Programación Aplicada y Lab.

PRÁCTICA NO.1 PROCESAMIENTO DIGITAL DE IMÁGENES PGM

César Mauricio Arellano Velásquez

Este documento presenta un programa, que a través del uso de arreglos bidimensionales, punteros, asignación dinámica de memoria y manejo de archivos binarios, poder procesar digitalmente imágenes en formato PGM (Pixel Gray Map).

El programa consta de 4 herramientas:

1.- Negativo de la imagen.

2.- Ecualizar linealmente la imagen.

3.- Reducir la imagen.

4.- Ampliar la imagen.

1. INTRODUCCIÓN

**E**n esta práctica un programa capaz de procesar, editar y guardar una imagen PGM. Para comenzar la resolución del problema, se segmentó requerimiento por requerimiento para poder dar paso al estudio y asignación de las funciones a ocupar en este proceso, se buscó información vital que permitiese entender cómo es que se manipulan las imágenes de PGM, la asignación de memoria de arreglos bidimensionales, reforzar el tema que se refiere a los bytes que ocupa cada tipo de variable, ya sea int, char y sus derivados, “unsigned char” es un ejemplo de ello, al igual de repasar el tema de cómo obtener la ecuación de la recta-pendiente, para el caso de la ecualización lineal.

1. ANÁLISIS

Tras haber investigado previamente los conocimientos enfocados al problema y retomando sobretodo los temas vistos en clase, se dió la pauta para solucionar el mismo. Inicié dividiendo de nuevo las tareas que debía realizar el programa, a través de funciones pero ahora de la manera más detallada posible, para que las instrucciones que el programa debe seguir puedan lograr la resolución de la manera más óptima posible.

*Función Principal*

En la función principal como tal se encuentran las llamadas a las funciones correspondientes (estas se mencionan a lo largo del documento) y el menú de opciones, el usuario determina qué herramienta utilizar de las cuatro disponibles que le puede aplicar a la imagen, en esta misma se asignan de manera dinámica a las matrices sus dimensiones.

1. *Creditos.*

En la función, no se le declaran parámetros de entrada, ni de salida, ya que ésta sólo presenta los créditos del programa, es decir, el nombre del desarrollador, y el objetivo general del programa, junto con algunas recomendaciones del uso.

1. *Leer\_Imagen.*

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Procesos** | **Salida** |
|  | Leer Nombre\_Archivo  Abre el archivo  Verifica que exista  Guarda el encabezado en una estructura.  Verifica si el archivo es formato PGM  Cierra el archivo. | Encabezado  (Valor1, Valor 2, X, Y, Max)  [Tipo Estructura].  Lectura\_Arch  [Tipo String] |

En esta función se le pedirá al usuario el nombre de la imagen a leer, se abrirá el archivo en modo lectura binaria, después se validará si el archivo existe o no, si existe el archivo, este procederá a escanear el encabezado del archivo y guardar cada valor en los campos correspondientes de la estructura declarada.

Cuando guarda los valores, este verifica si el archivo está en formato PGM, si efectivamente es PGM continúa a cerrar el archivo (por cada verificación no valida el programa muestra un mensaje de error y se sale de la ejecución).

1. *Guardar\_en\_Matriz.*

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Procesos** | **Salida** |
| Encabezado  [Tipo Estructura]  Lectura\_Arch  [Tipo String] | Abrir archivo en modo lectura binaria.  Moverse 15 bytes en el archivo  Leer binariamente los datos  Asignar cada valor a la matriz.  Cerrar el archivo. | Dimensiones  [Tipo int Bidimensional] |

*Previamente en la función principal, se asignó dinámicamente memoria específica a la matriz “Dimensiones”.*

Esta función lo que hace es abrir de nuevo el archivo en modo lectura y se recorre al byte 15, es decir, donde termina el encabezado de las imágenes PGM, después se recorre con dos ciclos for tanto en el eje de las Y como en el de las X para almacenar en un temporal cada de uno de los valores contenidos en la imagen (estos se analizan con el tamaño de bytes de unsigned char y se guardan como enteros) con la función fread, para después ser transformados a unsigned char y poder ser almacenados a la matriz con valores que van del rango [0,255].

Tras completarse los ciclos, se cierra el archivo.

1. *Negativo.*

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Procesos** | **Salida** |
| Encabezado  [Tipo Estructura] | Recorrer matriz  Asignar los nuevos valores a la matriz | Dimensiones  [Tipo int Bidimensional] |

Con dos ciclos for se va recorriendo la matriz para aplicar la fórmula de invertir los valores para cada pixel,, logrando así obtener el negativo de la imagen.

y=x\*(-1)+255;

x=pixel original.

y=nuevo pixel.

1. *Ecualizar.*

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Procesos** | **Salida** |
| Encabezado  [Tipo Estructura] | Declarar Menor, Mayor.  Asignar Valores a las mismas.  Recorrer todo el arreglo y encontrar el valor más pequeño y el más grande.  Obtener pendiente y ordenada al origen.  Recorrer el arreglo y aplicar la ecuación de la recta a cada pixel de la matriz. | Dimensiones  [Tipo int Bidimensional] |

Se asignan variables menor (valores previos =255) y mayor(valor previos=0)

Con dos ciclos for y dos if, se va recorriendo la matriz para saber cual es el pixel más alto y el más bajo contenido en la matriz.

Después de obtener los dos valores, se comienza a realizar las operaciones para obtener la pendiente y luego la ordenada al origen, para así aplicar a cada pixel el emparejamiento lineal a la imagen.

m=(255-0)/(mayor-menor).

b = (-1)\*(m \* Menor);

y=mx+b;

x= pixel original.

y=nuevo pixel.

1. *Reducir.*

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Procesos** | **Salida** |
| Encabezado  [Tipo Estructura] | Regresar a dimensiones originales  Recorrer Arreglo de manera par y asignar valores a la nueva matriz. | Dimensiones  [Tipo int Bidimensional]  Dimensiones2  [Tipo int Bidimensional] |

*Previamente en la función principal, se asignó dinámicamente memoria a la matriz “Dimensiones2”.*

*(En este caso el arreglo se declaró con la mitad de las dimensiones que el original ya que se pretende reducir al doble l*

*a imagen).*

Dentro de la función, las dimensiones se multiplican por 2 para devolverles las dimensiones de la matriz original y así poder recorrerla sin problemas.

Con dos ciclos for se va recorriendo la matriz original con sus respectivos índices [i][ j], mientras que la nueva matriz tendrá otros índices [y][x], son diferentes debido a que para reducir la imagen se deben eliminar las columnas y las filas de dos en dos.

Por lo que [i][j] incrementarán de dos en dos, mientras que [y][x] incrementarán de manera normal, es decir, de uno en uno.

Para así almacenar los pixeles a la mitad en la nueva matriz.

Dimensiones2[y][x]=Dimensiones[i][j];

1. *Ampliar.*

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Procesos** | **Salida** |
| Encabezado  [Tipo Estructura] | Regresar a dimensiones originales  Recorrer la matriz original de uno en uno y asignar valores a la nueva matriz de manera par.  Volver recorrer la matriz original de uno en uno y asignar valores a la nueva matriz de manera impar | Dimensiones  [Tipo int Bidimensional] |

*Previamente en la función principal, se asignó dinámicamente memoria a la matriz “Dimensiones2”*

*(En este caso el arreglo se declaró con el doble de las dimensiones que el original ya que se pretende ampliar al doble la imagen).*

En esta función a diferencia de la de reducir, se hace el proceso inverso.

Se vuelven a ocupar los índices [i][j] para la matriz original y para la nueva matriz [y][x], en este caso ahora [i][j] incrementarán de uno en uno y los valores de [y][x] de dos en dos.

Con dos ciclos for se recorrerá primero los índices pares de [y][x] empezando desde el 0, y guardando los valores de la matriz original a la nueva.

Después con otros dos ciclos for se recorrerá la matriz original, pero ahora con los índices impares de [y][x], empezando desde el valor 1 y guardando los valores de la matriz original a la nueva matriz.

Tras culminar estos ciclos se cumple el objetivo de duplicar tanto columnas como filas de la imagen original a la modificada.

1. *Guardar\_en\_Archivo.*

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Procesos** | **Salida** |
| Encabezado  [Tipo Estructura] | Solicitar el nombre del archivo la imagen a guardar.  Abrir el archivo en modo escritura binaria  Imprimir Encabezado [Texto]  Imprimir Pixeles [binario]  Cerrar Archivo | Dimensiones  [Tipo int Bidimensional] |

En esta función se le solicita al usuario introducir en la terminal el nombre del archivo a guardar las modificaciones de la imagen, se procede a abrir el archivo en modo escritura binaria, para así crear e imprimir lo que es el encabezado y luego con dos ciclos for, recorreremos la matriz para que guarde los píxeles de manera binaria en el archivo.

Estos se guardarán con la función predefinida fwrite, donde los parámetros serán los siguientes: (Matriz,TamañoenBytes(unsigned char),Elementos a leer,Archivo).

En este caso será así

fwrite((void \*)&(Dimensiones[i][j]),sizeof(unsigned char),1,Archivo);

Ya que se quiere obtener el contenido de la matriz, por eso el doble apuntador, sizeof(unsigned char) para determinar en cuántos bytes se va a ir moviendo, 1 elemento, ya que vamos a guardar de píxel en píxel al archivo.

1. *Liberar\_Matriz.*

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Procesos** | **Salida** |
| Encabezado  [Tipo Estructura]  Opción  [Tipo Entero] | Liberar memoria asignada a Dimensiones y a Dimensiones 2 (en caso que se le haya asignado a esta última) | Dimensiones  [Tipo int Bidimensional].  Dimensiones2  [Tipo int Bidimensional]. |

En esta función se espera obtener el valor de opción, la cual indica que modificación realizó el usuario, por lo que se determina si se libera espacio sólo en la “Matriz Original” o también en la “Nueva Matriz”.

Los encabezados realizan la función de liberar en el eje de las y la matriz correspondiente.

1. DISEÑO

**Diagrama IPO (Anexo en Página 10).**

**Pseudocódigo:**

**Principal**

{

Creditos();

Leer\_Imagen(Lectura\_Arch | Encabezado);

Asignar Memoria (Dimensiones);

Guardar\_en\_Matriz(Encabezado,Lectura\_Arch | Dimensiones);

Repite

{

LimpiarPantalla ();

Imprimir("PGM Optimizer Menu:");

Imprimir("1.- Negativo de la imagen");

Imprimir("2.- Ecualizar linealmente la imagen");

Imprimir("3.- Reducir la imagen");

Imprimir("4.- Ampliar la imagen");

Imprimir("Escoge una opción:");

Leer(Opcion);

Selección(Opcion)

{

caso 1:

Negativo(Encabezado | Dimensiones);

romper;

caso 2:

Ecualizar(Encabezado | Dimensiones);

romper;

caso 3:

Encabezado.Y/=2;

Encabezado.X/=2;

Asignar Memoria (Dimensiones2);

Reducir(Encabezado | Dimensiones, Dimensiones2);

romper;

caso 4:

Encabezado.Y\*=2;

Encabezado.X\*=2;

Asignar Memoria (Dimensiones2); Ampliar(Encabezado | Dimensiones, Dimensiones2);

romper;

Por defecto:

Imprimir("Opción Incorrecta, intenta ingresar otra opción.");

Imprimir("Presione Enter para continuar...");

Verificar=1;

romper;

}

} Mientras (Verificar=1);

Imprimir("Se realizó con éxito la operación.");

Imprimir("Presione Enter para guardar el archivo...");

si(Opcion=1 OR Opcion=2)

{

Guardar\_en\_Archivo(Encabezado | Dimensiones);

}

sino

{ Guardar\_en\_Archivo(Encabezado | Dimensiones2);

}

Liberar\_Matriz(Encabezado, Opcion | Dimensiones, Dimensiones2);

}

Creditos()

{

LimpiarPantalla();

Imprimir(“PGM Optimizer”);

Imprimir("Desarrollado por: César Mauricio Arellano Velásquez");

Imprimir("Presione Enter para continuar...");

LimpiarPantalla();

Imprimir("Objetivo del Programa:");

Imprimir("A través de este programa se busca que el usuario pueda editar de manera básica imágenes formato PGM.");

Imprimir("1.- Transformar imágenes PGM en su forma negativa.");

Imprimir("2.- Ecualizar de manera lineal la fotografía.");

Imprimir("3.- Reducir la imagen al doble.");

Imprimir(("4.- Ampliar la imagen al doble.");

Imprimir("Presione Enter para continuar...");

}

Leer\_Imagen(Lectura\_Arch | Encabezado)

{

Imprimir("Ingrese el nombre del archivo de la imagen");

Lectura(Lectura\_Arch);

AbrirArchivoBinario(Lectura\_Arch,Modo Lectura);

Si(Archivo=NULL)

{

imprimir("El archivo introducido no existe, intente de nuevo...");

salir(0);

}

sino

{

LeerArchivo(Archivo);

Si(Encabezado->Valor1 ='P' AND (Encabezado->Valor2='2' OR Encabezado->Valor2=='5'))

Verificar=1;

sino

Verificar=0;

si(Verificar=0)

{

Imprimir("El archivo no está en formato PGM, intenta con otro archivo.");

Imprimir("Saliendo del programa...\n");

salir(1);

}

CerrarArchivo(Archivo);

}

}

Guardar\_en\_Matriz(Encabezado,Lectura\_Arch | Dimensiones)

{

AbrirArchivoBinario(Lectura\_Arch,Modo Lectura);

RecorrerArchivo(Archivo,15 Bytes, Desde el Inicio del Archivo);

Para (Desde i = 0; Hasta i<Encabezado.Y; i++)

{

Para(Desde j = 0; Hasta j<Encabezado.X; j++)

{

LeerArchivoBinario(&Variable,TamañoBytes(char sin signo),1 Elemento,Archivo);

Dimensiones[i][j]=(char sin signo) Variable;

}

}

CerrarArchivoBinario(Archivo);

}

Negativo(Encabezado | Dimensiones);

{

Para(Desde i = 0; Hasta i<Encabezado.Y; i++)

{

Para (Desde j = 0; Hasta j<Encabezado.X; j++)

{

Dimensiones[i][j]=(Dimensiones[i][j] \* (-1))+255;

}

}

}

Ecualizar(Encabezado | Dimensiones);

{

Menor = 255;

Mayor = 0;

Para(Desde i = 0; Hasta i<Encabezado.Y; i++)

{

Para(Desde j = 0; Hasta j<Encabezado.X; j++)

{

Si(Menor > Dimensiones[i][j])

Menor=Dimensiones[i][j];

Si(Mayor < Dimensiones[i][j])

Mayor = Dimensiones[i][j];

}

}

m = 255 / (Mayor-Menor);

b = ((-1)\*(m \* Menor));

Para(Desde i = 0; i<Encabezado.Y; i++)

{

Para(Desde j = 0; j<Encabezado.X; j++)

{

Dimensiones[i][j]=(Dimensiones[i][j] \* m)+b;

}

}

}

Reducir(Encabezado | Dimensiones, Dimensiones2)

{

Encabezado.Y\*=2;

Encabezado.X\*=2;

Para(Desde i = 0; Hasta i<Encabezado.Y; i=i+2)

{

x=0;

for (Desde j = 0; Hasta j<Encabezado.X; j=j+2)

{

Dimensiones2[y][x]=Dimensiones[i][j];

x++;

}

y++;

}

}

Ampliar(Encabezado | Dimensiones, Dimensiones2);

{

Encabezado.Y/=2;

Encabezado.X/=2;

Para(Desde i = 0; Hasta i<Encabezado.Y; i++)

{

x=0;

Para(Desdent j = 0; Hasta j<Encabezado.X; j++)

{

Dimensiones2[y][x]=Dimensiones[i][j];

x=x+2;

}

y=y+2;

}

y=1;

Para(Desde i = 0; Hasta i<Encabezado.Y; i++)

{

x=1;

Para(Desde j = 0; Hasta j<Encabezado.X; j++)

{

Dimensiones2[y][x]=Dimensiones[i][j];

x=x+2;

}

y=y+2;

}

y=1;

Para(Desde i = 0; Hasta i<Encabezado.Y; i++)

{

x=0;

Para(Desde j = 0; Hasta j<Encabezado.X; j++)

{

Dimensiones2[y][x]=Dimensiones[i][j];

x=x+2;

}

y=y+2;

}

y=0;

Para(Desde i = 0; Hasta i<Encabezado.Y; i++)

{

x=1;

Para(Desde j = 0; Hasta j<Encabezado.X; j++)

{

Dimensiones2[y][x]=Dimensiones[i][j];

x=x+2;

}

y=y+2;

}

}

Guardar\_en\_Archivo(Encabezado | Dimensiones);

{

LimpiarPantalla();

Imprimir("Ingrese el nombre donde se guardará la imagen");

Leer(NombreArchivo);

AbrirArchivoBinario(NombreArchivo,Modo Escritura);

ImprimirEnArchivo("",Encabezado.Valor1,””);

ImprimirEnArchivo("",Encabezado.Valor1,””);

ImprimirEnArchivo(“ ”,Encabezado.X,” ”);

ImprimirEnArchivo(“ ”,Encabezado.Y,” ”);

ImprimirEnArchivo(“ ”,Encabezado.Max,” ”);

Para(Desde i = 0; Hasta i<Encabezado.Y; i++)

{

Para (Desde j = 0; Hasta j<Encabezado.X; j++)

{

ImprimirEnBinario(Dimensiones[i][j],  
TamañoBytes(char sin signo),1 Elemento,Archivo)

}

}

CerrarArchivoBinario(Archivo);

Imprimir("Se ha guardado con éxito el archivo, gracias por utilizar el programa");

}

Liberar\_Matriz(Encabezado, Opcion | Dimensiones, Dimensiones2)

{

Si(Opcion=1 OR Opcion=2)

free(Dimensiones);

Sino

{

free(Dimensiones);

free(Dimensiones2);

}

}

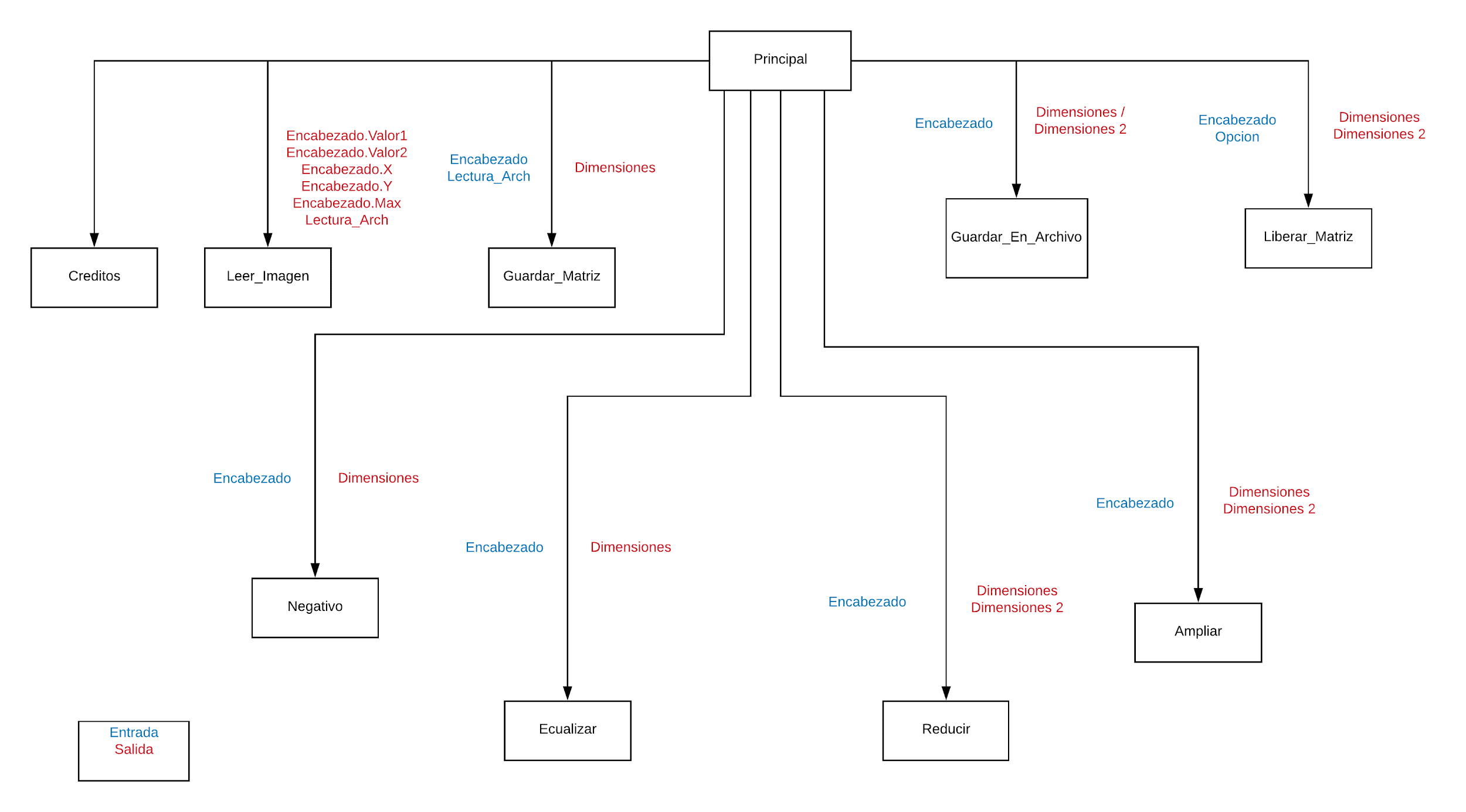


Diagrama IPO