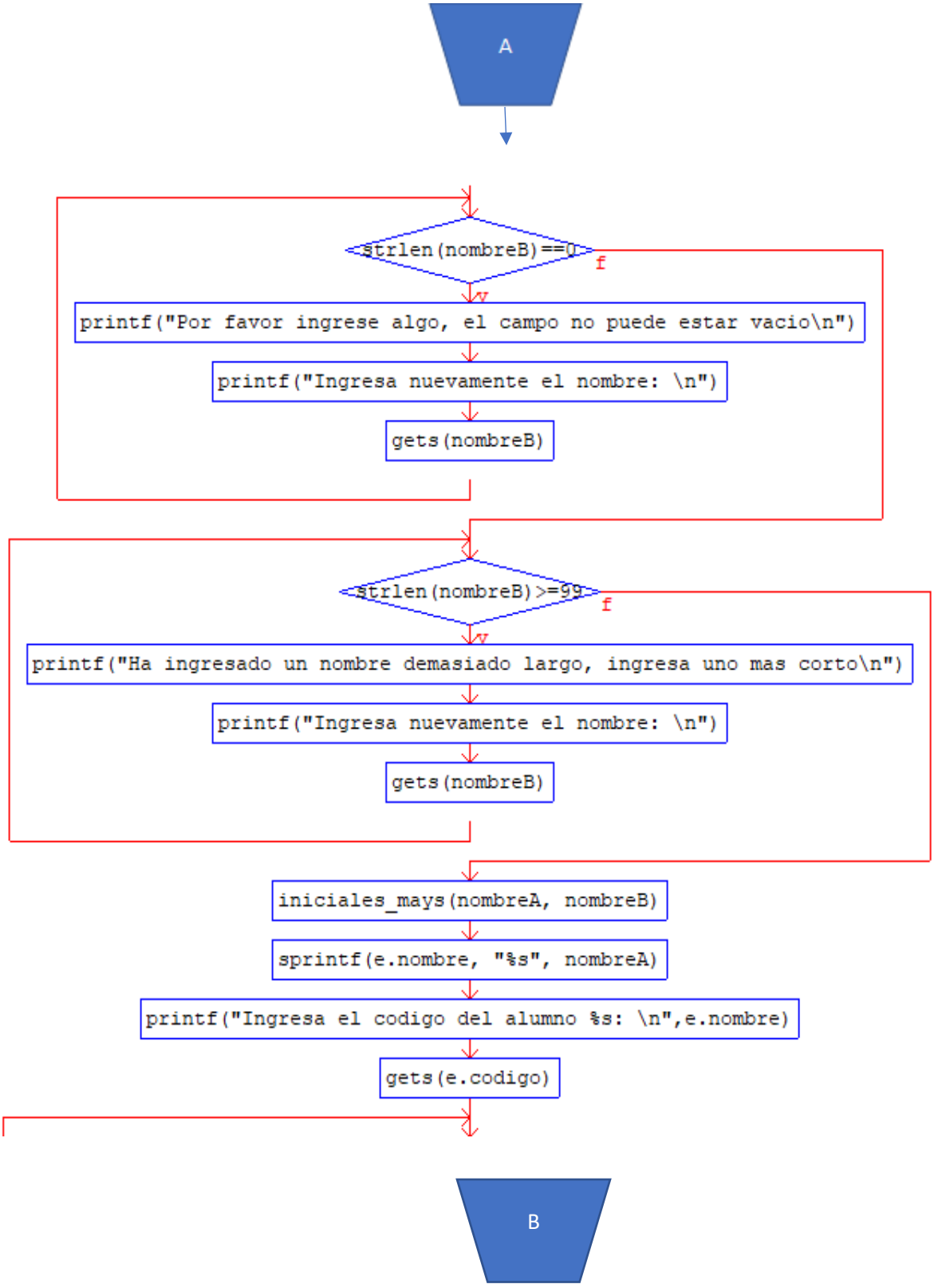

f
fprintf(fichero, "%s", espacio)
int B=0
B < vecesA
B++
fprintf(fichero, "%s%s", linea, cadenaX3)
fprintf(fichero, "%s", espacio)
int C=0
C < productoA
C++
fprintf(fichero, "%s%s", linea, linea_final)
char cadenaY1 [60]
sprintf(cadenaY1, "%s", e.materia3)
char cadenaY2 [6]
sprintf(cadenaY2, "%.2f", e.horainicio3)
char cadenaY2_1 [6]
sprintf(cadenaY2_1, "%.2f", e.horafin3)
char cadenaY3 [15]
sprintf(cadenaY3, "%s", e.diaMateria3)
int Y1 = strlen(cadenaY1)
int Y2 = strlen(cadenaY2)
int Y2_f= strlen(cadenaY2_1)
int Y2_2= Y2 + Y2_f
int Y3 = strlen(cadenaY3)
g

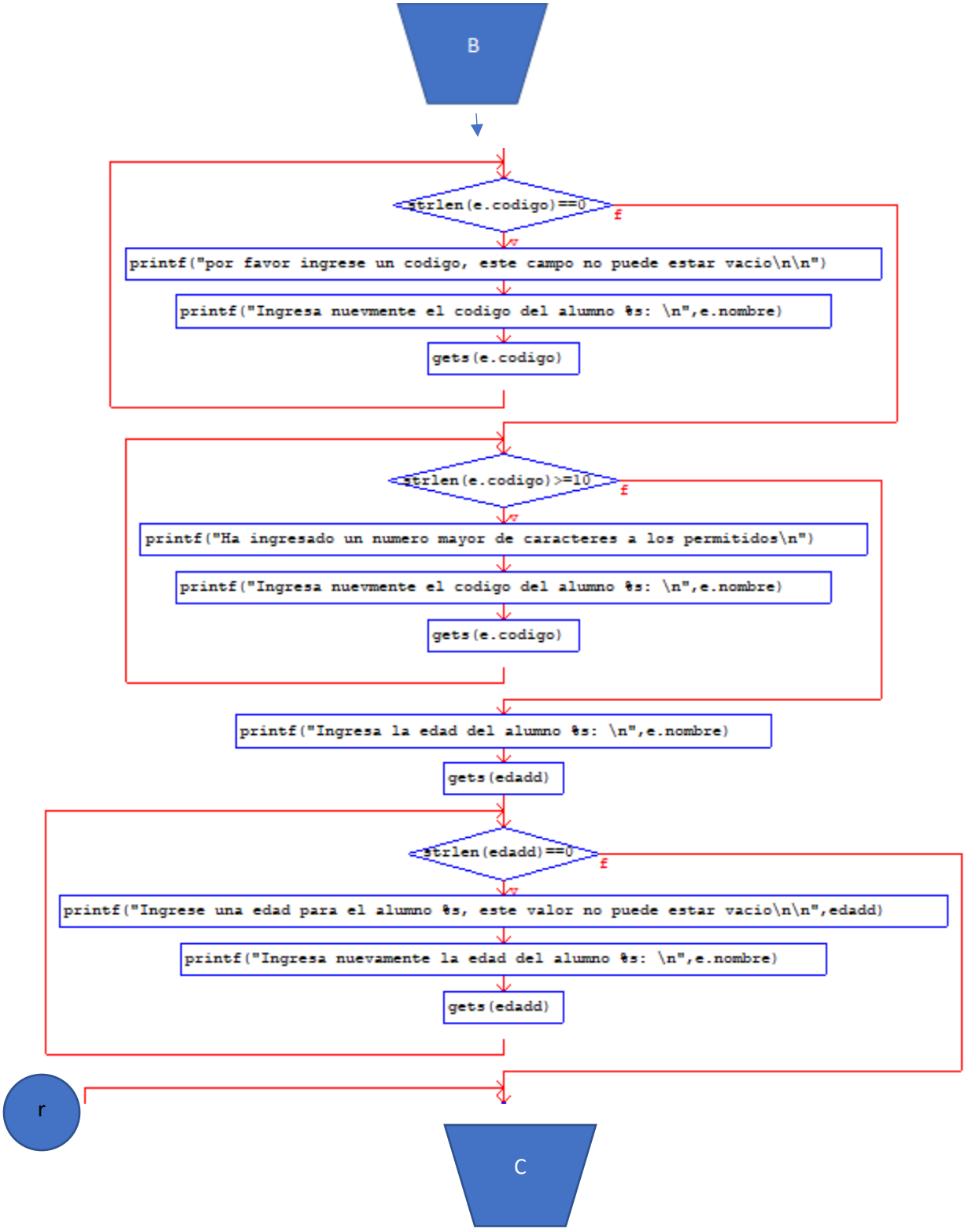

g
Y1 < 60
f
v
divisoresA = 0
divisoresA = 60 - (Y1)
(Y2) < (17)
vecesA = 0
vecesA = ((17) - (Y2_2))
Y3 < 20
productoA = 0
productoA = 20 - (Y3)
fprintf(fichero, "%s%s", linea, cadenaY1)
fprintf(fichero, "%s", espacio)
int A=0
A < divisoresA
A++
fprintf(fichero, "%s%s%s%s", linea, cadenaY2, guion, cadenaY2_1)
fprintf(fichero, "%s", espacio)
int B=0
B < vecesA
B++
fprintf(fichero, "%s%s", linea, cadenaY3)
fprintf(fichero, "%s", espacio)
int C=0
C < productoA
C++
fprintf(fichero, "%s%s", linea, linea_final)
h

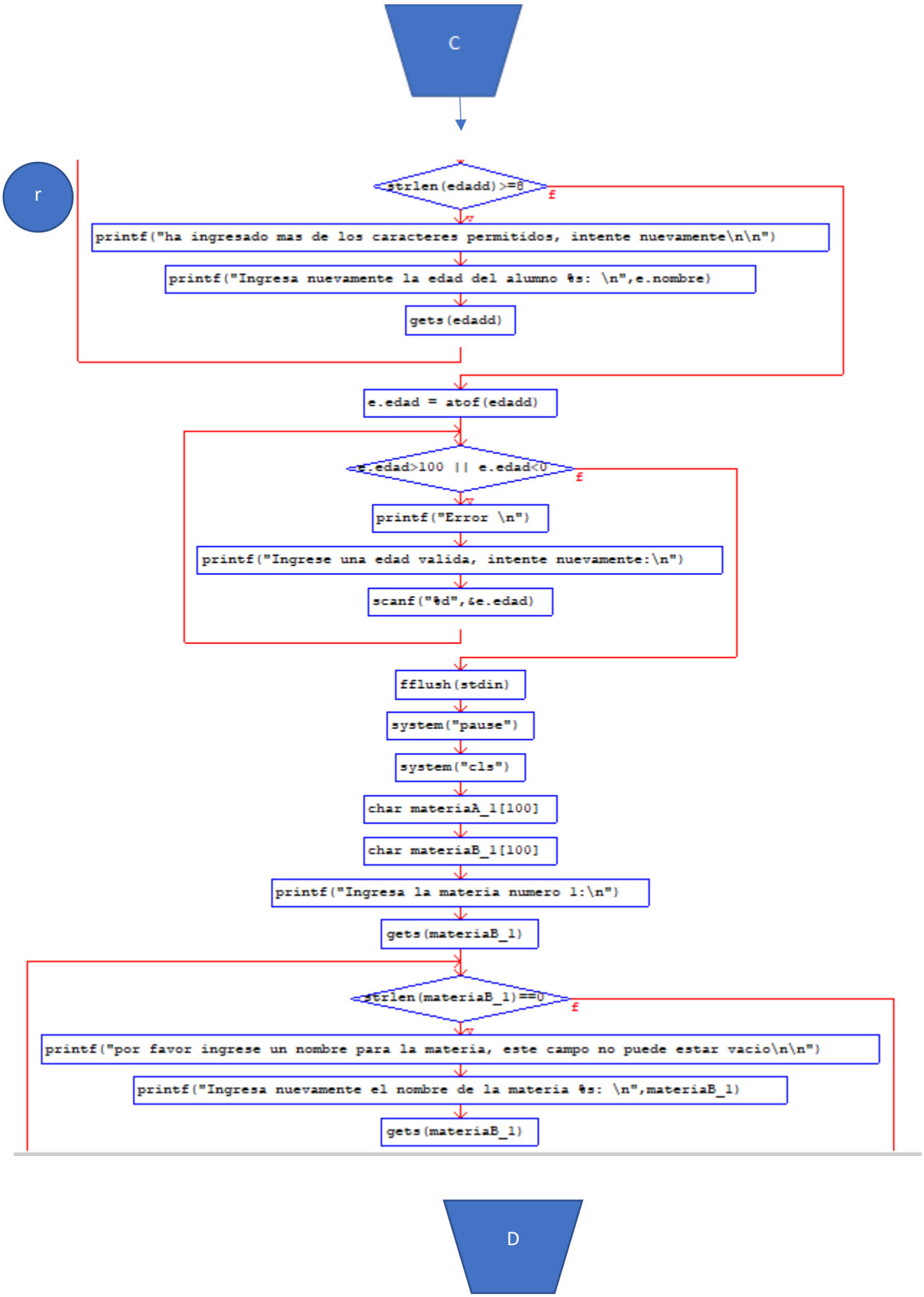


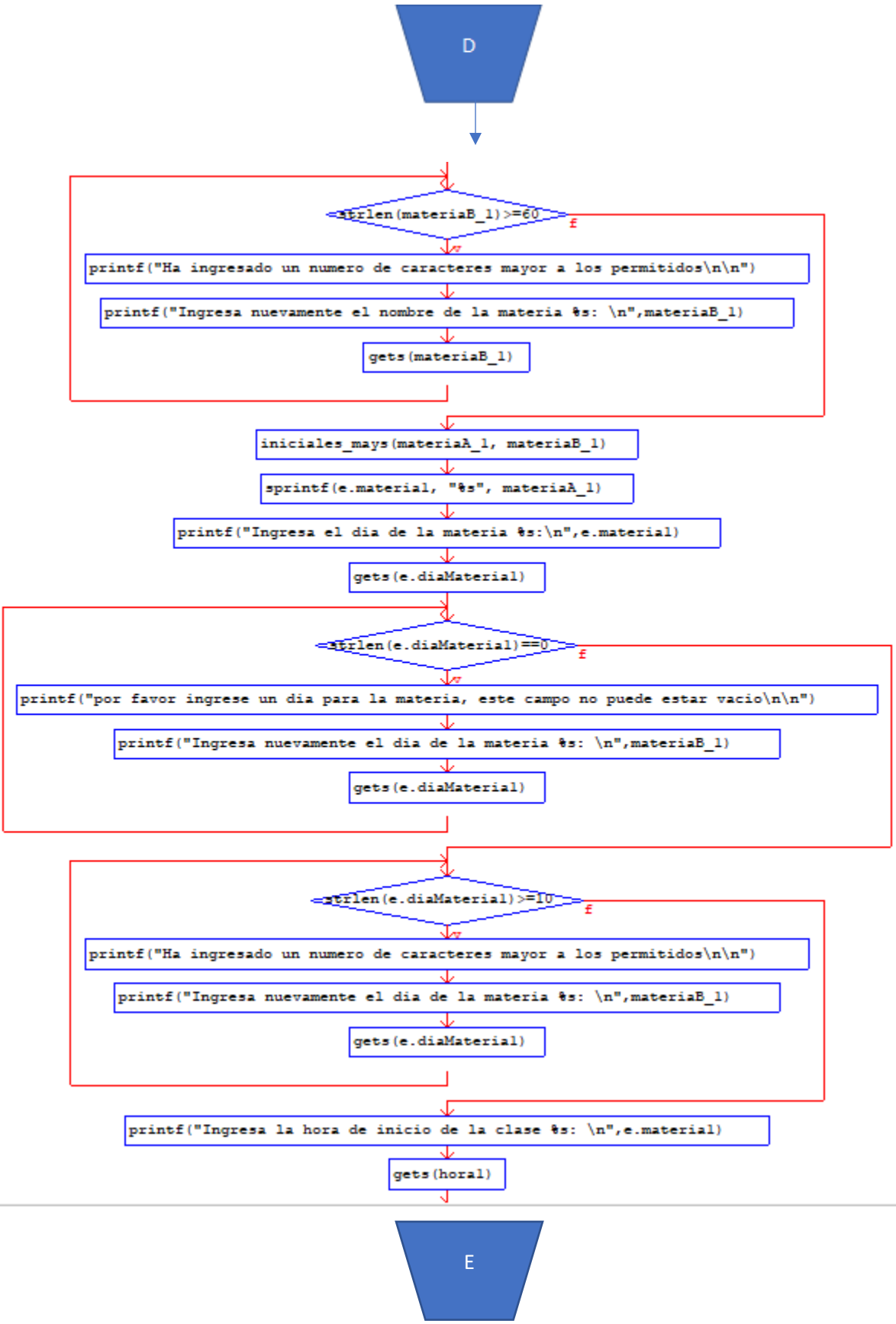
Función nuevoElem

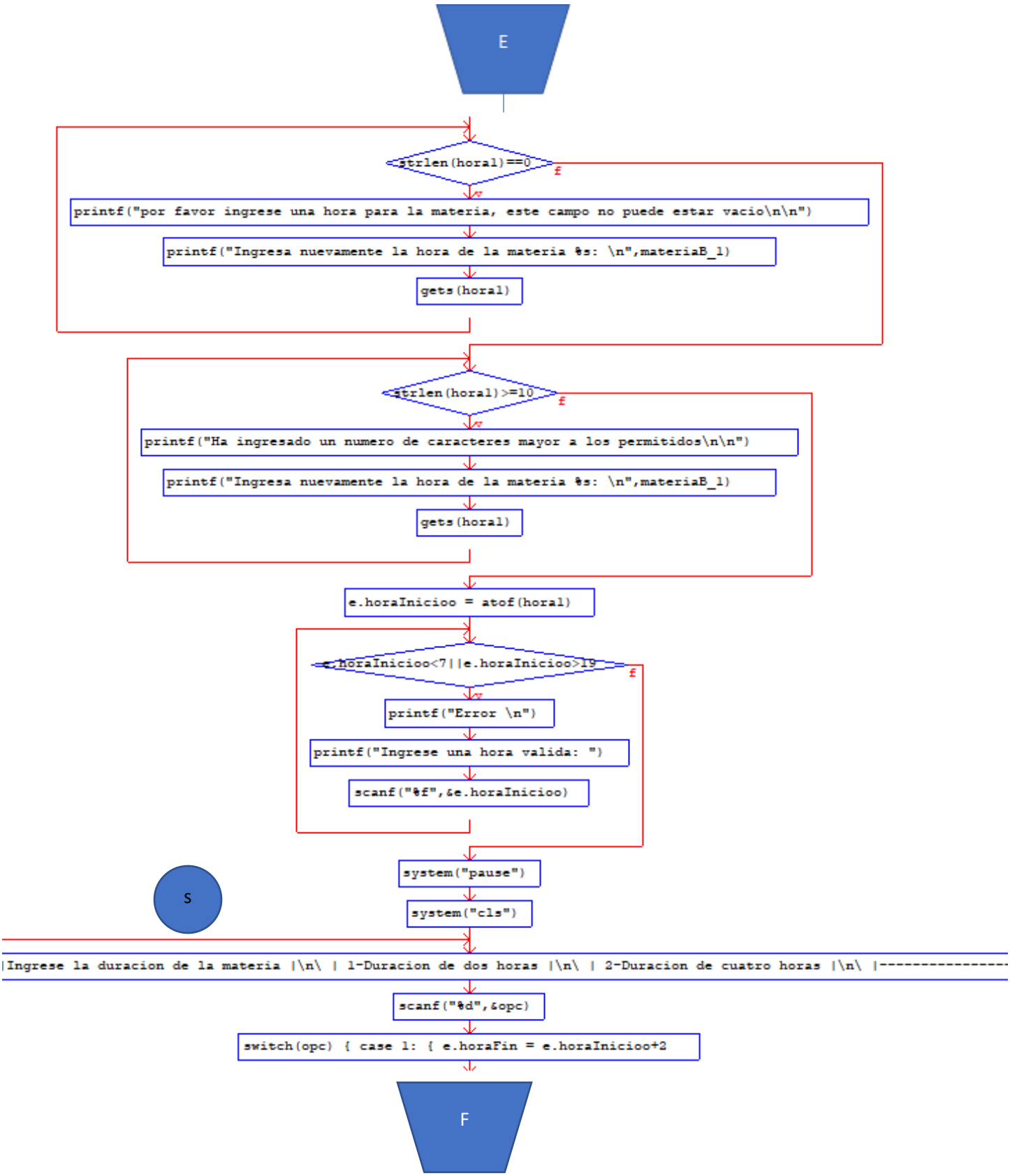


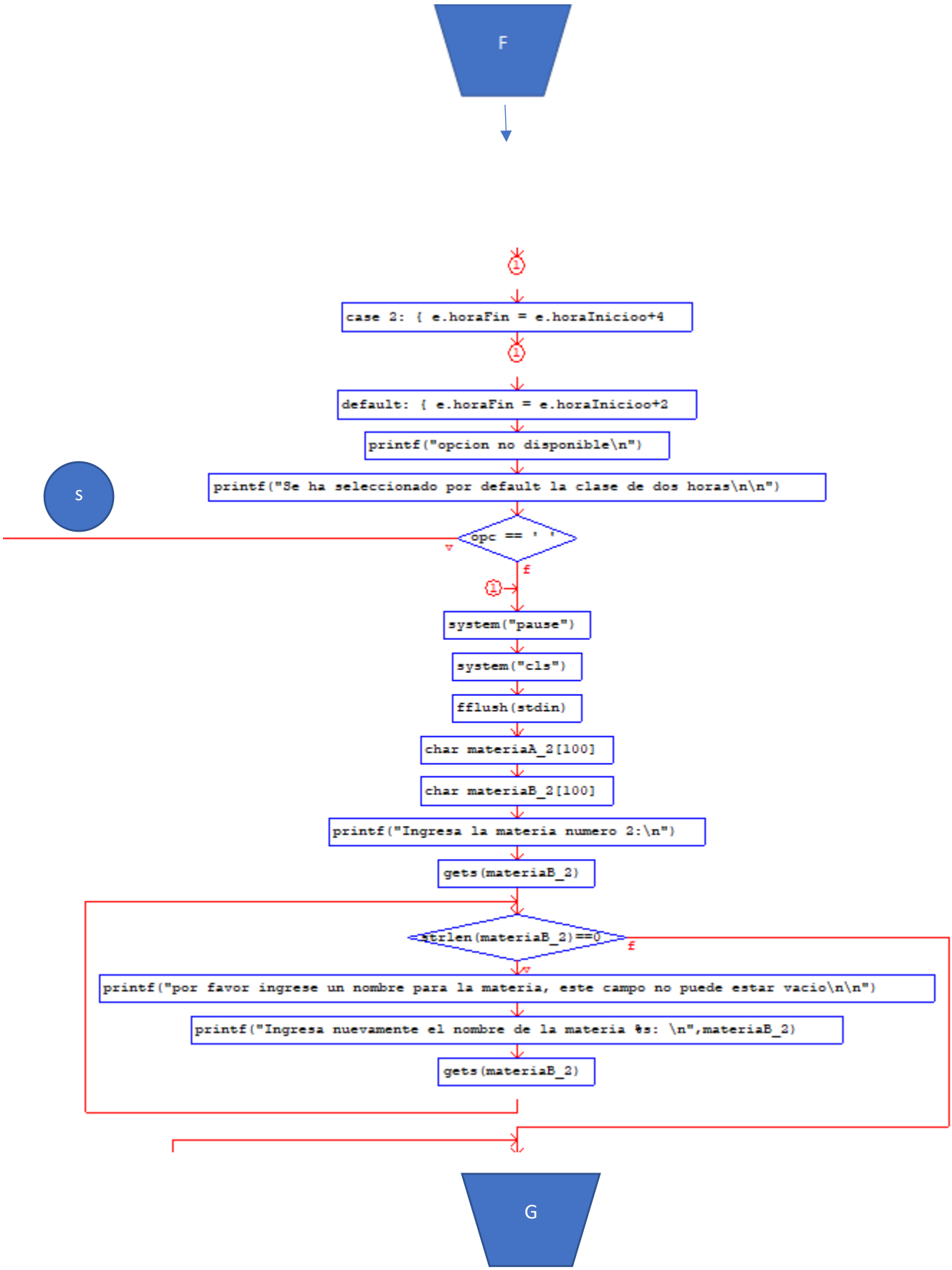

A
strlen(nombreB)==0
f
v
printf("Por favor ingrese algo, el campo no puede estar vacio\n")
printf("Ingresa nuevamente el nombre: \n")
gets(nombreB)
strlen(nombreB)>=99
f
v
printf("Ha ingresado un nombre demasiado largo, ingresa uno mas corto\n")
printf("Ingresa nuevamente el nombre: \n")
gets(nombreB)
iniciales_mays(nombreA, nombreB)
sprintf(e.nombre, "%s", nombreA)
printf("Ingresa el codigo del alumno %s: \n",e.nombre)
gets(e.codigo)
B

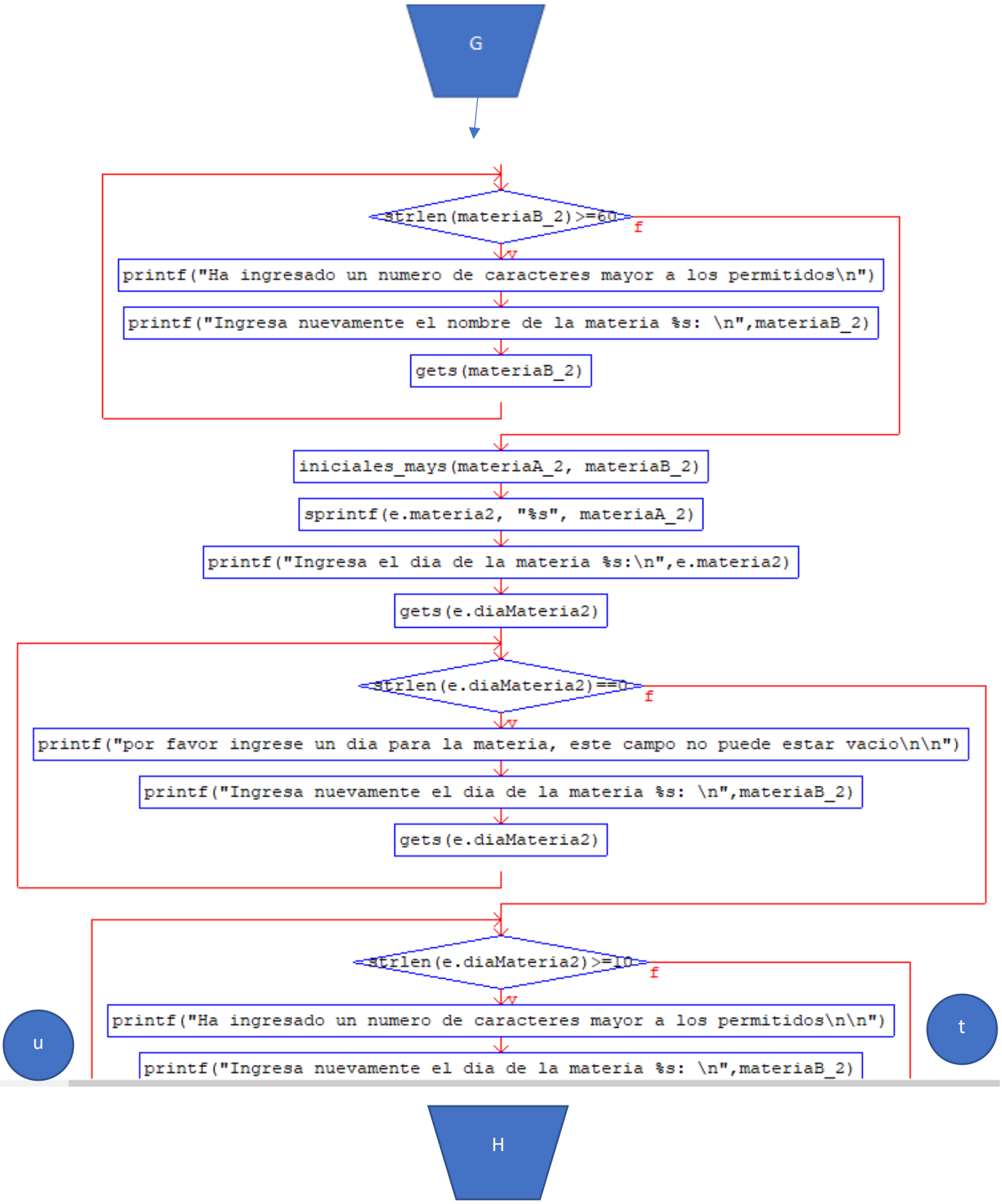

B
strlen(e.codigo)==0
f
v
printf("por favor ingrese un codigo, este campo no puede estar vacio\n\n")
printf("Ingresa nuevmente el codigo del alumno %s: \n",e.nombre)
gets(e.codigo)
strlen(e.codigo)>=10
f
v
printf("Ha ingresado un numero mayor de caracteres a los permitidos\n")
printf("Ingresa nuevmente el codigo del alumno %s: \n",e.nombre)
gets(e.codigo)
printf("Ingresa la edad del alumno %s: \n",e.nombre)
gets(edadd)
strlen(edadd)==0
f
v
printf("Ingrese una edad para el alumno %s, este valor no puede estar vacio\n\n",edadd)
printf("Ingresa nuevamente la edad del alumno %s: \n",e.nombre)
gets(edadd)
r
C

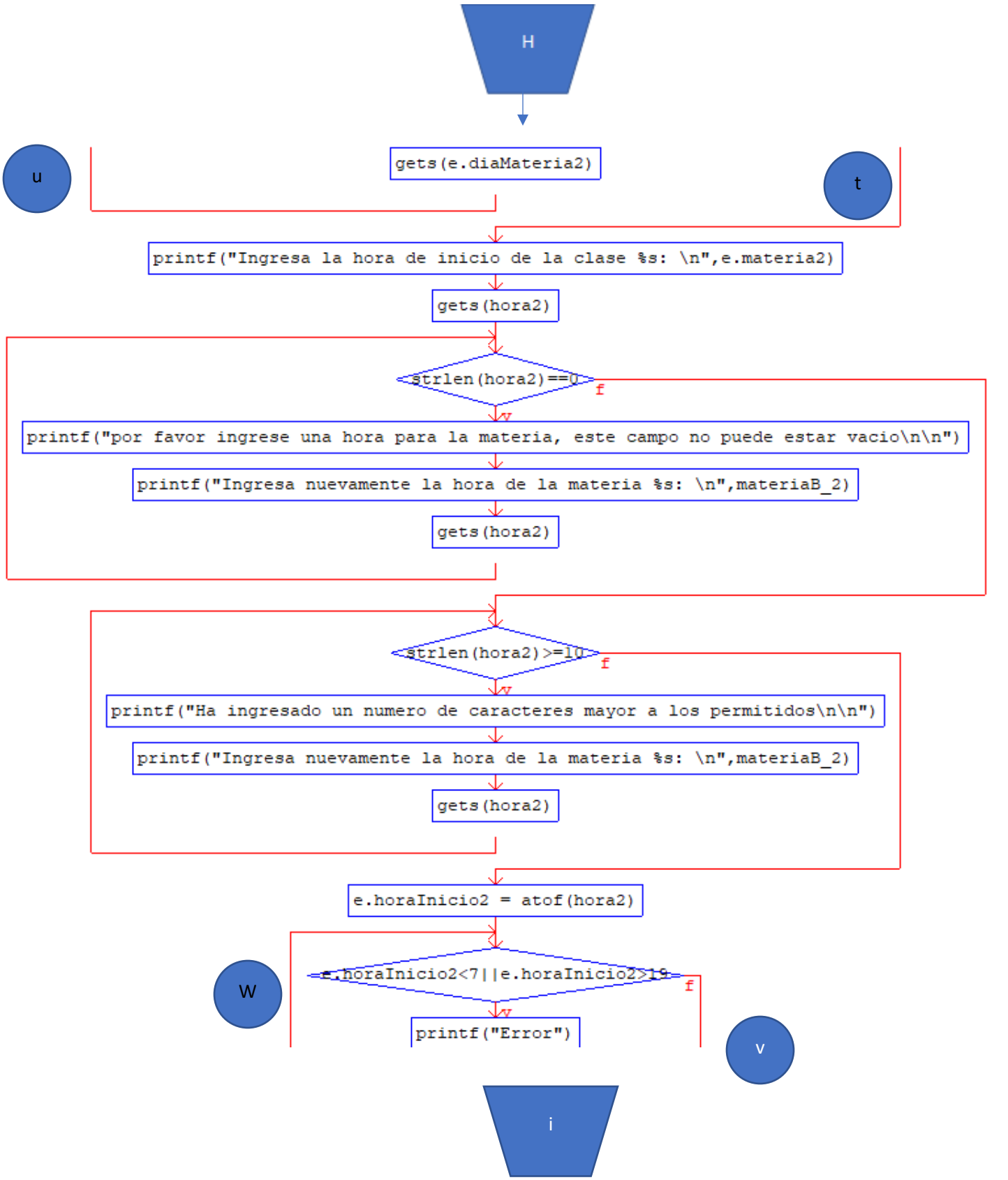

C
r
strlen(edadd)>=8
f
printf("ha ingresado mas de los caracteres permitidos, intente nuevamente\n\n")
printf("Ingresa nuevamente la edad del alumno %s: \n",e.nombre)
gets(edadd)
e.edad = atof(edadd)
e.edad>100 || e.edad<0
f
printf("Error \n")
printf("Ingrese una edad valida, intente nuevamente:\n")
scanf("%d",&e.edad)
fflush(stdin)
system("pause")
system("cls")
char materiaA_1[100]
char materiaB_1[100]
printf("Ingresa la materia numero 1:\n")
gets(materiaB_1)
strlen(materiaB_1)==0
f
printf("por favor ingrese un nombre para la materia, este campo no puede estar vacio\n\n")
printf("Ingresa nuevamente el nombre de la materia %s: \n",materiaB_1)
gets(materiaB_1)
D

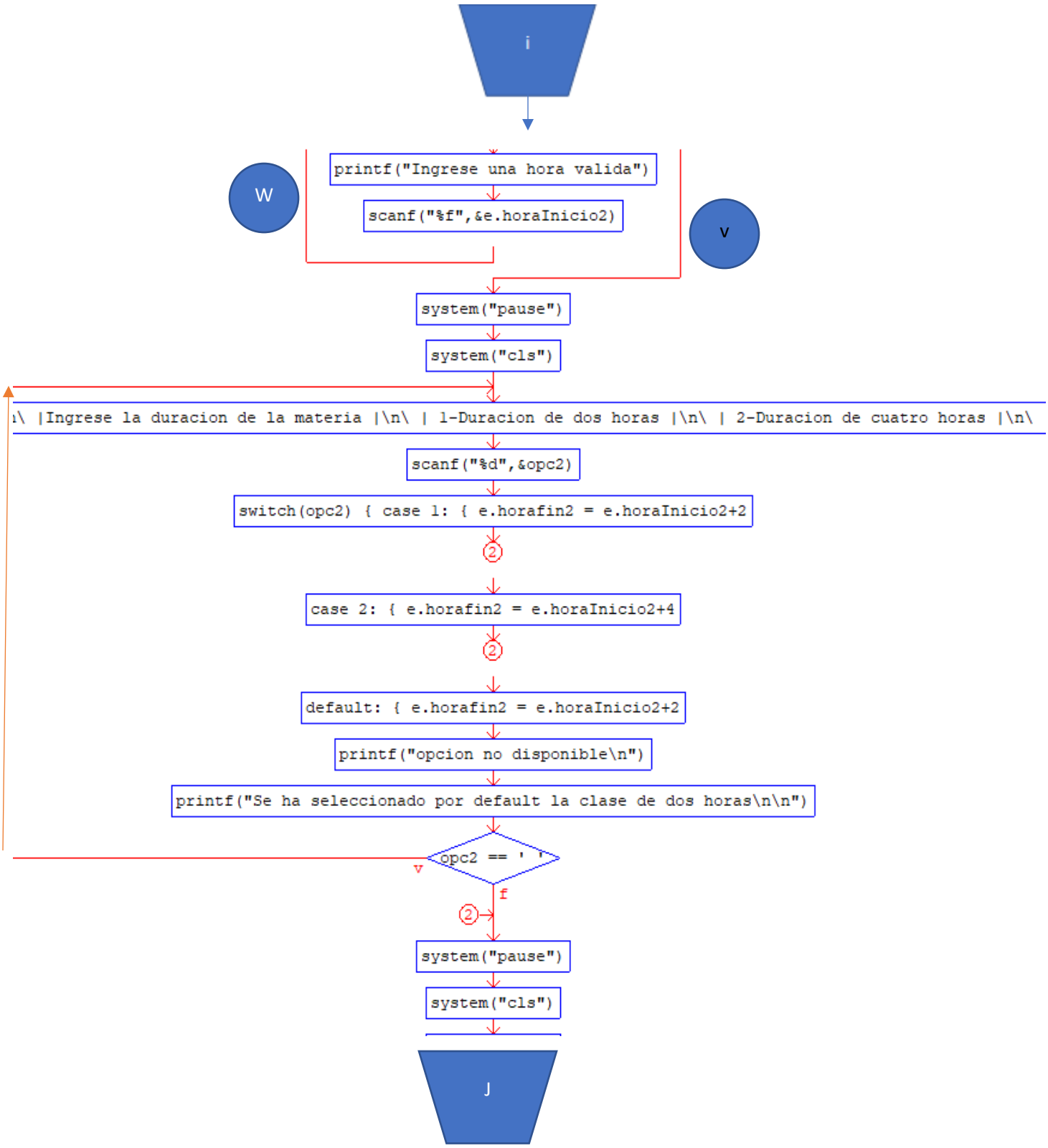

D
strlen(materiaB_1)>=60
f
v
printf("Ha ingresado un numero de caracteres mayor a los permitidos\n\n")
printf("Ingresa nuevamente el nombre de la materia %s: \n",materiaB_1)
gets(materiaB_1)
iniciales_mays(materiaA_1, materiaB_1)
sprintf(e.material, "%s", materiaA_1)
printf("Ingresa el dia de la materia %s:\n",e.material)
gets(e.diaMaterial)
strlen(e.diaMaterial)==0
f
v
printf("por favor ingrese un dia para la materia, este campo no puede estar vacio\n\n")
printf("Ingresa nuevamente el dia de la materia %s: \n",materiaB_1)
gets(e.diaMaterial)
strlen(e.diaMaterial)>=10
f
v
printf("Ha ingresado un numero de caracteres mayor a los permitidos\n\n")
printf("Ingresa nuevamente el dia de la materia %s: \n",materiaB_1)
gets(e.diaMaterial)
printf("Ingresa la hora de inicio de la clase %s: \n",e.material)
gets(hora1)
E

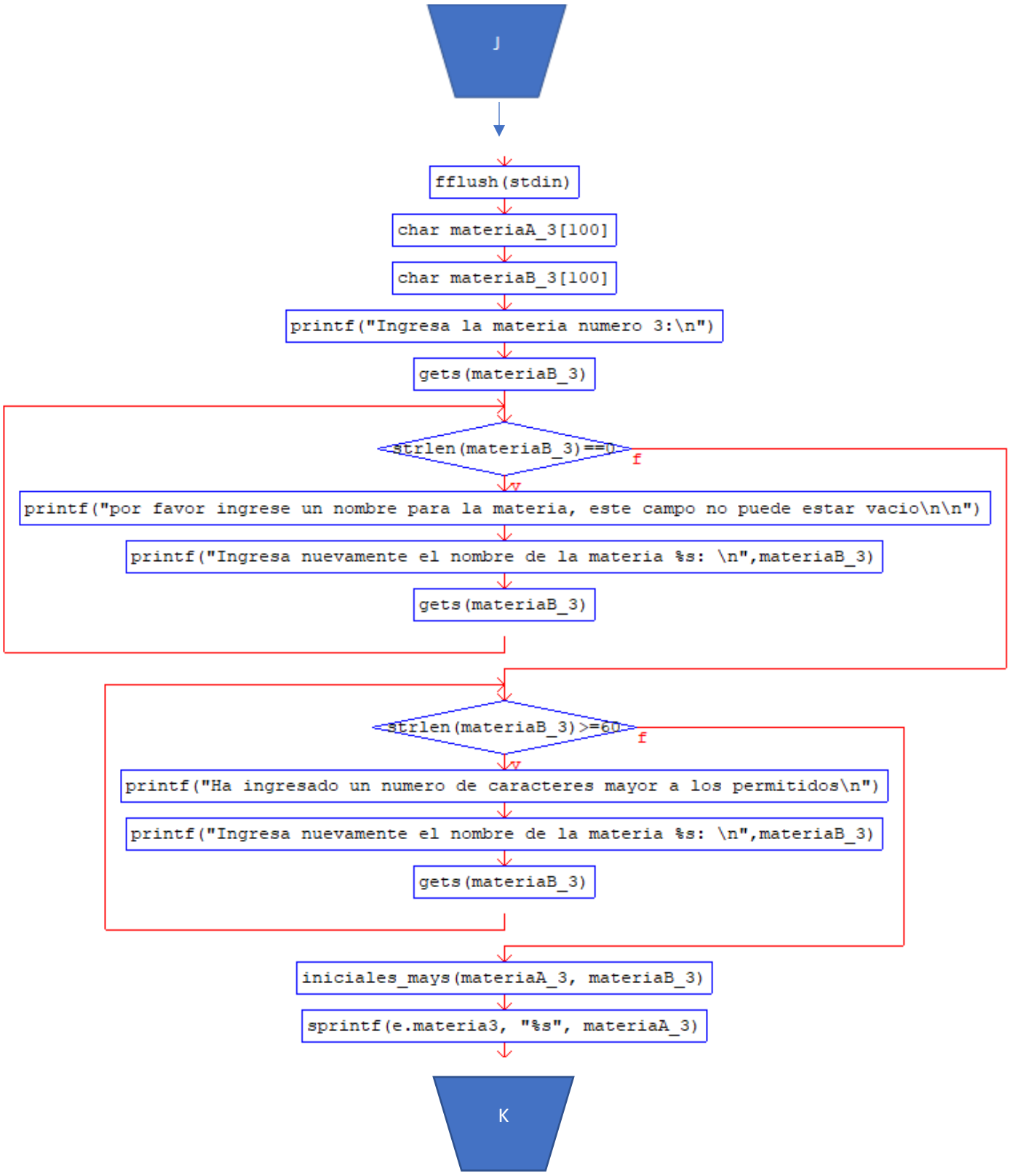

E
strlen(horal)==0
f
printf("por favor ingrese una hora para la materia, este campo no puede estar vacio\n\n")
printf("Ingresa nuevamente la hora de la materia %s: \n",materiaB_1)
gets(horal)
strlen(horal)>=10
f
printf("Ha ingresado un numero de caracteres mayor a los permitidos\n\n")
printf("Ingresa nuevamente la hora de la materia %s: \n",materiaB_1)
gets(horal)
e.horaInicioo = atof(horal)
e.horaInicioo<7||e.horaInicioo>19
f
printf("Error \n")
printf("Ingrese una hora valida: ")
scanf("%f",&e.horaInicioo)
system("pause")
system("cls")
S
|Ingrese la duracion de la materia |\n\ | 1-Duracion de dos horas |\n\ | 2-Duracion de cuatro horas |\n\ |----------
scanf("%d",&opc)
switch(opc) { case 1: { e.horaFin = e.horaInicioo+2
F

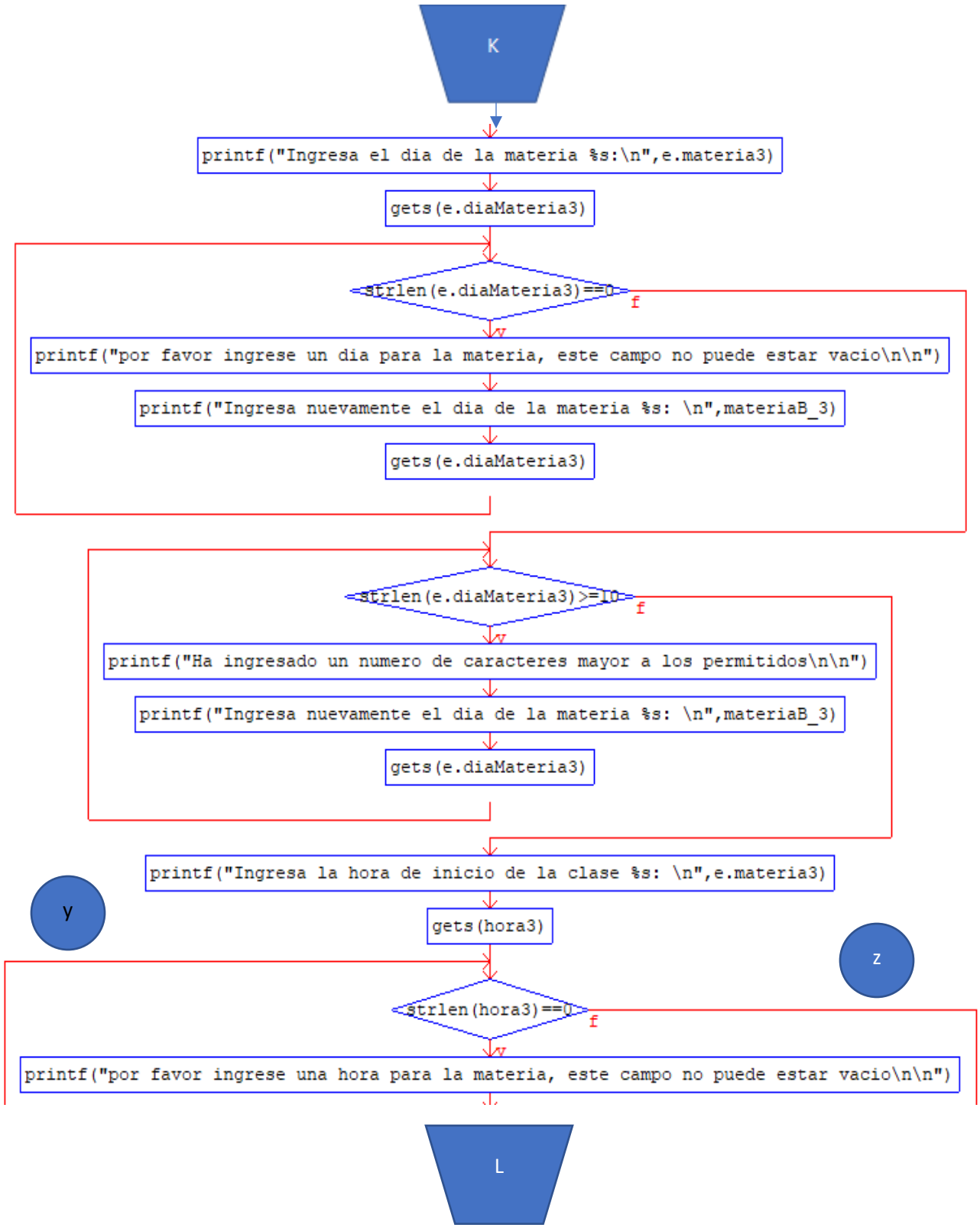

F
1
case 2: { e.horaFin = e.horaInicioo+4
1
default: { e.horaFin = e.horaInicioo+2
printf("opcion no disponible\n")
printf("Se ha seleccionado por default la clase de dos horas\n\n")
S
opc == ' '
f
1
system("pause")
system("cls")
fflush(stdin)
char materiaA_2[100]
char materiaB_2[100]
printf("Ingresa la materia numero 2:\n")
gets(materiaB_2)
strlen(materiaB_2)==0
f
printf("por favor ingrese un nombre para la materia, este campo no puede estar vacio\n\n")
printf("Ingresa nuevamente el nombre de la materia %s: \n",materiaB_2)
gets(materiaB_2)
G

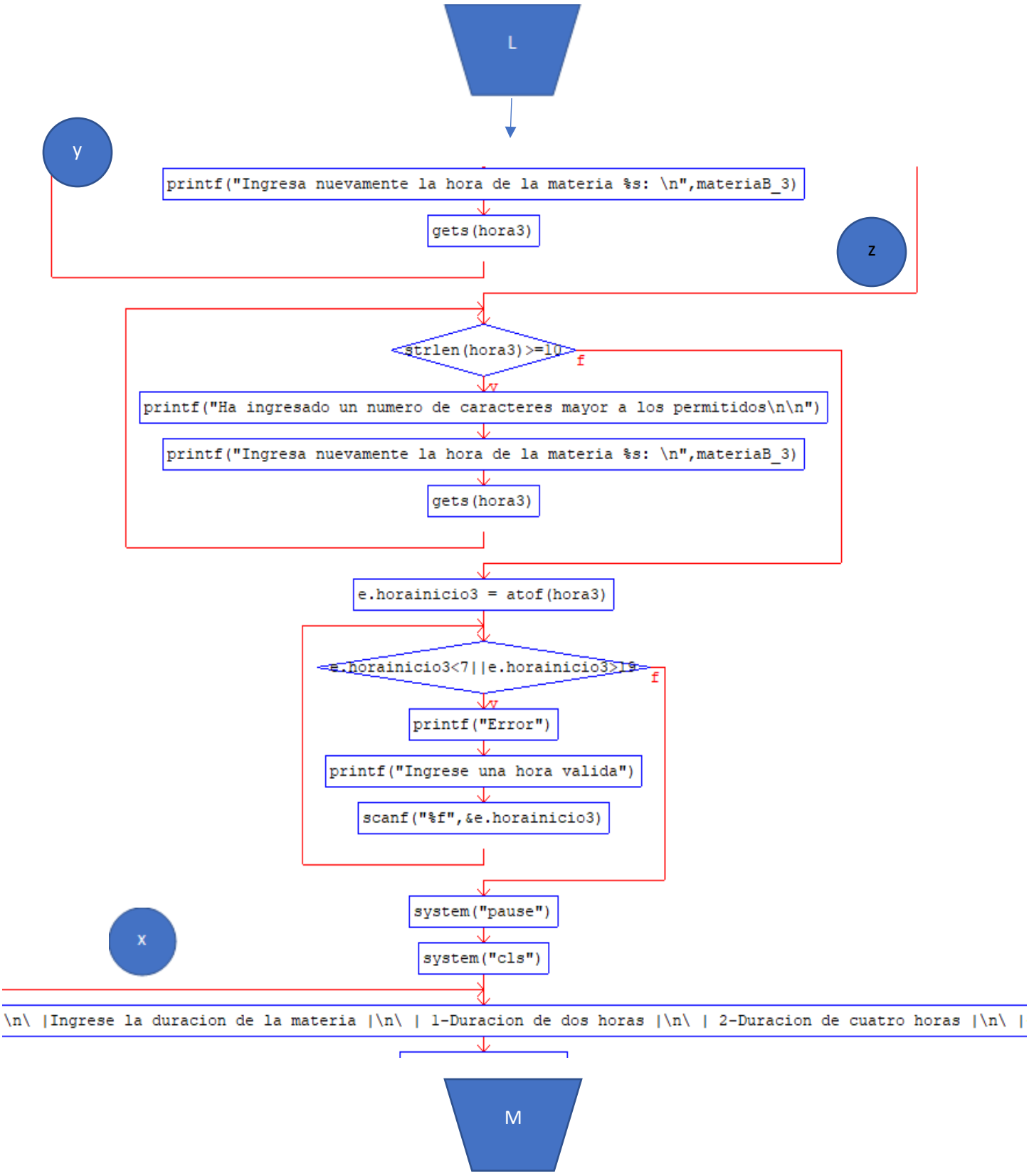

G
strlen(materiaB_2)>=60
f
v
printf("Ha ingresado un numero de caracteres mayor a los permitidos\n")
printf("Ingresa nuevamente el nombre de la materia %s: \n",materiaB_2)
gets(materiaB_2)
iniciales_mays(materiaA_2, materiaB_2)
sprintf(e.materia2, "%s", materiaA_2)
printf("Ingresa el dia de la materia %s:\n",e.materia2)
gets(e.diaMateria2)
strlen(e.diaMateria2)==0
f
v
printf("por favor ingrese un dia para la materia, este campo no puede estar vacio\n\n")
printf("Ingresa nuevamente el dia de la materia %s: \n",materiaB_2)
gets(e.diaMateria2)
strlen(e.diaMateria2)>=10
f
v
printf("Ha ingresado un numero de caracteres mayor a los permitidos\n\n")
printf("Ingresa nuevamente el dia de la materia %s: \n",materiaB_2)
u
t
H

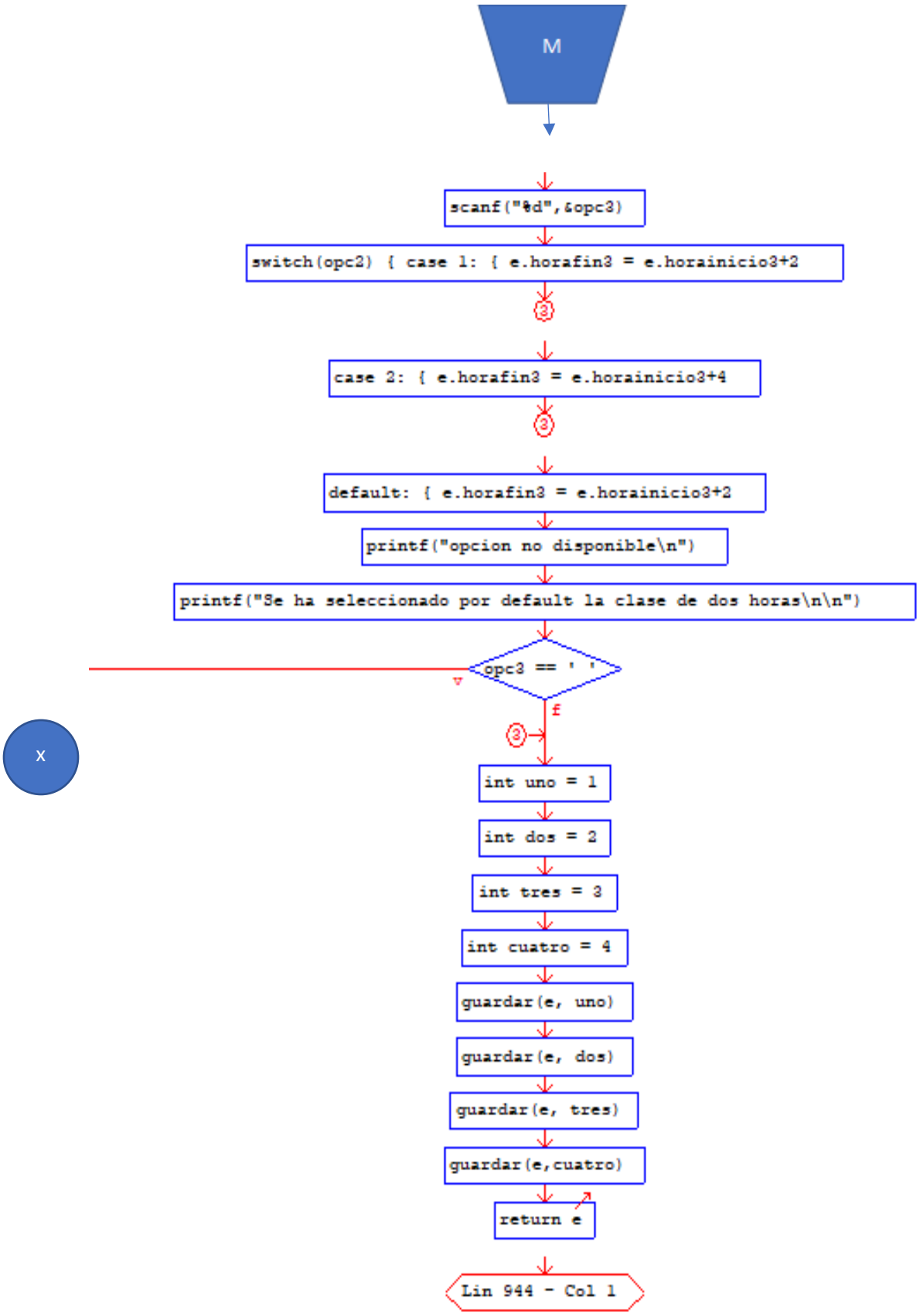

H
u
gets(e.diaMateria2)
t
printf("Ingresa la hora de inicio de la clase %s: \n",e.materia2)
gets(hora2)
strlen(hora2)==0
f
v
printf("por favor ingrese una hora para la materia, este campo no puede estar vacio\n\n")
printf("Ingresa nuevamente la hora de la materia %s: \n",materiaB_2)
gets(hora2)
strlen(hora2)>=10
f
v
printf("Ha ingresado un numero de caracteres mayor a los permitidos\n\n")
printf("Ingresa nuevamente la hora de la materia %s: \n",materiaB_2)
gets(hora2)
e.horaInicio2 = atof(hora2)
W
e.horaInicio2<7||e.horaInicio2>19
f
v
printf("Error")
v
i


i
W
printf("Ingrese una hora valida")
scanf("%f",&e.horaInicio2)
v
system("pause")
system("cls")
\ |Ingrese la duracion de la materia |\n\ | 1-Duracion de dos horas |\n\ | 2-Duracion de cuatro horas |\n\
scanf("%d",&opc2)
switch(opc2) { case 1: { e.horafin2 = e.horaInicio2+2
2
case 2: { e.horafin2 = e.horaInicio2+4
2
default: { e.horafin2 = e.horaInicio2+2
printf("opcion no disponible\n")
printf("Se ha seleccionado por default la clase de dos horas\n\n")
opc2 == ' '
v
f
2
system("pause")
system("cls")
J

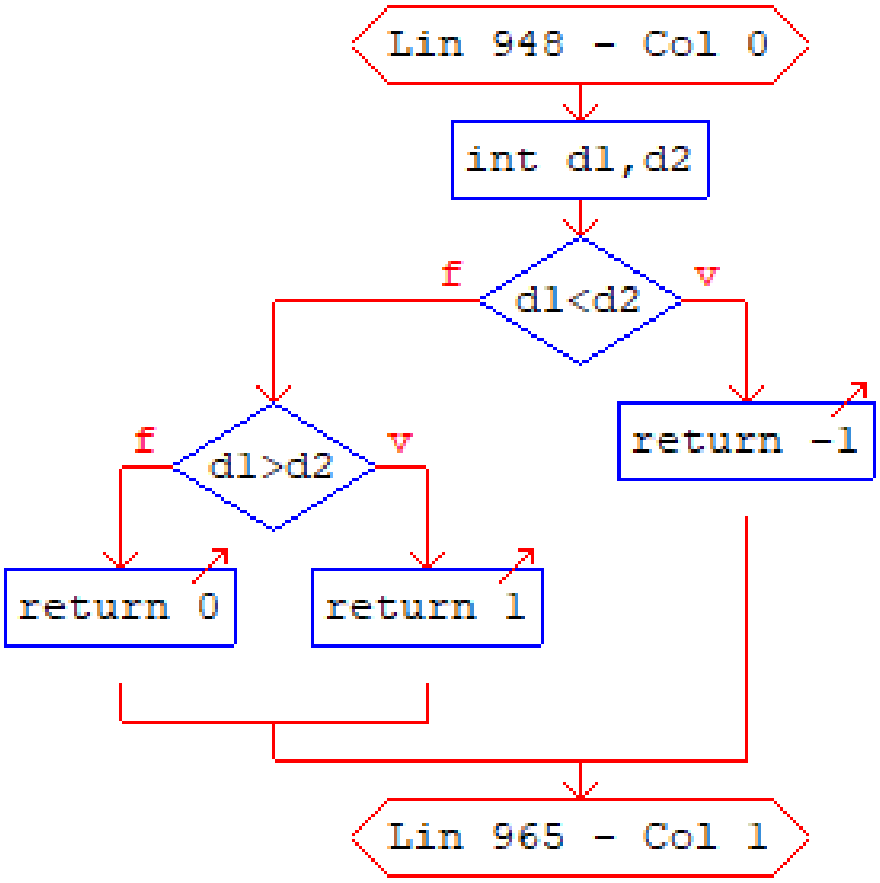

J
fflush(stdin)
char materiaA_3[100]
char materiaB_3[100]
printf("Ingresa la materia numero 3:\n")
gets(materiaB_3)
strlen(materiaB_3)==0
f
v
printf("por favor ingrese un nombre para la materia, este campo no puede estar vacio\n\n")
printf("Ingresa nuevamente el nombre de la materia %s: \n",materiaB_3)
gets(materiaB_3)
strlen(materiaB_3)>=60
f
v
printf("Ha ingresado un numero de caracteres mayor a los permitidos\n")
printf("Ingresa nuevamente el nombre de la materia %s: \n",materiaB_3)
gets(materiaB_3)
iniciales_mays(materiaA_3, materiaB_3)
sprintf(e.materia3, "%s", materiaA_3)
K


K
printf("Ingresa el dia de la materia %s:\n",e.materia3)
gets(e.diaMateria3)
strlen(e.diaMateria3)==0
f
v
printf("por favor ingrese un dia para la materia, este campo no puede estar vacio\n\n")
printf("Ingresa nuevamente el dia de la materia %s: \n",materiaB_3)
gets(e.diaMateria3)
strlen(e.diaMateria3)>=10
f
v
printf("Ha ingresado un numero de caracteres mayor a los permitidos\n\n")
printf("Ingresa nuevamente el dia de la materia %s: \n",materiaB_3)
gets(e.diaMateria3)
printf("Ingresa la hora de inicio de la clase %s: \n",e.materia3)
y
gets(hora3)
z
strlen(hora3)==0
f
v
printf("por favor ingrese una hora para la materia, este campo no puede estar vacio\n\n")
L


L
y
printf("Ingresa nuevamente la hora de la materia %s: \n",materiaB_3)
gets(hora3)
z
strlen(hora3)>=10
f
v
printf("Ha ingresado un numero de caracteres mayor a los permitidos\n\n")
printf("Ingresa nuevamente la hora de la materia %s: \n",materiaB_3)
gets(hora3)
e.horainicio3 = atof(hora3)
e.horainicio3<7||e.horainicio3>18
f
v
printf("Error")
printf("Ingrese una hora valida")
scanf("%f",&e.horainicio3)
system("pause")
x
system("cls")
\n\ |Ingrese la duracion de la materia |\n\ | 1-Duracion de dos horas |\n\ | 2-Duracion de cuatro horas |\n\ |
M

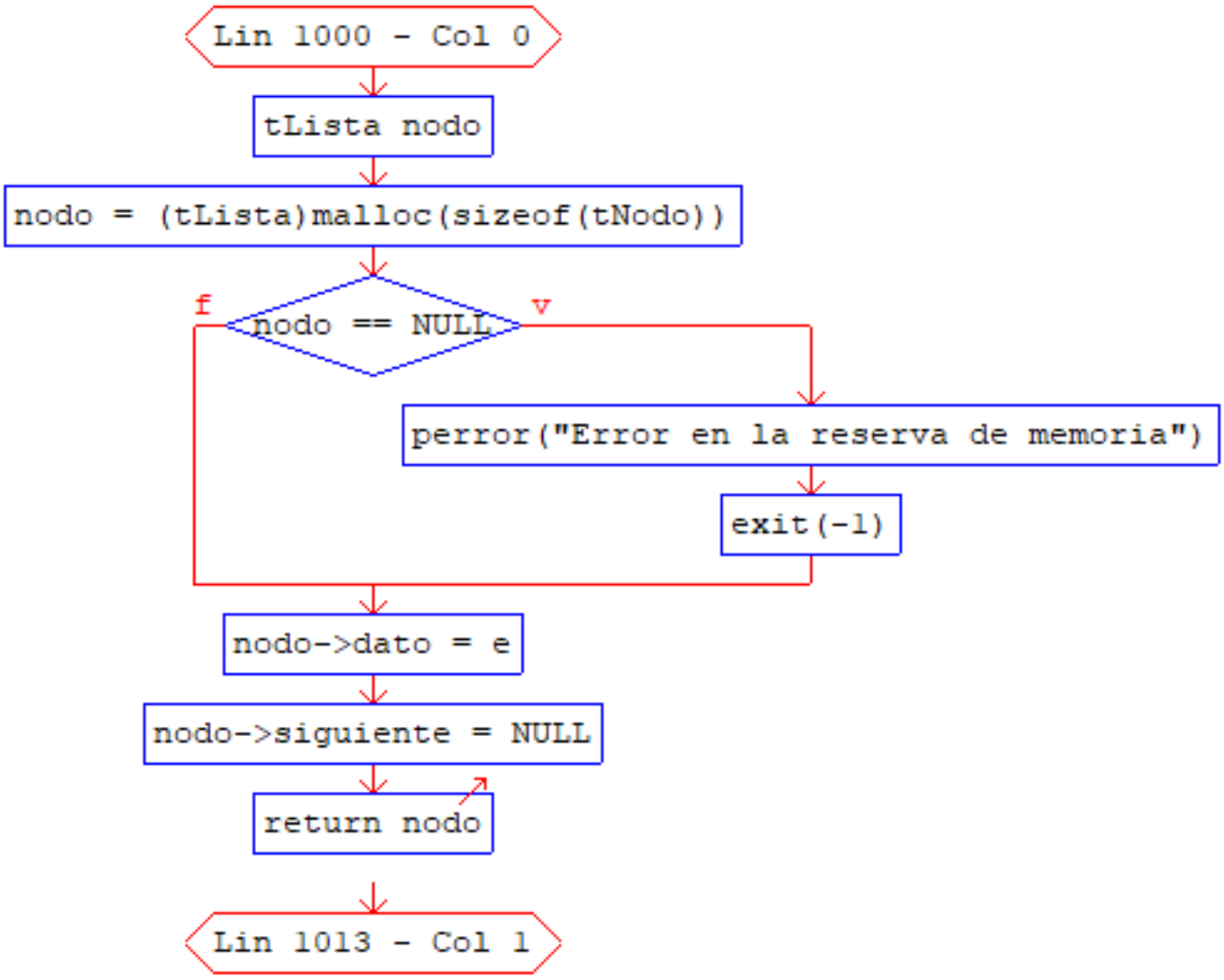

M
scanf("%d",&opc3)
switch(opc2) { case 1: { e.horafin3 = e.horainicio3+2
3
case 2: { e.horafin3 = e.horainicio3+4
3
default: { e.horafin3 = e.horainicio3+2
printf("opcion no disponible\n")
printf("Se ha seleccionado por default la clase de dos horas\n\n")
opc3 == ' '
f
3
int uno = 1
int dos = 2
int tres = 3
int cuatro = 4
guardar(e, uno)
guardar(e, dos)
guardar(e, tres)
guardar(e,cuatro)
return e
Lin 944 - Col 1
x

Función comparaElem

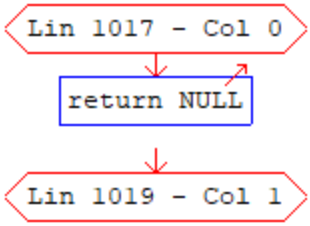


Función mostrarElem

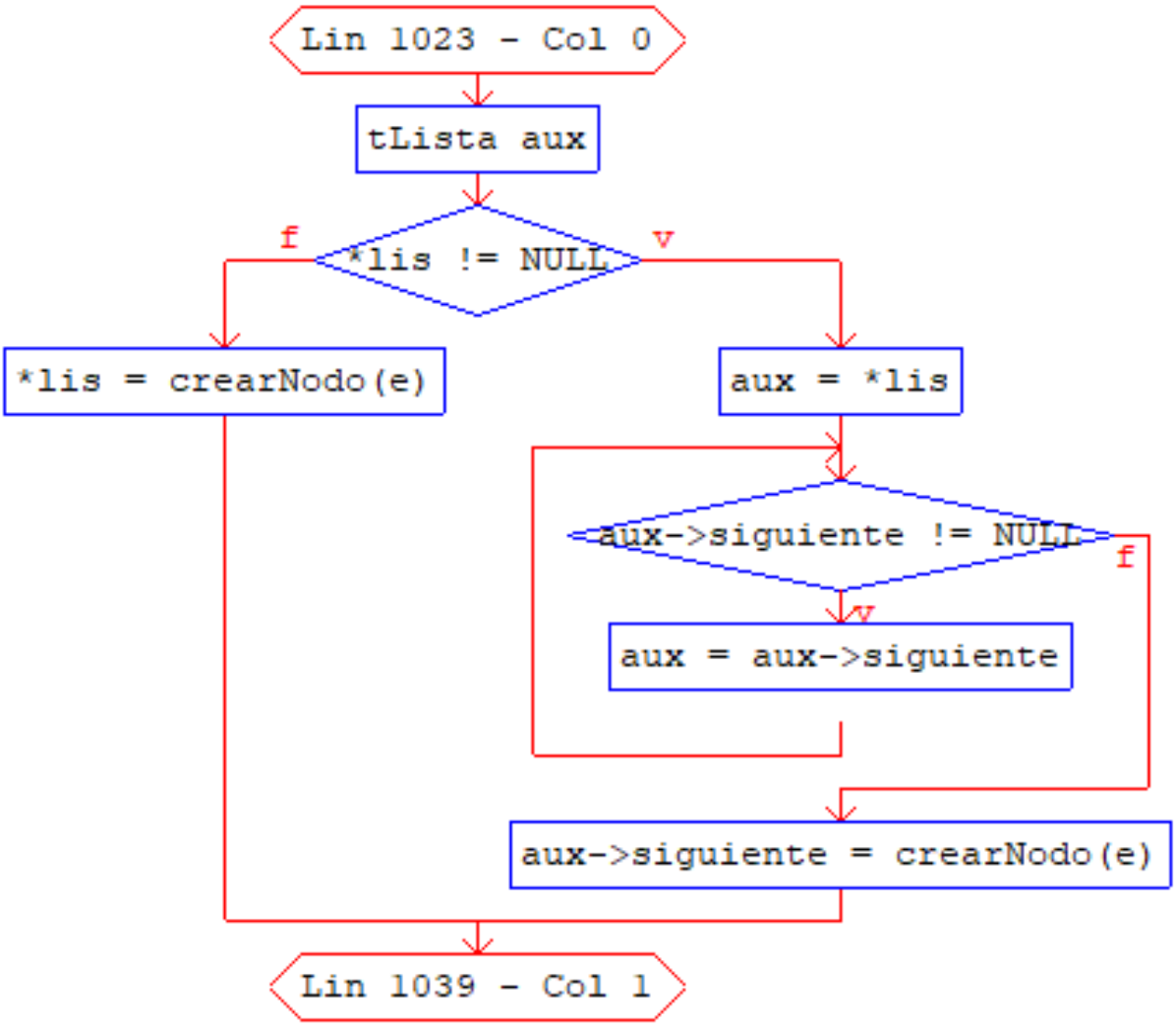
Función crearNodo



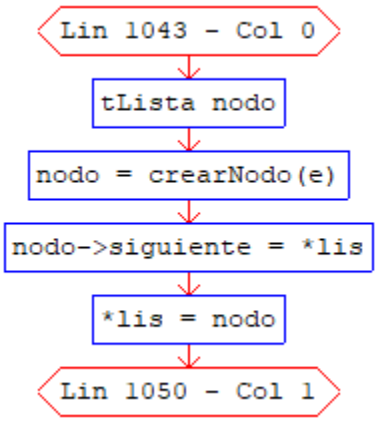
Función nuevaLista



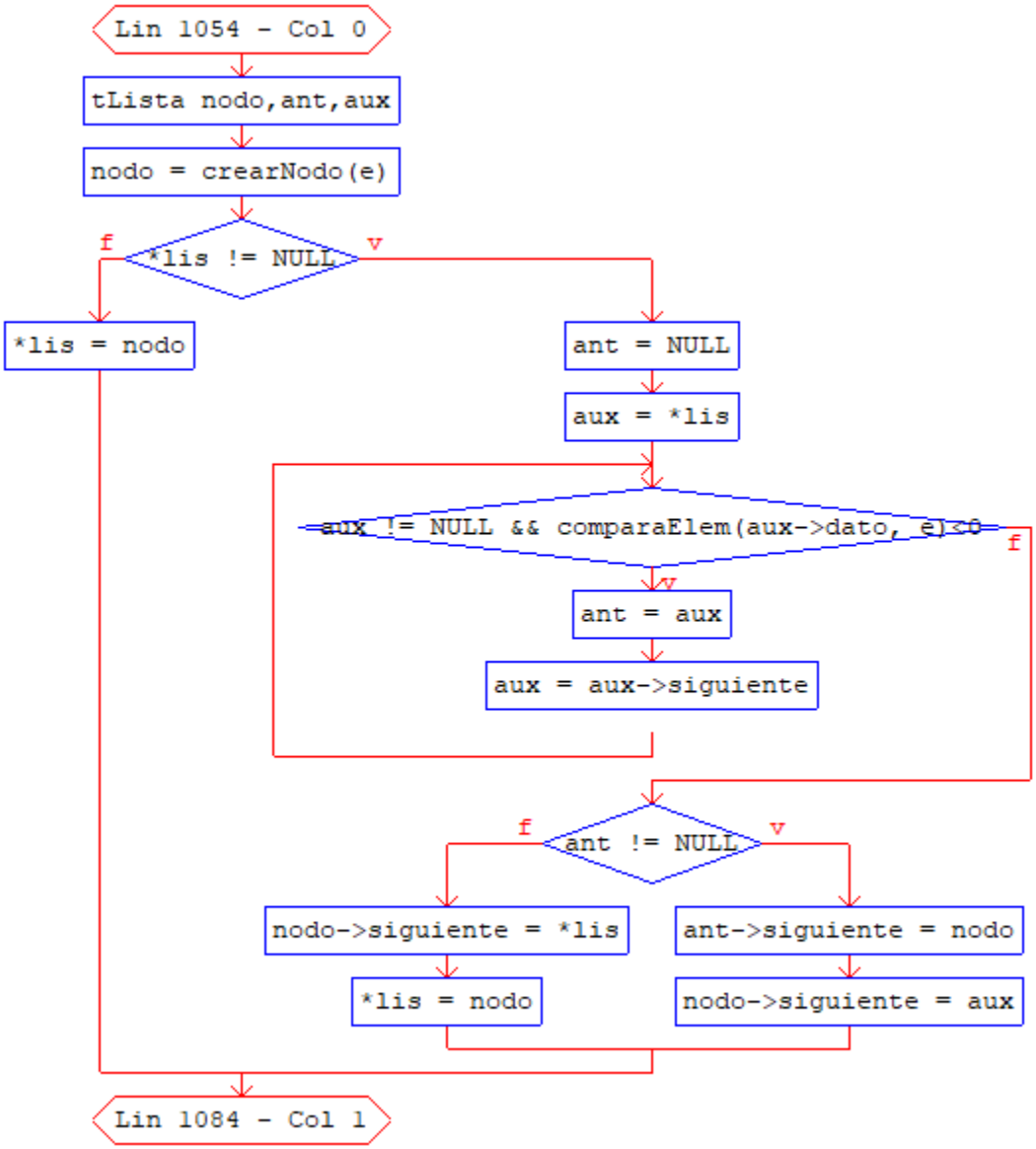
Función AgregarFinal



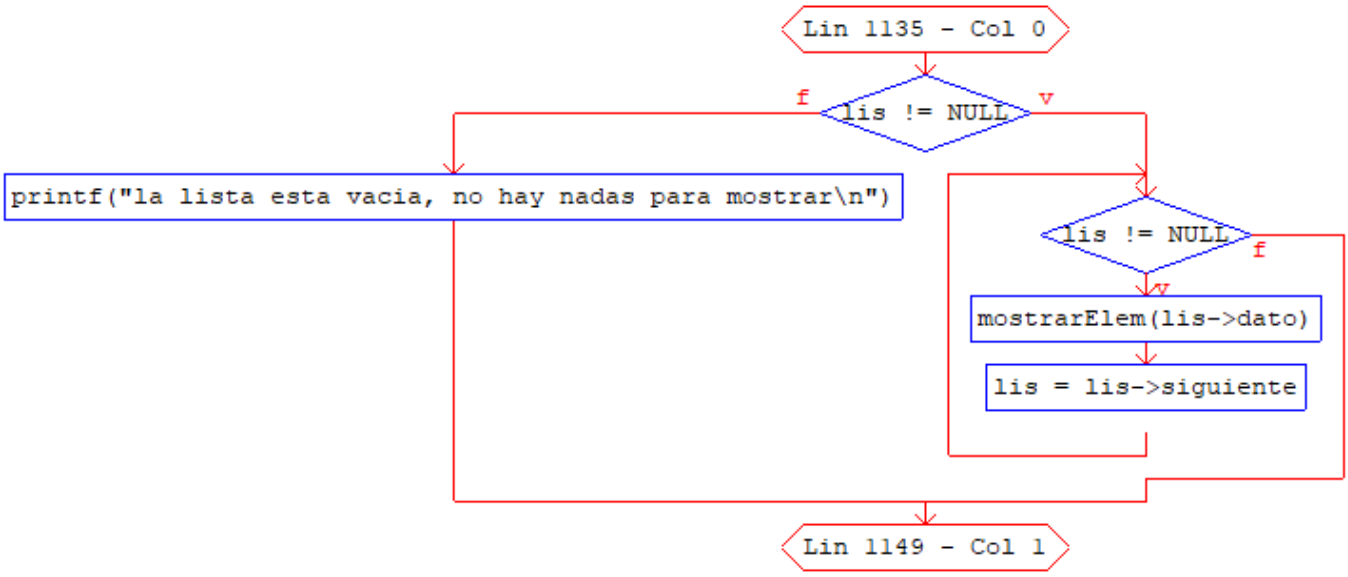
Función Agregarinicio



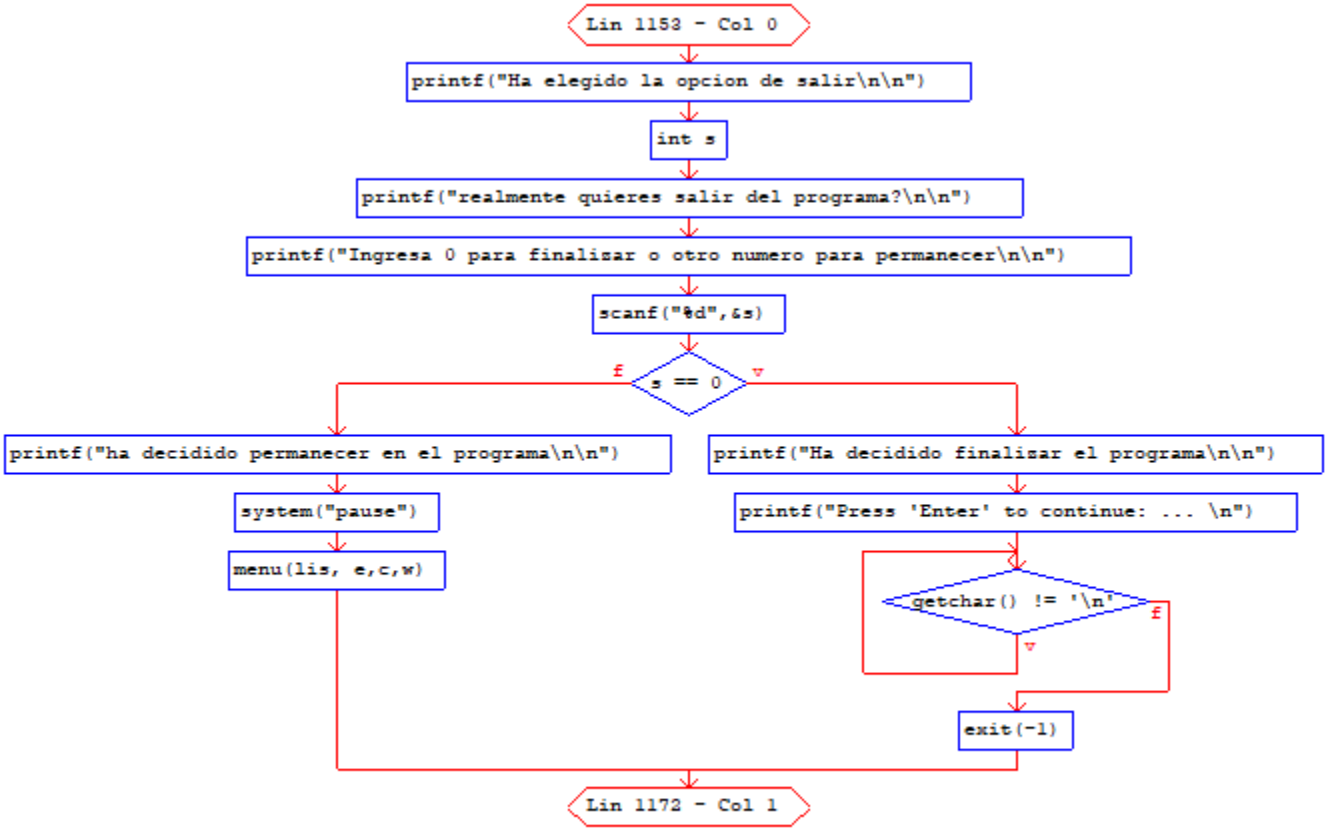
Función agregarOrdenado



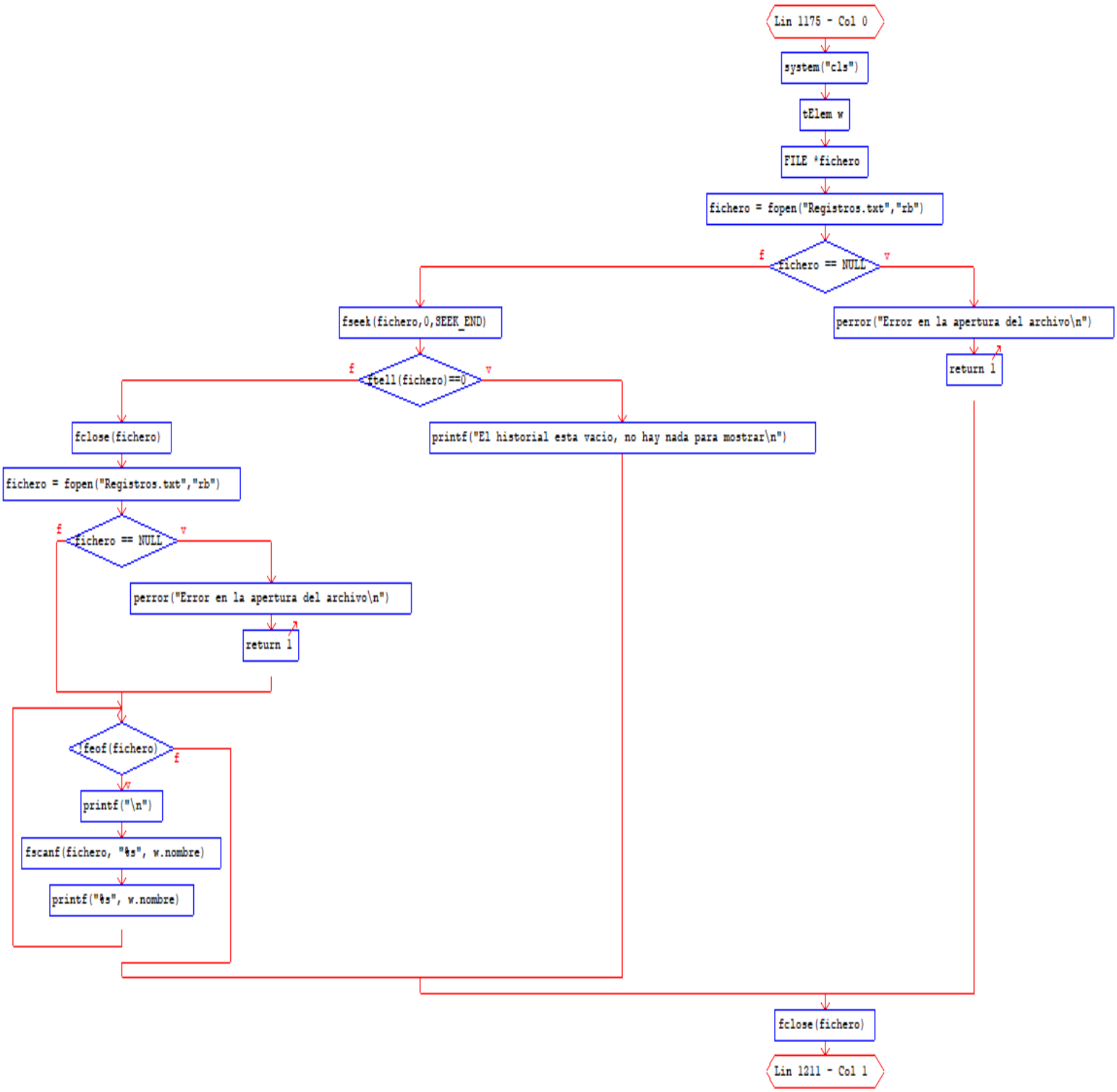
Función MostrarLista



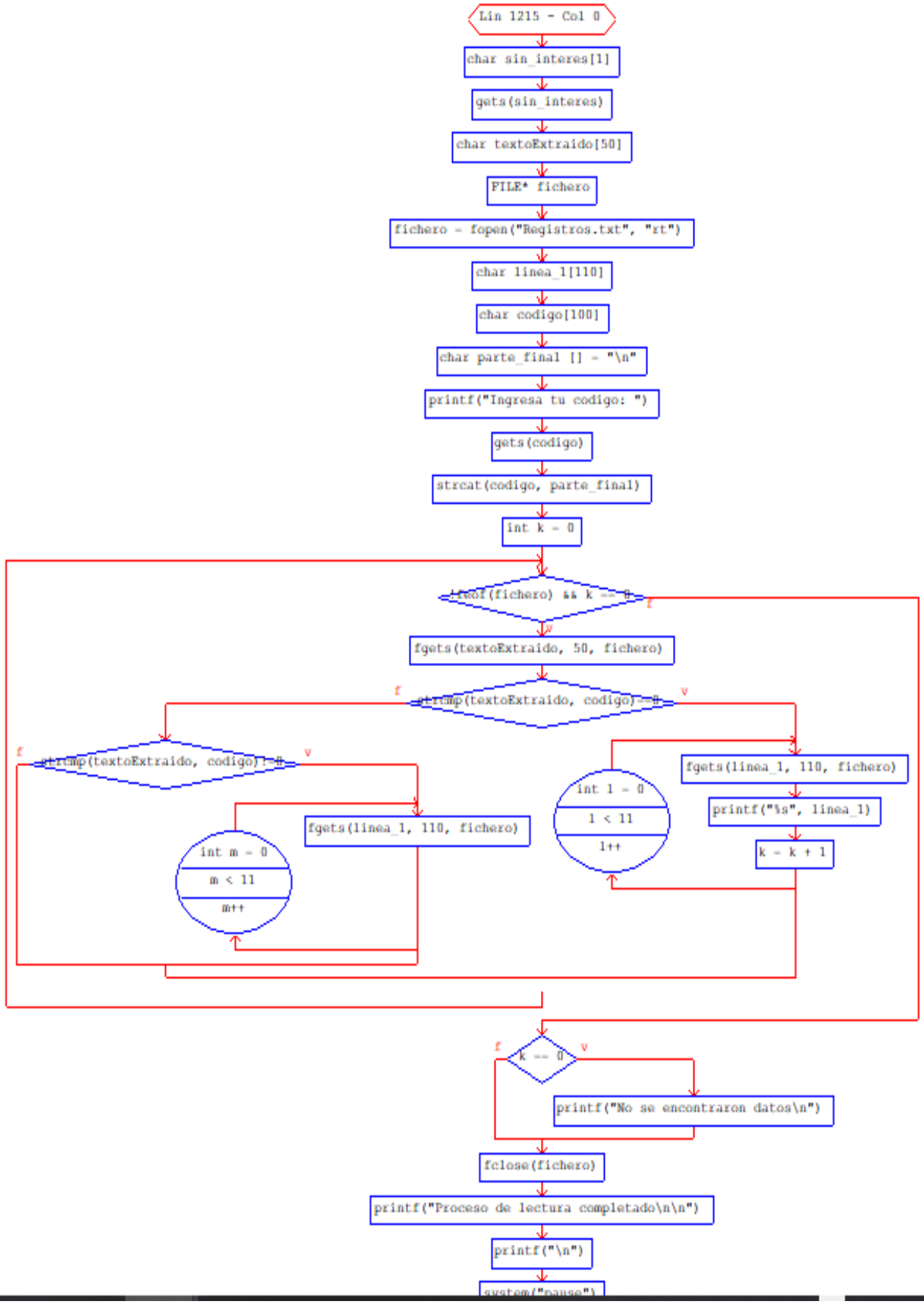
Función salir



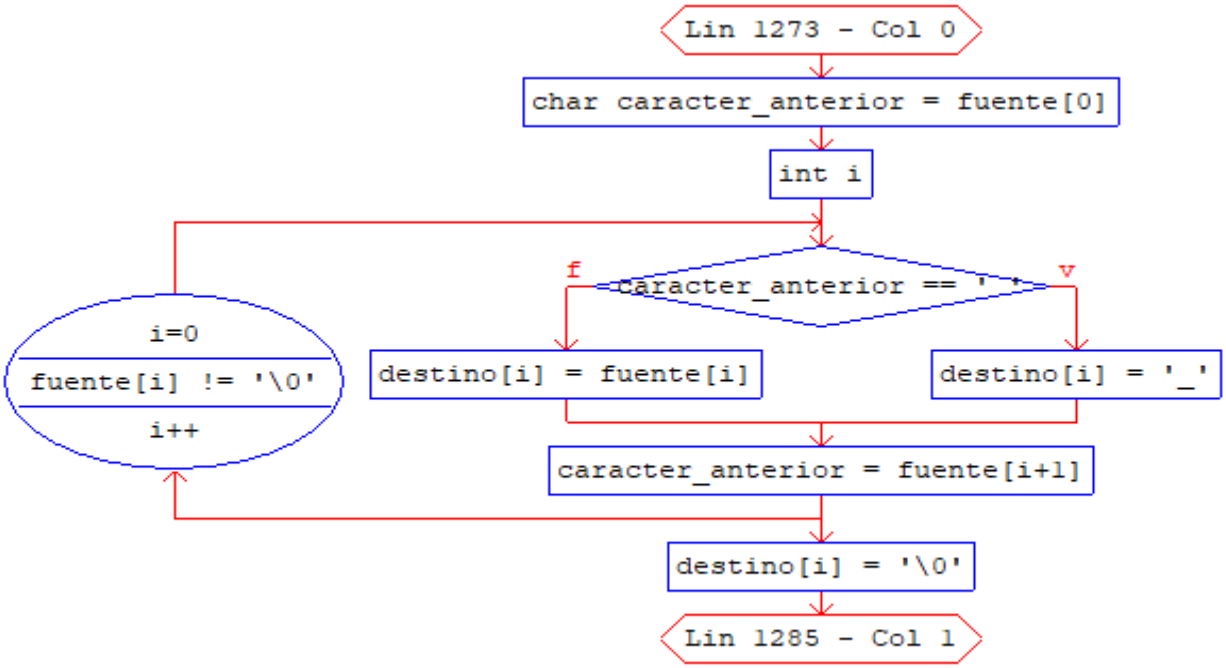
Función Historia



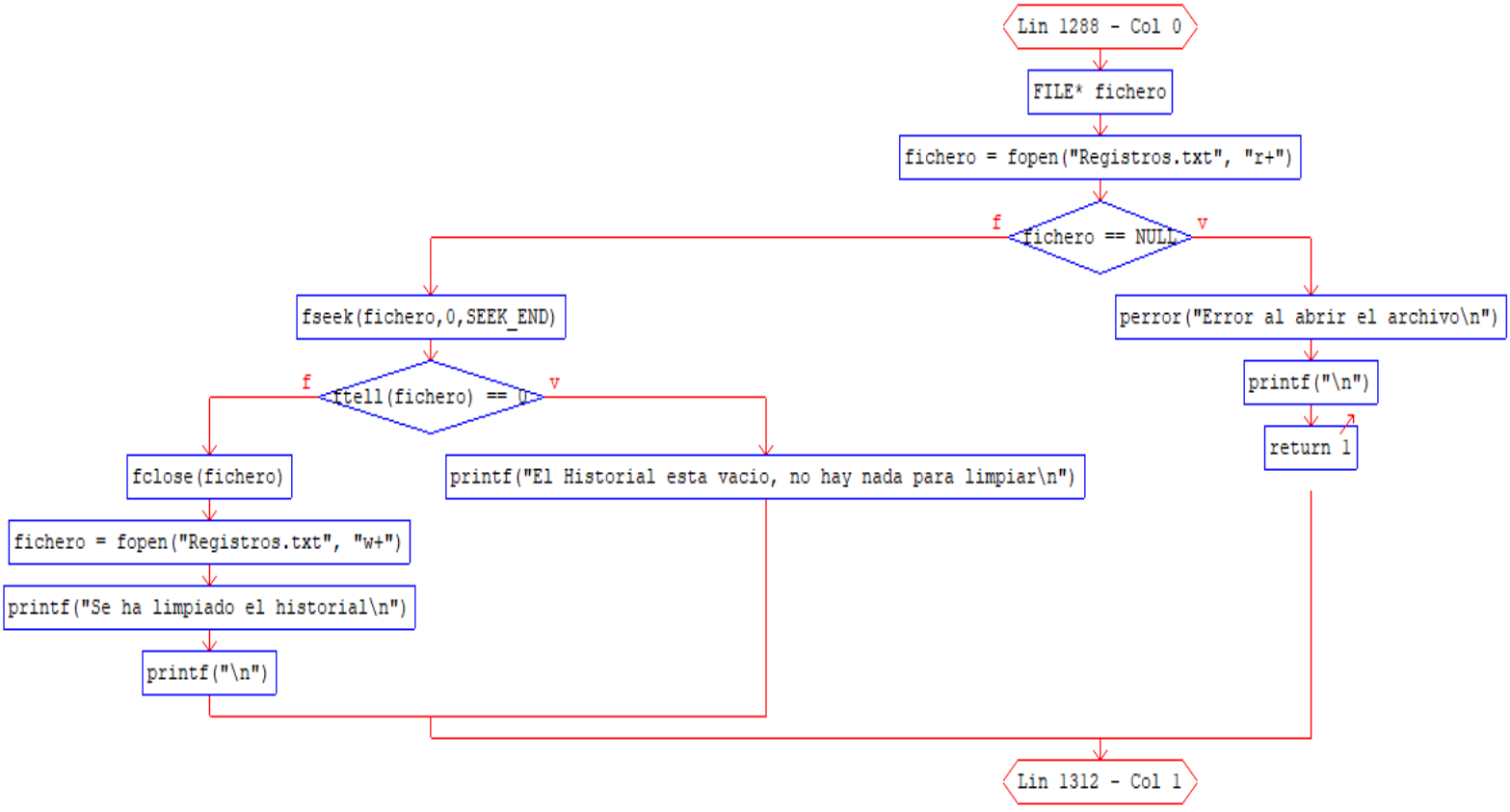
Función Buscador



Función iniciales_mays



Función limpiarHistorial



Sección 2: Pseudocódigo

```
typedef struct
Inicio
        char nombre[30];
        int edad;
        char codigo[10];
        char materia1[60],materia2[60],materia3[60],diaMateria1[30],diaMateria2[30],diaMateria3[30];
        float horaInicioo,horaInicio2,horainicio3,horaFin,horafin2,horafin3;
Final tElem;

typedef struct Nodo
Inicio
        tElem dato;
        struct Nodo* siguiente;
Final tNodo;
typedef tNodo* tLista;

tLista nuevaLista();
void menu2(tLista lis, tElem e,tElem c,tElem w);
void menu (tLista lis, tElem e,tElem c,tElem w);
tElem nuevoElem();
int comparaElem(tElem e1, tElem e2);
void agregarFinal(tLista* lis, tElem e);
void agregarInicio(tLista* lis, tElem e);
void agregarOrdenado(tLista* lis, tElem e);
void eliminarElem(tLista* lis, tElem e);
void mostrarLista(tLista lis);
void salir(tLista lis , tElem e,tElem c,tElem w);
void guardar(tElem e, int paso);
tElem nombreElim();
```

```
int historia();
void Buscador();
void iniciales_mays(char destino[], char fuente[]);
int limpiaHistorial();

int main(void)
Inicio
    system("color E0");

        tLista lis;
        tElem e;
    tElem c;
    tElem w;
        lis = nuevaLista();

    menu(lis, e,c,w);
        Retornar 0;
Final

void menu(tLista lis, tElem e, tElem c,tElem w)
Inicio
    int opc;
    Hacer
    Inicio
       Limpiar-pantalla;
        Imprimir:"\
              ------------------Menú------------------\n\
              | 1-Agregar un alumno                |\n\
              | 2-Mirar el historial               |\n\
```

```
        | 3-Buscar tu información        |\n\
        | 4-Borrar todo el historial       |\n\
        | 5-Salir                          |\n\
        ----------------------------------------\n";
Leer: opc;
Según sea(opc)
Inicio
   Caso 1:
   Inicio
     menu2(lis, e,c,w);
     Romper;
   Final
   Caso 2:
   Inicio
      Imprimir: "Mirando historial... \n";
      historia(w);
      Sistema-en-pausa;
      Romper;
   Final
   Caso 3:
      Imprimir: "Buscando tu información...\n";
      Buscador();
      Romper;
  Caso 4:
  Inicio
      Limpiar-pantalla;
      Imprimir: "Limpiando el historial...\n\n";
      limpiaHistorial();
      Sistema-en-pausa;
      Romper;
```

```
            Final
            Caso 5:
            Inicio
                Imprimir: "salir";
                salir(lis, e,c,w);
                Romper;
            Final
            Caso contrario:
            Inicio
                Imprimir: "Opción no disponible\n";
            Final
        Final
    Final Mientras (opc != ' ');
Final

void menu2(tLista lis, tElem e, tElem c,tElem w)
Inicio
          int opc;
          Hacer
    Inicio
        Limpiar-pantalla;
        Imprimir:"\
              ------------------Menú----------------------\n\
              | 1-Agregar un elemento al final       |\n\
              | 2-Agregar un elemento al inicio      |\n\
              | 3-Agregar un elemento ordenado   |\n\
              | 4-Mostrar lista                      |\n\
              | 5-Volver al menú principal           |\n\
              | 6-Salir                              |\n\
              ------------------------------------------------\n";
```

```
Imprimir: "Ingresa una opción :";
Leer: opc;

        Según sea(opc)
        Inicio
            Caso 1:
            Inicio
                Limpiar-pantalla;
                Imprimir: "Ingrese los valores a insertar: \n";
                e = nuevoElem();
                agregarFinal(&lis, e);
                Imprimir: "Valores agregado al final con éxito\n";
                Sistema-en-pausa;
                Romper;
            Final
            Caso 2:
            Inicio
                Limpiar-pantalla;
                Imprimir: "Ingrese un Elemento al inicio: \n";
                e = nuevoElem();
                agregarInicio(&lis, e);
                Imprimir: "Valores agregado al inicio con éxito\n";
                Sistema-en-pausa;
                Romper;
            Final
            Caso 3:
            Inicio
                Limpiar-pantalla;
                Imprimir: "Ingrese un elemento ordenado\n";
                e = nuevoElem();
```

```
            agregarOrdenado(&lis, e);
            Imprimir: "los valores han sido agregados en el orden correspondiente\n";
            Sistema-en-pausa;
            Romper;
        Final
        Caso 4:
        Inicio
            Limpiar-pantalla;
            Imprimir: "Mostrando la lista\n\n";
            mostrarLista(lis);
            Sistema-en-pausa;
            Romper;
        Final
        Caso 5:
        Inicio
           Limpiar-pantalla;
            Imprimir: "Volviendo al menú principal\n";
            menu(lis, e,c,w);
            Sistema-en-pausa;
            Romper;
        Final
        Caso 6:
        Inicio
            Limpiar-pantalla;
            salir(lis, e,c,w);
            Sistema-en-pausa;
            Romper;
        Final
        Caso contrario:
        Inicio
```

```
                    Limpiar-pantalla;
                    Imprimir: "Opción no valida\n");
                    Sistema-en-pausa;
                    Romper;
                Final
            Final
        Final Mientras (opc != ' ');

Final

void guardar(tElem e, int paso)
Inicio
   FILE* fichero;
   fichero = Abrir ("Registros.txt", "a+");
   Según sea(paso)
   Inicio
      Caso (1): ;

         char cadena1 [10];
         Imprimir-en-archivo: cadena1, "%s", e.codigo;
         Añadir-a-cadena (cadena1, "\n\n");
         Imprimir-en-archivo (fichero, "%s", cadena1);
         Cerrar(fichero);

         Romper;
      Caso (2): ;
         char cadenaa [] =
"|=====================================================|=================|==================|\n";
         char cadenab [] = "|_Nombre:_";
         char cadenaI [60];
```

```
        Imprimir-en-archivo: cadenaI, "%s", e.nombre);
        char cadenac [] = "|_Codigo:_";
        char cadenaI2 [10];
        Imprimir-en-archivo: cadenaI2, "%s", e.codigo);
        char cadenad [] = "|_Edad:_";
        char cadenaI3 [3];
        Imprimir-en-archivo: cadenaI3, "%d", e.edad);
        char cadenae [] = "|\n";//60,20,20
        char cadenaA [] =
"|=============================================================|==============|====================|\n";
        char cadenaB [] =
"|_____________________________Materia_____________________________|_______Horas_______|
_________Dia_________|\n";
        char cadenaC [] =
"|=============================================================|==============|====================|\n";

        int divisoresA1;
        int vecesA1;
        int productoA1;

        int A1_1 = tamaño de (cadenaI);
        int A2_1 = tamaño de (cadenaI2);
        int A3_1 = tamaño de (cadenaI3);

        Si (A1_1 < 51)
        Inicio
            divisoresA1 = 51 - (A1_1);
        Final
        Otro
        Inicio
```

```
    divisoresA1 = 0;
Final
Si ((A2_1) < 11)
Inicio
    vecesA1 = 11 - (A2_1);
Final
Otro
Inicio
    vecesA1 = 0;
Final
Si (A3_1 < 13)
Inicio
    productoA1 = 13 - (A3_1);
Final
Otro
Inicio
    productoA1 = 0;
Final

char espacioA [] = "_";

Imprimir-en-archivo: fichero, cadenaa, cadenab, cadenaI;
Para (int A=0; A < divisoresA1; A++)
Inicio
    Imprimir-en-archivo: fichero, espacioA;
Final
Imprimir-en-archivo: fichero, cadenac, cadenaI2;
Para (int B=0; B < vecesA1; B++)
Inicio
    Imprimir-en-archivo: fichero, espacioA;
```

```
    Final
    Imprimir-en-archivo: fichero, cadenad, cadenaI3;
    Para (int C=0; C < productoA1; C++)
    Inicio
       Imprimir-en-archivo: fichero, espacioA;
    Final
    Imprimir-en-archivo: fichero, cadenae;
    Imprimir-en-archivo: fichero, cadenaA, cadenaB, cadenaC;
    Cerrar (fichero);

    Romper;
Caso (3): ;
    int divisoresA;
    int vecesA;
    int productoA;

    char linea [] = "|";
    char espacio [] = "_";
    char guion [] = "_-_";
    char linea_final [] = "\n";

    char cadenaA1 [60];
    Imprimir-en-archivo: cadenaA1, e.materia1;
    char cadenaA2 [6];
    Imprimir-en-archivo: cadenaA2, e.horaInicioo;
    char cadenaA2_1 [6];
    Imprimir-en-archivo: cadenaA2_1, e.horaFin;
    char cadenaA3 [15];
    Imprimir-en-archivo: cadenaA3, e.diaMateria1;
```

```
int A1 = tamaño de (cadenaA1);
int A2 = tamaño de (cadenaA2);
int A2_f= tamaño de (cadenaA2_1);
int A2_2= A2 + A2_f;
int A3 = tamaño de (cadenaA3);

Si (A1 < 60)
Inicio
   divisoresA = 60 - (A1);
Final
Otro
Inicio
   divisoresA = 0;
Final
Si ((A2) < (17))
Inicio
   vecesA = ((17) - (A2_2));
Final
Otro
Inicio
   vecesA = 0;
Final
Si (A3 < 20)
Inicio
   productoA = 20 - (A3);
Final
Otro
Inicio
   productoA = 0;
Final
```

```
Imprimir-en-archivo: fichero, linea, cadenaA1;
Para (int A=0; A < divisoresA; A++)
Inicio
   Imprimir-en-archivo: fichero, espacio;
Final
Imprimir-en-archivo: fichero, linea, cadenaA2, guion, cadenaA2_1;
Para (int B=0; B < vecesA; B++)
Inicio
   Imprimir-en-archivo: fichero, espacio;
Final
Imprimir-en-archivo: fichero, linea, cadenaA3;
Para (int C=0; C < productoA; C++)
Inicio
   Imprimir-en-archivo: fichero, espacio;
Final
Imprimir-en-archivo: fichero, linea, linea_final;

char cadenaX1 [60];
Imprimir-en-archivo: cadenaX1, e.materia2;
char cadenaX2 [6];
Imprimir-en-archivo: cadenaX2, e.horaInicio2;
char cadenaX2_1 [6];
Imprimir-en-archivo: cadenaX2_1, e.horafin2;
char cadenaX3 [15];
Imprimir-en-archivo: cadenaX3, e.diaMateria2;

int X1 = tamaño de (cadenaX1);
int X2 = tamaño de (cadenaX2);
int X2_f= tamaño de (cadenaX2_1);
int X2_2= X2 + X2_f;
```

```
int X3 = tamaño de (cadenaX3);

Si (X1 < 60)
Inicio
   divisoresA = 60 - (X1);
Final
Otro
Inicio
   divisoresA = 0;
Final
Si ((X2) < (17))
Inicio
   vecesA = ((17) - (X2_2));
Final
Otro
Inicio
   vecesA = 0;
Final
Si (X3 < 20)
Inicio
   productoA = 20 - (X3);
Final
Otro
Inicio
   productoA = 0;
Final
Imprimir-en-archivo: fichero, linea, cadenaX1;
Para (int A=0; A < divisoresA; A++)
Inicio
   Imprimir-en-archivo: fichero, espacio;
```

```
Final
Imprimir-en-archivo: fichero, linea, cadenaX2, guion, cadenaX2_1;
Para (int B=0; B < vecesA; B++)
Inicio
   Imprimir-en-archivo: fichero, espacio;
Final
Imprimir-en-archivo: fichero, linea, cadenaX3;
Para (int C=0; C < productoA; C++)
Inicio
   Imprimir-en-archivo: fichero, espacio;
Final
Imprimir-en-archivo: fichero, linea, linea_final;

char cadenaY1 [60];
Imprimir-en-archivo: cadenaY1, e.materia3;
char cadenaY2 [6];
Imprimir-en-archivo: cadenaY2, e.horainicio3;
char cadenaY2_1 [6];
Imprimir-en-archivo: cadenaY2_1, e.horafin3;
char cadenaY3 [15];
Imprimir-en-archivo: cadenaY3, e.diaMateria3;

int Y1 = tamaño de (cadenaY1);
int Y2 = tamaño de (cadenaY2);
int Y2_f= tamaño de (cadenaY2_1);
int Y2_2= Y2 + Y2_f;
int Y3 = tamaño de (cadenaY3);

Si (Y1 < 60)
Inicio
```

```
    divisoresA = 60 - (Y1);
Final
Otro
Inicio
    divisoresA = 0;
Final
Si ((Y2) < (17))
Inicio
    vecesA = ((17) - (Y2_2));
Final
Otro
Inicio
    vecesA = 0;
Final
Si (Y3 < 20)
Inicio
    productoA = 20 - (Y3);
Final
Otro
Inicio
    productoA = 0;
Final
Imprimir-en-archivo: fichero, linea, cadenaY1;
Para (int A=0; A < divisoresA; A++)
Inicio
    Imprimir-en-archivo: fichero, espacio;
Final
Imprimir-en-archivo: fichero, linea, cadenaY2, guion, cadenaY2_1;
Para (int B=0; B < vecesA; B++)
Inicio
```

```
            Imprimir-en-archivo: fichero, espacio;
        Final
        Imprimir-en-archivo: fichero, linea, cadenaY3;
        Para (int C=0; C < productoA; C++)
        Inicio
            Imprimir-en-archivo: fichero, espacio;
        Final
        Imprimir-en-archivo: fichero, linea, linea_final;
        Cerrar(fichero);


        Romper;
    Caso (4): ;
        char inicio [] =
"|=====================================================|==========
=====|====================|\n<<<<<<<<<<<\n";


        Imprimir-en-archivo: fichero, inicio;
        Cerrar(fichero);


        Romper;
  Final
Final


tElem nuevoElem()
Inicio
        tElem e;
        fflush(stdin);


        char hora1[10];
        char hora2[10];
        char hora3[10];
```

```
char edadd[10];
int opc;
int opc2;
int opc3;

char nombreA[100];
char nombreB[100];

Imprimir: "Ingresa el nombre del alumno: \n";
Obtener (nombreB);
Mientras (tamaño de (nombreB) == 0)
Inicio
    Imprimir: "Por favor ingrese algo, el campo no puede estar vacio\n";
    Imprimir: "Ingresa nuevamente el nombre: \n";
    Obtener (nombreB);
Final
Mientras (tamaño de (nombreB) >= 99)
Inicio
    Imprimir: "Ha ingresado un nombre demasiado largo, ingresa uno más corto\n";
    Imprimir: "Ingresa nuevamente el nombre: \n";
    Obtener (nombreB);
Final

iniciales_mays(nombreA, nombreB);
Imprimir-en-archivo: e.nombre, nombreA;
Imprimir: "Ingresa el código del alumno %s: \n",e.nombre;
Obtener (e.codigo);

Mientras (tamaño de (e.codigo) == 0)
Inicio
```

```
            Imprimir: "por favor ingrese un código, este campo no puede estar vacío\n\n";
            Imprimir: "Ingresa nuevamente el código del alumno %s: \n",e.nombre;
            Obtener (e.codigo);
        Final

        Mientras (tamaño de (e.codigo) >=10 )
    Inicio
      Imprimir: "Ha ingresado un número mayor de caracteres a los permitidos\n";
      Imprimir: "Ingresa nuevamente el código del alumno %s: \n",e.nombre;
      Obtener (e.codigo);
    Final

        Imprimir: "Ingresa la edad del alumno %s: \n",e.nombre;
        Obtener (edadd);

        Mientras (tamaño de (edadd)==0)
        Inicio
            Imprimir: "Ingrese una edad para el alumno %s, este valor no puede estar
vacío\n\n",edadd;
            Imprimir: "Ingresa nuevamente la edad del alumno %s: \n",e.nombre;
            Obtener (edadd);
        Final
        Mientras (tamaño de (edadd) >= 8)
        Inicio
            Imprimir: "ha ingresado más de los caracteres permitidos, intente nuevamente\n\n";
            Imprimir: "Ingresa nuevamente la edad del alumno %s: \n",e.nombre;
            Obtener (edadd);
        Final
        e.edad = atof(edadd);

        Mientras (e.edad>100 || e.edad<0)
```

```
    Inicio
        Imprimir: "Error \n";
        Imprimir: "Ingrese una edad valida, intente nuevamente:\n");
        Leer: e.edad;
    Final
            fflush(stdin);

            Sistema-en-pausa;
            Limpiar-pantalla;

            char materiaA_1[100];
            char materiaB_1[100];



            Imprimir: "Ingresa la materia número 1:\n";
            Obtener (materiaB_1);

            Mientras (tamaño de (materiaB_1) == 0)
            Inicio
                    Imprimir: "por favor ingrese un nombre para la materia, este campo no puede estar
vacío\n\n";
                    Imprimir: "Ingresa nuevamente el nombre de la materia %s: \n",materiaB_1;
                    Obtener (materiaB_1);
            Final

            Mientras (tamaño de (materiaB_1) >= 60)
            Inicio
                    Imprimir: "Ha ingresado un numero de caracteres mayor a los permitidos\n\n";
                    Imprimir: "Ingresa nuevamente el nombre de la materia %s: \n",materiaB_1;
                    Obtener (materiaB_1);
            Final
```

```
        iniciales_mays(materiaA_1, materiaB_1);
        Imprimir-en-archivo: e.materia1, materiaA_1;

        Imprimir: "Ingresa el día de la materia %s:\n",e.materia1;
        Obtener (e.diaMateria1);

        Mientras (tamaño de (e.diaMateria1) == 0)
        Inicio
                Imprimir: "por favor ingrese un día para la materia, este campo no puede estar
vacío\n\n";
                Imprimir: "Ingresa nuevamente el dia de la materia %s: \n",materiaB_1;
                Obtener (e.diaMateria1);
        Final

        Mientras (tamaño de (e.diaMateria1) >= 10)
        Inicio
                Imprimir: "Ha ingresado un numero de caracteres mayor a los permitidos\n\n";
                Imprimir: "Ingresa nuevamente el día de la materia %s: \n",materiaB_1;
                Obtener (e.diaMateria1);
        Final

        Imprimir: "Ingresa la hora de inicio de la clase %s: \n",e.materia1;
        Obtener (hora1);

        Mientras (tamaño de (hora1) == 0)
        Inicio
                Imprimir: "por favor ingrese una hora para la materia, este campo no puede estar
vacío\n\n";
                Imprimir: "Ingresa nuevamente la hora de la materia %s: \n",materiaB_1;
                Obtener (hora1);
```

```
        Final

        Mientras (tamaño de (hora1) >= 10)
        Inicio
                Imprimir: "Ha ingresado un numero de caracteres mayor a los permitidos\n\n";
                Imprimir: "Ingresa nuevamente la hora de la materia %s: \n",materiaB_1;
                Obtener (hora1);
        Final

        e.horaInicioo = atof(hora1);

        Mientras (e.horaInicioo <7 || e.horaInicioo > 19)
    Inicio
       Imprimir: "Error \n";
       Imprimir: "Ingrese una hora valida:  ";
       Leer: e.horaInicioo;
    Final

        Sistema-en-pausa;
        Limpiar-pantalla;
        Hacer
        Inicio
                Imprimir:"\
                        |-------------------------Duración de materias-------------------------|\n\
                        |Ingrese la duración de la materia                                      |\n\
                        | 1-Duración de dos horas                                               |\n\
                        | 2-Duración de cuatro horas                                            |\n\
                        |-----------------------------------------------------------------------|\n";
                Leer: opc;
```

```
            Según sea(opc)
            Inicio
                Caso 1:
                Inicio
                    e.horaFin = e.horaInicioo+2;
                    Romper;
                Final
                Caso 2:
                Inicio
                    e.horaFin = e.horaInicioo+4;
                    Romper;
                Final
                Caso contrario:
                Inicio
                    e.horaFin = e.horaInicioo+2;
                    Imprimir: "opción no disponible\n";
                    Imprimir: "Se ha seleccionado por Caso contrario la clase de dos
horas\n\n";
                Final
            Final

    Final Mientras (opc == ' ');

    Sistema-en-pausa;
    Limpiar-pantalla;

    fflush(stdin);

    char materiaA_2[100];
    char materiaB_2[100];
    Imprimir: "Ingresa la materia número 2:\n";
```

```
        Obtener (materiaB_2);

        Mientras (tamaño de (materiaB_2) == 0)
        Inicio
                Imprimir: "por favor ingrese un nombre para la materia, este campo no puede estar
vacío\n\n";
                Imprimir: "Ingresa nuevamente el nombre de la materia %s: \n",materiaB_2;
                Obtener (materiaB_2);
        Final

        Mientras (tamaño de (materiaB_2) >= 60)
        Inicio
                Imprimir: "Ha ingresado un numero de caracteres mayor a los permitidos\n";
                Imprimir: "Ingresa nuevamente el nombre de la materia %s: \n",materiaB_2;
                Obtener (materiaB_2);
        Final
        iniciales_mays(materiaA_2, materiaB_2);
        Imprimir-en-archivo: e.materia2, materiaA_2);

        Imprimir: "Ingresa el día de la materia %s:\n",e.materia2;
        Obtener (e.diaMateria2);

        Mientras (tamaño de (e.diaMateria2) == 0)
        Inicio
                Imprimir: "por favor ingrese un día para la materia, este campo no puede estar
vacío\n\n";
                Imprimir: "Ingresa nuevamente el día de la materia %s: \n",materiaB_2;
                Obtener (e.diaMateria2);
        Final

        Mientras (tamaño de (e.diaMateria2) >= 10)
```

```
        Inicio
                Imprimir: "Ha ingresado un numero de caracteres mayor a los permitidos\n\n";
                Imprimir: "Ingresa nuevamente el día de la materia %s: \n",materiaB_2;
                Obtener (e.diaMateria2);
        Final

        Imprimir: "Ingresa la hora de inicio de la clase %s: \n",e.materia2;
        Obtener (hora2);

        Mientras (tamaño de (hora2) == 0)
        Inicio
                Imprimir: "por favor ingrese una hora para la materia, este campo no puede estar
vacío\n\n";
                Imprimir: "Ingresa nuevamente la hora de la materia %s: \n",materiaB_2;
                Obtener (hora2);
        Final

        Mientras (tamaño de (hora2) >= 10)
        Inicio
                Imprimir: "Ha ingresado un numero de caracteres mayor a los permitidos\n\n";
                Imprimir: "Ingresa nuevamente la hora de la materia %s: \n",materiaB_2;
                Obtener (hora2);
        Final

        e.horaInicio2 = atof(hora2);

        Mientras (e.horaInicio2<7||e.horaInicio2>19)
    Inicio
       Imprimir: "Error";
       Imprimir: "Ingrese una hora valida";
       Leer: e.horaInicio2);
```

```
Final

    Sistema-en-pausa;
    Limpiar-pantalla;

    Hacer
    Inicio
        Imprimir:"\
            |-------------------------Duración de materias-------------------------|\n\
            |Ingrese la duración de la materia                                     |\n\
            | 1-Duración de dos horas                                              |\n\
            | 2-Duración de cuatro horas                                           |\n\
            |----------------------------------------------------------------------|\n";
        Leer: opc2;

        Según sea(opc2)
        Inicio
            Caso 1:
            Inicio
                e.horafin2 = e.horaInicio2+2;
                Romper;
            Final
            Caso 2:
            Inicio
                e.horafin2 = e.horaInicio2+4;
                Romper;
            Final
            Caso contrario:
            Inicio
                e.horafin2 = e.horaInicio2+2;
```

```
                    Imprimir: "opción no disponible\n";
                    Imprimir: "Se ha seleccionado por Caso contrario la clase de dos horas\n\n";
                Final
            Final

    Final Mientras (opc2 == ' ');

    Sistema-en-pausa;
    Limpiar-pantalla;

    fflush(stdin);
    char materiaA_3[100];
    char materiaB_3[100];
    Imprimir: "Ingresa la materia numero 3:\n";
    Obtener (materiaB_3);

    Mientras (tamaño de (materiaB_3) == 0)
    Inicio
        Imprimir: "por favor ingrese un nombre para la materia, este campo no puede estar vacío\n\n";
        Imprimir: "Ingresa nuevamente el nombre de la materia %s: \n",materiaB_3;
        Obtener (materiaB_3);
    Final

    Mientras (tamaño de (materiaB_3) >= 60)
    Inicio
        Imprimir: "Ha ingresado un numero de caracteres mayor a los permitidos\n";
        Imprimir: "Ingresa nuevamente el nombre de la materia %s: \n",materiaB_3;
        Obtener (materiaB_3);
    Final
```

```
        iniciales_mays(materiaA_3, materiaB_3);
        Imprimir-en-archivo: e.materia3, materiaA_3;

        Imprimir: "Ingresa el día de la materia %s:\n",e.materia3;
        Obtener (e.diaMateria3);

        Mientras (tamaño de (e.diaMateria3) == 0)
        Inicio
                Imprimir: "por favor ingrese un día para la materia, este campo no puede estar
vacío\n\n";
                Imprimir: "Ingresa nuevamente el día de la materia %s: \n",materiaB_3;
                Obtener (e.diaMateria3);
        Final

        Mientras (tamaño de (e.diaMateria3)>=10)
        Inicio
                Imprimir: "Ha ingresado un numero de caracteres mayor a los permitidos\n\n";
                Imprimir: "Ingresa nuevamente el día de la materia %s: \n",materiaB_3;
                Obtener (e.diaMateria3);
        Final


        Imprimir: "Ingresa la hora de inicio de la clase %s: \n", e.materia3;
        Obtener (hora3);

        Mientras (tamaño de (hora3) == 0)
        Inicio
                Imprimir: "por favor ingrese una hora para la materia, este campo no puede estar
vacío\n\n";
                Imprimir: "Ingresa nuevamente la hora de la materia %s: \n",materiaB_3;
```

```
            Obtener(hora3);
        Final

        Mientras (tamaño de (hora3) >= 10)
        Inicio
            Imprimir: "Ha ingresado un numero de caracteres mayor a los permitidos\n\n";
            Imprimir: "Ingresa nuevamente la hora de la materia %s: \n",materiaB_3;
            Obtener(hora3);
        Final

        e.horainicio3 = atof(hora3);

        Mientras (e.horainicio3<7||e.horainicio3>19)
    Inicio
       Imprimir: "Error";
       Imprimir: "Ingrese una hora valida";
       Leer: e.horainicio3;
    Final

        Sistema-en-pausa;
        Limpiar-pantalla;

        Hacer
        Inicio
            Imprimir:"\
                |-------------------------Duración de materias-------------------------|\n\
                |Ingrese la duración de la materia                                     |\n\
                | 1-Duración de dos horas                                              |\n\
                | 2-Duración de cuatro horas                                           |\n\
                |----------------------------------------------------------------------|\n";
```

```
            Leer: opc3;

            Según sea(opc2)
            Inicio
                  Caso 1:
                  Inicio
                        e.horafin3 = e.horainicio3+2;
                        Romper;
                  Final
                  Caso 2:
                  Inicio
                        e.horafin3 = e.horainicio3+4;
                        Romper;
                  Final
                  Caso contrario:
                  Inicio
                        e.horafin3 = e.horainicio3+2;
                        Imprimir: "opción no disponible\n";
                        Imprimir: "Se ha seleccionado por Caso contrario la clase de dos
horas\n\n";
                  Final
            Final

      Final Mientras (opc3 == ' ');

   int uno = 1;
   int dos = 2;
   int tres = 3;
   int cuatro = 4;

   guardar(e, uno);
```

```
    guardar(e, dos);
    guardar(e, tres);
    guardar(e,cuatro);
    Retornar e;
Final

int comparaElem(tElem e1, tElem e2)
Inicio
        int d1,d2;
        Si (d1<d2)
        Inicio
                Retornar -1;
        Final
        Otro
        Inicio
                Si (d1>d2)
                Inicio
                        Retornar 1;
                Final
                Otro
                Inicio
                        Retornar 0;
                Final
        Final
Final

void mostrarElem(tElem e)
Inicio
        Imprimir: "|----------------------------------------------------------------------------------------|\n";
        Imprimir: "|Nombre del alumno: %s\n",e.nombre;
```

```
        Imprimir: "|Código del alumno: %s\n",e.codigo;
        Imprimir: "|Edad del alumno: %d\n",e.edad;

        Imprimir: "|-----------------------------------------------------------------------------------------------|\n";
        Imprimir: "|Materia numero 1: %s\n",e.materia1;
        Imprimir: "|Día de la materia %s: %s\n",e.materia1,e.diaMateria1;
        Imprimir: "|Materia 1 Inicio de clases: %.1f       |         ",e.horaInicioo;
        Imprimir: "|Materia 1 fin de clases: %.1f\n",e.horaFin;

        Imprimir: "|-----------------------------------------------------------------------------------------------|\n";
        Imprimir: "|Materia numero 2: %s\n",e.materia2;
        Imprimir: "|Día de la materia %s: %s\n",e.materia2,e.diaMateria2;
        Imprimir: "|Materia 2 Inicio de clases: %.1f       |         ",e.horaInicio2;
        Imprimir: "|Materia 2 fin de clases: %.1f\n",e.horafin2;

        Imprimir: "|-----------------------------------------------------------------------------------------------|\n";
        Imprimir: "|Materia numero 3: %s\n",e.materia3;
        Imprimir: "|Día de la materia %s: %s\n",e.materia3,e.diaMateria3;
        Imprimir: "|Materia 3 Inicio de clases: %.1f       |         ",e.horainicio3;
        Imprimir: "|Materia 3 fin de clases: %.1f \n\n\n",e.horafin3;
        Imprimir: "|-----------------------------------------------------------------------------------------------|\n";
        Imprimir: "\n";
Final

tLista crearNodo(tElem e)
Inicio
        tLista nodo;
        nodo = (tLista)malloc(sizeof(tNodo));
        Si (nodo == NULL)
        Inicio
```

```
            Perror "Error en la reserva de memoria";
            exit(-1);
        Final

        nodo->dato = e;
        nodo->siguiente = NULL;

        Retornar nodo;
Final

tLista nuevaLista()
Inicio
        Retornar NULL;
Final

void agregarFinal(tLista* lis, tElem e)
Inicio
        tLista aux;
        Si (*lis != NULL)
        Inicio
            aux = *lis;
            Mientras (aux->siguiente != NULL)
            Inicio
                aux = aux->siguiente;
            Final
            aux->siguiente = crearNodo(e);
        Final
        Otro
        Inicio
            *lis = crearNodo(e);
```

```
        Final
Final

void agregarInicio(tLista* lis, tElem e)
Inicio
        tLista nodo;

        nodo = crearNodo(e);

        nodo->siguiente = *lis;
        *lis = nodo;
Final

void agregarOrdenado(tLista* lis, tElem e)
Inicio
        tLista nodo,ant,aux;
        nodo = crearNodo(e);

        Si (*lis != NULL)
        Inicio
                ant = NULL;
                aux = *lis;

                Mientras (aux != NULL && comparaElem(aux->dato, e)<0)
                Inicio
                        ant = aux;
                        aux = aux->siguiente;
                Final
                Si (ant != NULL)
                Inicio
```

```
                    ant->siguiente = nodo;
                    nodo->siguiente = aux;
                Final
                Otro
                Inicio
                    nodo->siguiente = *lis;
                    *lis = nodo;
                Final

        Final
        Otro
        Inicio
            *lis = nodo;
        Final
Final
tElem nombreElim()
Inicio
   tElem c;
   fflush(stdin);
   Imprimir: "Ingresa el nombre del alumno a eliminar:\n";
   Obtener(c.nombre);

   Retornar c;
Final

void eliminarElem(tLista* lis, tElem e)
Inicio
        tLista ant, aux;

        Si (*lis != NULL)
```

```
Inicio
    ant = NULL;
    aux = *lis;
    Mientras (aux != NULL && comparaElem(aux->dato, e) != 0)
    Inicio
        ant = aux;
        aux = aux->siguiente;
        Imprimir: "Se ha eliminado el registro con exito!\n";
    Final
    Si (aux != NULL)
    Inicio
        Si (ant != NULL)
        Inicio
            ant->siguiente = aux->siguiente;
            Imprimir: "Se ha eliminado el registro con éxito!\n";
        Final
        Otro
        Inicio
            *lis = aux->siguiente;
        Final
    Final
    Otro
    Inicio
        Imprimir: "No se ha encontrado el valor para eliminar\n";
    Final
Final
Otro
Inicio
    Imprimir: "La lista está vacía, no hay ningún elemento para eliminar\n";
Final
```

```
Final

void mostrarLista(tLista lis)
Inicio
	Si (lis != NULL)
	Inicio
		Mientras (lis != NULL)
		Inicio
			mostrarElem(lis->dato);
			lis = lis->siguiente;
		Final
	Final
	Otro
	Inicio
		Imprimir: "la lista esta vacía, no hay nadas para mostrar\n";
	Final
Final

void salir(tLista lis, tElem e,tElem c,tElem w)
Inicio
	Imprimir: "Ha elegido la opción de salir\n\n");
	int s;
	Imprimir: "realmente quieres salir del programa?\n\n";
	Imprimir: "Ingresa 0 para finalizar u otro número para permanecer\n\n";
	Leer: s;
	Si (s == 0)
	Inicio
		Imprimir: "Ha decidido finalizar el programa\n\n";
		Imprimir: "Press 'Enter' to continue: ... \n";
		Mientras (getchar() != '\n');
```

```
                exit(-1);
        Final
        Otro
        Inicio
                Imprimir: "ha decidido permanecer en el programa\n\n";
                Sistema-en-pausa;
                menu(lis, e,c,w);
        Final
Final

int historia()
Inicio
    Limpiar-pantalla;
    tElem w;
    FILE *fichero;
    fichero = Abrir ("Registros.txt","rb");
    Si (fichero == NULL)
    Inicio
        Perror "Error en la apertura del archivo\n";
        Retornar 1;
    Final
    Otro
    Inicio
                fseek(fichero,0,SEEK_END);
                Si (ftell(fichero)==0)
                Inicio
                        Imprimir: "El historial esta vacío, no hay nada para mostrar\n";
                Final
                Otro
                Inicio
```

```
                    Cerrar (fichero);
                    fichero = Abrir ("Registros.txt","rb");
                    Si (fichero == NULL)
                    Inicio
                        Perror "Error en la apertura del archivo\n";
                        Retornar 1;
                    Final
                    Mientras (!feof(fichero))
                    Inicio
                        Imprimir: "\n";
                        leer-del-archivo :fichero, "%s", w.nombre;
                        Imprimir: "%s", w.nombre;
                    Final
        Final
    Final
    Cerrar(fichero);
Final

void Buscador()
Inicio
    char sin_interes[1];
    Obtener (sin_interes);

    char textoExtraido[50];
    FILE* fichero;

    fichero = Abrir ("Registros.txt", "rt");
    char linea_1[110];

    char codigo[100];
```

```
char parte_final [] = "\n";
Imprimir: "Ingresa tu código: ";
Obtener (codigo);

Añadir-a-cadena (codigo, parte_final);

int k = 0;
Mientras (!feof(fichero) && k == 0)
Inicio
    leer-del-archivo (textoExtraido, 50, fichero);
    Si (comparación-de (textoExtraido, codigo)==0)
    Inicio
        Para (int l = 0; l < 11; l++)
        Inicio
            leer-del-archivo(linea_1, 110, fichero);
            Imprimir: "%s", linea_1;

            k = k + 1;
        Final
    Final
    Otro Si (comparación-de (textoExtraido, codigo) != 0)
    Inicio
        Para (int m = 0; m < 11; m++)
        Inicio
            leer-del-archivo (linea_1, 110, fichero);
        Final
    Final
Final
Si (k == 0)
Inicio
```

```
        Imprimir: "No se encontraron datos\n";
    Final

    Cerrar (fichero);
    Imprimir: "Proceso de lectura completado\n\n";
          Imprimir:"\n";

    Sistema-en-pausa;
Final

void iniciales_mays(char destino[], char fuente[])
Inicio
    char caracter_anterior = fuente[0];
    int i;
    Para (i=0; fuente[i] != '\0'; i++)
        Inicio
        Si (caracter_anterior == ' ')
            destino[i] = '_';
        Otro
            destino[i] = fuente[i];
        caracter_anterior = fuente[i+1];
    Final
    destino[i] = '\0';
Final

int limpiaHistorial()
Inicio
          FILE* fichero;
          fichero = Abrir ("Registros.txt", "r+");
          Si (fichero == NULL)
```

```
        Inicio
                Perror "Error al abrir el archivo\n";
                Imprimir: "\n";
                Retornar 1;
        Final
        Otro
        Inicio
                fseek(fichero,0,SEEK_END);
                Si (ftell(fichero) == 0)
                Inicio
                        Imprimir: "El Historial está vacío, no hay nada para limpiar\n";
                Final
                Otro
                Inicio
                        Cerrar (fichero);
                        fichero = Abrir ("Registros.txt", "w+");
                        Imprimir: "Se ha limpiado el historial\n";
                        Imprimir: "\n";
                Final
        Final
Final
```

Sección 3: Estructuras usadas