DECSAI

MEMORIA PRACTICA FINAL

METODOLOGIA DE LA PROGRAMACIÓN

Marcos Avilés Luque 25342389-T

Javier García Mancilla 15517099-B

En este documento redactaremos los problemas que han ido surgiendo a la hora de la implementación de nuestro código a nuestra práctica.

Iremos comentando cada uno de los errores, dividiendo el desarrollo en dos partes. En una parte trataremos los problemas ocasionados en el apartado realizado con la metodología del vector dinámico, y en el siguiente apartado trataremos los problemas ocasionas en la metodología de matriz dinámica.

A) VECTOR

- PROBLEMA DE INICIALIZACIÓN

Tuvimos un problema, que durante la ejecución del compilador no definía ningún tipo de error, pero a la hora de evaluarlo mediante "Valgrind" nos mostraba errores secundarios. Estos errores sumaban una cantidad de 105900 errores.

El motivo de estos errores era que no se inicializaba las posiciones del vector en "0", es decir, equivalente a negro. Teniendo en cuenta que aun no inicializando estas posiciones, la ejecución del programa era correcta.

Dicha solución fue inicializar en cada uno de los métodos correspondientes estos valores a "0".

- PROBLEMA CON LA LIBERACION DE MEMORIA O DESTRUCTOR

A la hora de la creación del método destruir, omitimos la comprobación de que este vector contenía o no datos en memoria.

Por ello, cada vez que se usaba dicho método, intentaba eliminar memoria no reservada.

Nos dimos cuenta, al evaluar la ejecución con "Valgrind", que liberaba más bloques de memoria que los que realmente se usaban.

Dicha solución para este problema fue implementar la comprobación de si el vector estaba vacío o no.

- PROBLEMA CON LA SOBRECARGA DEL OPERADOR DE SUMA.

```
Imagen Imagen::operator+(const Imagen &p) const{
    int filas;
    if(nfilas<p.nfilas){
        filas=p.nfilas;
    }
    else {
        filas=nfilas;
    }
    Imagen resultado(filas, ncolumnas+p.ncolumnas);
    int posicion=0;
    for(int i=0; i<resultado.nfilas; i++){
        for(int x=0; x<ncolumnas; x++){
            resultado.datos[posicion]=get(i,x);
            posicion++;
        }
        for(int j=0; j<p.ncolumnas; j++){
            resultado.datos[posicion]=p.get(i,j);
            posicion++;
        }
    }
    return resultado;
}
```

Este algoritmo, para la sobrecarga del operador de suma, el cual implementamos en un primer momento, el compilador lo evaluaba correctamente y su función la realizaba correctamente (concatenaba perfectamente las dos imágenes), pero a la hora de evaluarlo con "Valgrind" no aparecían 60000 errores por el motivo del método "get", el cual es un método "const" al igual que este método del operador de suma. Por consiguiente resolvimos el problema con el siguiente código:

```
Imagen Imagen::operator+(const Imagen &p) const{
    int filas;
    if(nfilas<p.nfilas){
        filas=p.nfilas;
    }
    else {
        filas=nfilas;
    }
    Imagen resultado(filas, ncolumnas+p.ncolumnas);
    int posicion=0;
    for(int i=0; i<resultado.nfilas; i++){
        if(i<nfilas){
            for(int x=0; x<ncolumnas; x++){
                resultado.datos[posicion]=datos[i*ncolumnas+x];
                posicion++;
            }
        }
        else{
            for(int x=0; x<ncolumnas; x++){
                posicion++;
            }
        }
        if(i<p.nfilas){
            for(int j=0; j<p.ncolumnas; j++){
                resultado.datos[posicion]=p.datos[i*p.ncolumnas+j];
                posicion++;
            }
        }
        else{
            for(int j=0; j<p.ncolumnas; j++){
                posicion++;
            }
        }
    }
    return resultado;
}
```

Postdata: Creemos que esto es un chapuza.

B) MATRIZ

- PROBLEMA DE INICIALIZACIÓN

Tuvimos el mismo problema comentado en el apartado anterior, pero con la matriz.

La solución fue prácticamente la misma.

- PROBLEMA AL CREAR LA MATRIZ

En un principio, implementamos la creación de la matriz usando un array 1D de punteros a arrays, ocasionando problemas a la hora del acceso a los elementos y derivados.

La solución fue implementar la matriz usando una array 1D de punteros a un único array.