multi

Fonaments de Programació II Pràctica 3: “El joc del picross”

**Curs 2021-22**

**Estudiants:** Matías Ariel Larrosa Babio,

Pau Revertè,

Pablo

**Professor/a:** Ramon Castells

**Data de lliurament:** x/x/2022

**Contingut**

[Decisions de disseny 3](#_bookmark0)

[Procediments base 3](#_bookmark1)

[Procediments extres 5](#_bookmark2)

[Algorisme 8](#_bookmark3)

[Joc de proves 16](#_bookmark4)

[Procediments base: 16](#_bookmark5)

[Procediments extres: 17](#_bookmark6)

[Procediments en profunditat: 18](#_bookmark7)

[Conclusions 22](#_bookmark8)

[Checklist 23](#_bookmark9)

# Decisions de disseny

A l’hora de dissenyar el programa el problema principal és crear un registre útil per guardar tota la informació necessària del joc. El registre que hem decidit crear és casella\_t i té 3 camps on és guarda tota la informació de la imatge.

**typedef struct{**

**char** valor**;**

**bool** revelat**;**

**bool** flag**;**

**}**casella\_t**;**

**Valor** és l’estat en la qual una casella es troba. Guardarà el caràcter ‘0’ quan en aquella casella estigui buida i, un ‘1’ quan en aquella no ho estigui (■). Aquest valor només es modificarà al inicialitzar el tauler.

**Revelat** és un valor booleà que determina si aquella casella es visible o no per a l’usuari. Al inicialitzar el tauler totes les posicions estaran ocultes (false) i, l’usuari podrà revelar aquella posició seleccionant aquella casella.

**Flag** és un valor booleà que determina si aquella casella esta marcada per l’usuari.Funciona semblant a una bandera del buscamines. L’usuari podrà marcar una casella com un flag per tenir una guia visual per resoldré el tauler a més, podrà introduir un valor especial per triar automàticament totes les caselles que no estiguin marcades.

Així doncs, amb el registre definit podem dissenyar els procediments de la pràctica.

## Procediments base

En aquest apartat s’explicaran els procediments mínims demanats; carregarDades, calcularCantonades, selecció, printTaulerJoc.

**bool** carregarDades(**char** \*filename, **int** \*m, **int** \*n, **int** \*e, **casella\_t** joc[][MAXROWCOL]);

Carrega el contingut del fitxer amb el nom **filename** (\*.txt) al tauler passat “joc”.

El format que accepta el nostre programa es un fitxer de text on la primera línia conte 3 enters corresponents a; les files del tauler, les columnes del tauler, i els errors màxims que te l’usuari al jugar en aquest tauler. La resta del fitxer conté files \* columnes 1s i 0s que correspon al tauler.

Si hi ha menys informació de la que deuria haver-hi o si no s’ha pogut obrir el fitxer el mètode retornarà fals.

**void** calcularCantonades(**int** m, **int** n, **casella\_t** joc[][MAXROWCOL], **int** maxCombinations, **char** cantonades[][maxCombinations]);

Calcula els valors de les files i les columnes que corresponen a la disposició del tauler en aquella fila o columna.

El mètode demana el tauler (dimensions mxn) i una taula on es guardaran aquests valors que tindra m+n files imaxCombinations[[1]](#footnote-1).

Les dades de cada cantonada es guardaran per files on les m primeres files ([0, m-1]) correspondran a les cantonades de les files i les n ([m, n-1]) següents correspondran a les de les columnes. Cada fila de la taula tindrà un valor centinella al final quan la disposició d’aquella cantonada hagi acabat.

Per calcular la disposició de les cantonades es recorre el tauler per files i columnes, contant quants cops apareix un 1 i guardant aquest nombre quan es troba un 0.

**bool** seleccio(**int** m, **int** n, **casella\_t** joc[][MAXROWCOL], **int** i, **int** j, **bool** flag);

Selecciona una posició del tauler del joc, decidint també si es vol revelar la posició o si es vol posar un flag. Per fer-hi aquesta funció necessitem principalment la dimensió de la taula tant les files com les columnes, el tauler del joc (joc[][MAXROWCOL]).

Aquest procés és un booleà per aquest motiu retrornarà si la posició és correcta (true) o no (false). Primerament comneça revisant que la posición seleccionada es trobe dintre de les mesures de la taula, que no es posen números negatius i que la posición seleccionada no es trobi ja revelada. Un cop revisat aixó donarà la resposta a posicio\_correcta, permeten així o no si entra dintre del if que utilitza el mètode .revelat si l’usuari revela la posición o el .flag si vol posar-hi una flag.

**bool** afegirExtensio (**char**\* fitxer, **char**\* extensio);

Afegeix una extenció al final de la cadena de text eliminan en si el carácter ‘\n’ del final d’aquest. Per fer aquest procés necessitem la cadena de text a modificar i la cadena de text que conté l’extensió a afegir.

Primerament, inicialitzem una variable que serà la que determinarà la posició a la cadena de text començant en 0. Un cop inicialitzada la variable farem un bucle fins que saltLinea es trobi en la posició del '\n', cal tenir en compte que repetirà el bucle avançant mentre sigui diferent de '\n' i saltLinea sigui més petit que CADMAX. Amb el saltLinea a la posició on es descobreix el '\n' sumarem 5 posicions per poder posar les extensions ('.txt\0' o '.pbm\0'), en cas d'exercir-se de la CADMAX retornarà false, ja que no s'haurà posat l'extensió de forma correcta. En cas de poder haver pogut afegir l'extensió entrarà a un bucle el qual escriurà l'extensió al fitxer.

Finalment fitxer[] a la posició i + saltLinea es col·locarà un '\0' per assegurar que existeixi aquest, retornant així un true a la funció d'afegirExtensió.

**bool** combinarPath (**char** \*fitxer, **char** \*path, **char** \*pathFinal);

Ajuntar el path relatiu i el nom d'un fitxer al path final. Per fer-hi aquesta operació és necessari el nom del fitxer a crear, el path del directori relatiu i el pathFinal que és el path relatiu del fitxer. En aquest procés es retornarà true si s'han pogut ajuntar el path i el fitxer, si no, retornarà false. El primer procés que fem servir és el de posar el path al pathFinal, per posteriorment fer un bucle per a introducir el fitxer dintre del pathFinal. Cal tenir en compte que no s'ha de sobreescriure per aquest motiu se suma el indexfinal que anteriorment determinava el final del path. Finalment, un cop ja es troba tot dintre del pathFinal, es revisa que sigui més petit que CADMAX, si és més petit estarà tot dintre i retornarà true a la funció, si no és així no entrarà al if i retornarà un false.

**void** restaurarJoc (**int** m, **int** n, casella\_t joc[][MAXROWCOL]);

Restaura tot el tauler del joc. Aquest procés el fa mitjançant principalment dos bucles que són els encarregats de recórrer la taula joc[i][j] de .flag, .revelat i .valor, el que fa és posar a la posició d'aquell moment false, és a dir, un 0 a les flags i a revelat, ja que se està restaurant el joc d'aquesta forma. També cal recordar que hi ha un if, que és l'encarregat d'incrementar els píxels restants fins al seu màxim, això ho fa si en la posició actual troba un 1, ja que seria la posició d'un quadrat.

**void** taulerAleatori (**int** m, **int** n, casella\_t joc[][MAXROWCOL]);

Emplenar de jugades un tauler de mesures mxn de forma aleatòria. És necessari principalment la dimensió de la taula, el tauler de joc [][MAXROWCOL] i els píxels restants que serà la quantitat de posicions correctes que hi haurà. Aquest procediment començarà amb la densitat que serà fent un procés mitjançant aquesta fórmula ((rand() % 4 + 1) \* 20) que proporcionarà que el tauler estigui entre un 20 i un 80 % de ple, això ho fem per evitar un joc el qual sigui completament buit o completament ple. Un cop fet això farem dos bucles que determinin la posició a .valor, .revelat i .flag. Cal crear una variable que és 'a' que compararà un número aleatori entre el 1 i el 100 amb la densitat, en cas de ser-hi menor o igual serà true, si no, seria false, si és true incrementarà el valor de pixelsRestants, si no serà false i no entrarà al if. Després s'escriurà el valor 'a' a la posició joc[i][j].valor, mentre també es posa en false aquesta mateixa posició al .revelat i al .flag per posteriorment començar el joc amb un tauler totalment aleatori i nou.

## Procediments extres

**funció** capgira\_imatge(imatge: imatgepgm\_t) **retorna booleà és**

Capgira la imatgepgm\_t passada per referència.

Per fer-ho, sempre que hi hagi una imatge carregada correctament, recorreré la matriu fins a la **meitat de les files** (alçada) intercanviant els valors a les **files** inicials per les finals5.

Novament, donat que l’única condició per realitzar aquesta funció és que hi hagi una imatge carregada, el valor que retorna és imatge.totcorrecte.

**funció** mirall\_imatge(**var** imatge: imatgepgm\_t) **retorna booleà és**

*Miralla* la imatgepgm\_t passada per referència.

Per fer-ho, sempre que hi hagi una imatge carregada correctament, recorreré la matriu fins a la **meitat de les columnes** (amplada) intercanviant els valors a les **columnes** inicials per les finals6.

Novament, donat que l’única condició per realitzar aquesta funció és que hi hagi una imatge carregada, el valor que retorna és imatge.totcorrecte.

**funció** aixafa\_imatge ( **var** imatge: imatgepgm\_t, ratioaplastar: **enter** ) **retorna booleà és**

Aixafa la imatge passada per referència com si l’haguessin trepitjat, per un ràtio de ratioaplastar. Per tant, la imatge tindrà una alçada final de (imatge.files div ratioaplastar).

Per fer-ho, només aixafaré la imatge sempre i quan hi ha una imatge carregada correctament i el ratioaplastar estigui dins del rang ((0, imatge.files]) ja que si és més gran que imatge.files les files resultants serien 0.

Un cop comprovades aquestes condicions recorreré “imatge.matriuimatge” reassignant 1 de cada ratioaplastar files a les files inicials de imatge.matriuimatge.

4 Els valors de píxel que hi ha amb n := 4 són {0, 64, 128, 192}, (van de 256/n en 256/n). Per tant, si el píxel val 73 el píxel final valdrà 64 ja que al fer la divisió entera per 256/n valdrà 1.

5 imatge.matriuimatge[i][j] intercambiar-la per imatge.matriuimatge[imatge->files – i – 1][j].

6 imatge.matriuimatge[i][j] intercambiar-la per imatge.matriuimatge[i][imatge->columnes – j – 1].

Aquesta funció retorna “ratiocorrecte i imatge.totcorrecte”, que significa “si el ràtio està dins del rang ((0, imatge.files]) i hi ha una imatge carregada” retorna cert, fals en cas contrari.

**funció** comprovar\_nom\_arxiu\_correcte(nomarxiu: **taula[] caràcters**) **retorna booleà és**

Comprova que la cadena passada per referència acabi amb .pgm. Retorna cert si la cadena acaba en .pgm, en cas contrari fals.

**acció** demanar\_nom\_arxiu ( **var** nomarxiu: **taula[] de caràcters** ) **és**

Demana una cadena de caràcters i la guarda a nomarxiu fins que sigui correcte. Per fer-ho s’ajuda de comprovar\_nom\_arxiu\_correcte().

**acció** escriureopcions() **és**

Escriu les opcions del programa (carregar imatge, desar imatge, mitjana píxels, binaritzar imatge, negativitzar imatge, reduir nivells, capgirar imatge, mirllar imatge, aixafar imatge, puzlejar imatge, sortir).

**acció** combinarpath(nomarxiu: **taula[] de caràcters** , pathrelatiu: **taula[] de caràcters** , **var** pathfinal:

**taula[] de caràcters**) **és**

Combina dues cadenes de caràcters una darrere de l’altre.

**funció** puzle\_imatge ( **var** imatge: imatgepgm\_t, **var** dfil: **enter** , **var** dcol: **enter** ) **retorna booleà és**

Transforma la imatgepgm\_t passada per referència en un puzle que tindrà com a mínim MINIMDIVISIONS \* MINIMDIVISIONS7 peces.

Per poder *puzlejar* la imatge necessito trobar un divisor de les files i les columnes més petit que aquestes mateixes8 i, tenir una imatge carregada correctament. Els divisors de les dimensions els guardaré a mfil i a mcol que són valors passats per referència, per poder mostrar al *main* quantes peces té el puzle. Un cop comprovades aquestes condicions recorreré totes les peces ([0, mfil \* mcol)) intercanviant els píxels de la peça recorreguda pels píxels d’una altre peça aleatòria.

Aquesta funció retorna “imatge.totcorrecte i dfil <= imatge->files / 2 i dcol <= imatge->columnes / 2 i MINIMDIVISIONS != 1”, que significa “si hi ha una imatge carregada i s’ha trobat un divisor de les files i les columnes i MINIMDIVISIONS és diferent d’1” retorna cert, fals en cas contrari.

7 MINIMDIVISIONS és una constant que té de valor inicial 3. I que utilitza puzle\_imatge() per trobar un divisor de cada dimensió. Rang ∈ 𝑅: (1, min(imatge->files / 2, imatge->columnes / 2)]

8 Si no trobés aquest múltiple significaria que aquella dimensió és un nombre primer o bé, el MINIMDIVISIONS és més gran que tots els divisors restants de la dimensió ometent la dimensió mateixa i l’1.

Necessito trobar un divisor diferent a 1 o a la dimensió mateixa. Ja que 1 fa que com a mínim una de les dimensions de cada peça sigui la dimensió mateixa. Per tant, només hi hauria 1 peça, i la dimensió mateixa fa que estigui intercanviant peces on com a mínim una de les seves dimensions és 1, on el pitjor dels casos estaria movent píxel a píxel.

**algorisme** Pràctica2 : "qüestió d'imatge" **és**

A l’hora de dissenyar el *main* una cosa que cal destacar és com deso i d’on llegeixo les imatges .pgm. Per mantenir un ordre tindré dues carpetes, una des d’on llegiré les imatges base (.\imatges\) i una altre on desaré les imatges modificades (.\imatgesModificades\). Per tant, per poder utilitzar el meu programa **cal tenir aquestes dues carpetes creades on hi és el *main* i guardar les imatges base a la carpeta “imatges”**. Per poder carregar les imatges i desar-les al directori corresponent utilitzaré el mètode combinar\_path on combinaré el nom de l’arxiu amb el path corresponent9.

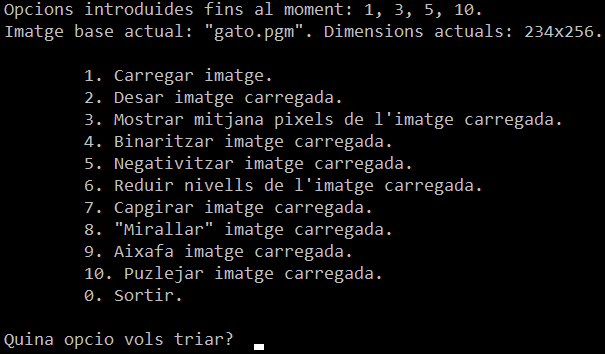
El cos del *main* és simple, s’escriu un missatge de benvinguda seguit d’un bucle fer/mentre que escriu les opcions del programa i demana opcions a l’usuari. Aquest bucle és repetirà fins que l’usuari triï l’opció 0 (sortir) o introdueixi un caràcter com a opció. La primera opció introduïda té tres sortides:

1. L’usuari ha triat una opció correcta. S’executarà i és netejarà la pantalla. I, en comptes del missatge de benvinguda és mostraran les opcions triades fins al moment i, si s’ha carregat una imatge correctament, s’escriurà el seu nom i les seves dimensions actuals.

Si no s’ha triat l’opció 1 o bé, si no s’ha carregat una imatge correctament es mostraran missatges corresponents. “NO hi ha una imatge carregada” i “La carrega ha sortit malament”, respectivament. Posteriorment, sortirà novament totes les opcions demanant a l’usuari triar una nova opció.

1. L’usuari **no** ha triat una opció correcta. Per tant, s’escriurà que no és una opció vàlida, es netejarà la pantalla i, en comptes del missatge de benvinguda s’escriurà “Encara no s’ha triat una opció vàlida”. Posteriorment, sortirà novament totes les opcions demanant a l’usuari triar una nova opció.
2. L’usuari **escriu un caràcter**. Es surt del programa.

Cada opció que pot triar l’usuari té un cos molt semblant. S’escriu el nom de la opció triada, si cal és demanen els valor necessaris per executar el procediment, i s’escriu si el procediment s’ha executat correctament o no. La consola després d’haver triat 4 opcions:



9 Al main hi ha dues cadenes inicialitzades, pathrelatiuimatges i pathrelatiuimatgesMOD, que guarden “.\\imatges\\” i “.\\imatgesModificades\\” respectivament. A l’utilitzar carregar\_pgm combinaré el nom de l’arxiu i pathrelatiuimatges i, en desar\_pgm combinaré el nom de l’arxiu amb pathrelatiuimatgesMOD.

# Anàlisi cost

**const**

MAXIMARESOLUCIO := **256** ; VALORMAXIMPIXEL := **256** ; MAXOPCIONS := **100** ; CADENAMAX := **100** ; MINIMDIVISIONS := **3** ; CENTINELLA := **-1** ;

**fconst tipus**

imatgepgm\_t: **registre**

columnes, files, valormaximpixel: **enter** ; mode: taula[4] caracters ;

matriuimatge: taula[MAXIMARESOLUCIO][MAXIMARESOLUCIO] **de enters** ;

totcorrecte: **booleà** ;

**fregistre ftipus**

**$ comproba que el mode passat sigui correcte $**

**funció** modecorrecte ( mode: **taula[] de caràcters** ) **retorna booleà és inici**

**retorna** ( mode[0] = 'P' **i** mode[1] = '2' **i** mode[2] = '\n' **i** mode[3] = '\0' ) ;

**fi ffunció**

**$ actualitza el mode de la imatge pasada per referéncia $ acció** actualitzarmode ( **var** imatge imatgepgm\_t ) **és inici**

imatge.mode[0] := 'P';

imatge.mode[1] := '2';

imatge.mode[2] := '\n';

imatge.mode[3] := '\0';

**fi facció**

**$ comproba que les dimensions de la imatge siguin correctes ((0, 256]) $ funció** comprobafilescolumnes ( imatgepgm\_t imatge ) **retorna booleà és inici**

**retorna** ( imatge.files > **0 i** imatge.files <= MAXIMARESOLUCIO **i** imatge.columnes > **0 i** imatge.columnes <= MAXIMARESOLUCIO ) ;

**fi ffunció**

**$ carrega el fitxer amb el nom passat a la imatgepgm\_t $**

**funció** carrega\_pgm ( nomfitxer: **taula[] de caràcters** , **var** imatge: imatgepgm\_t ) **retorna booleà és var**

n, m, count: **enter** ; fit: **fitxer** ;

**fvar inici**

**$ inicialitzo variables $**

fit := obrirF ( nomfitxer, "L" ) ; imatge.totcorrecte := **fals** ; count := **0** ;

**$ codi $**

**si** ( fit != NULL ) **fer**

llegirF(fit, imatge.mode);

**si** ( modecorrecte(imatge.mode) ) **fer** actualitzarmode(imatge); llegirF(fit, imatge.columnes); llegirF(fit, imatge.files);

**si** ( comprobafilescolumnes(imatge) ) **fer**

llegirF(fit, imatge.valormaximpixel);

**si** ( imatge.valormaximpixel = VALORMAXIMPIXEL ) **fer per** ( n = **0** ; n < imatge.files; n := n + **1** ) **fer**

m := **0** ;

**mentre** ( m < imatge.columnes **i no** finalF(fit) ) **fer**

llegirF(fit, imatge.matriuimatge[n][m]);

**si** ( imatge.matriuimatge[i][j] > VALORMAXIMPIXEL ) **fer**

imatge.matriuimatge[i][j] := VALORMAXIMPIXEL;

**sino si** ( imatge.matriuimatge[i][j] < **0** ) **fer**

imatge.matriuimatge[i][j] := **0** ;

**fsi**

count := count + **1** ; j := j + **1** ;

**fmentre fper**

imatge.totcorrecte := ( count = imatge.columnes \* imatge.files ) ;

**fsi fsi**

**fsi fsi**

tancarF(fit);

**retorna** imatge.totcorrecte;

**fi ffunció**

**$ desa la imatge en un fitxer pgm nou amb el format correcte $**

**funció** desar\_pgm ( nomfitxer: **taula[] de caràcters** , imatge: imatgepgm\_t ) **retorna booleà és var**

n, m: **enter** ; totbe: **booleà** ; fit: **fitxer** ;

**fvar inici**

**$ inicialitzar $**

totbe := **fals** ;

fit := obrirF(nomfitxer, "E");

**$ codi $**

**si** ( imatge.totcorrecte **i** fit != NULL ) **fer** escriureF(fit, imatge.mode); escriureF(fit, imatge.columnes); escriureF(fit, imatge.files); escriureF(fit, imatge.valormaximpixel);

**per** ( n := **0** ; n < imatge.files; n := n + **1** ) **fer**

**per** ( m := **0** ; m < imatge.columnes; j := j + **1** ) **fer**

escriureF(fit, imatge.matriuimatge[i][j]);

**fper fper**

totbe := **cert** ;

**fsi**

**retorna** totbe;

**fi ffunció**

**$ retorna la mitjana de pixels si hi ha una imatge carregada, en cas contrari retorna -1 $**

**funció** mitjana\_pixels ( imatge: imatgepgm\_t ) **retorna real és var**

mitjana: **real** ;

totalpixels, sumapixels, n, m: **enter** ;

**fvar inici**

**$ inicialitzar $**

**si** ( imatge.totcorrecte ) **fer**

totalpixels := imatge.columnes \* imatge.files ;

**fsi**

sumapixels := **0** ;

**$ codi $**

**si** ( imatge.totcorrecte ) **fer**

**per** ( n := **0** ; n < imatge.files; n := n + **1** ) **fer**

**per** ( j := **0** ; j < imatge.columnes; j := j + **1** ) **fer**

sumapixels += imatge.matriuimatge[i][j];

**fper fper**

mitjana := sumapixels / totalpixels;

**sino**

mitjana := **-1** ;

**fsi**

**retorna** mitjana ;

**fi ffunció**

**$ binaritzar la imatge segons un valor llindar $**

**funció** binaritza\_imatge ( **var** imatge: imatgepgm\_t, llindar: **enter** ) **retorna booleà és var**

valorllindarcorrecte: **booleà** ; n, m: **enter** ;

**fvar inici**

**$ inicialitzar $**

valorllindarcorrecte := ( llindar > **0 i** llindar <= VALORMAXIMPIXEL ) ;

**$ codi $**

**si** ( valorllindarcorrecte **i** imatge.totcorrecte ) **fer per** ( n = **0** ; n < imatge.files; n := n + **1** ) **fer**

**per** ( m = **0** ; j < imatge.columnes; m := m + 1) **fer si** ( imatge.matriuimatge[n][m] < llindar ) **fer**

imatge.matriuimatge[n][m] := **0** ;

**sino**

imatge.matriuimatge[n][m] := VALORMAXIMPIXEL;

**fsi fper**

**fper fsi**

**retorna** ( valorllindarcorrecte **i** imatge.totcorrecte ) ;

**fi ffunció**

**funció** negativa\_imatge ( **var** imatge: imatgepgm\_t) **retorna booleà és**

**var**

n, m: **enter** ;

**fvar inici**

**$ codi $**

**si** ( imatge.totcorrecte ) **fer**

**per** ( n := **0** ; n < imatge.files; n := n + **1** ) **fer**

**per** ( m := **0** ; m < imatge.columnes; m := m + **1** ) **fer**

imatge.matriuimatge[n][m] := imatge.valormaximpixel - imatge.matriuimatge[n][m];

**fper fper**

**fsi**

**retorna** imatge.totcorrecte;

**fi ffunció**

**$ redueix nivells $**

**funció** redueix\_nivells ( **var** imatge: imatgepgm\_t, n: **enter** ) **retorna booleà és var**

j, k: **enter** ; divisor256: **booleà** ;

**fvar inici**

divisor256 := ( n > **0 i 256** mod n = **0** ) ;

**$ codi $**

**si** ( divisor256 **i** imatge.totcorrecte ) **fer**

**per** ( j = **0** ; j < imatge.totcorrecte; j := j + **1** ) **fer per** ( k = **0** ; k < imatge.totcorrecte; k := k + **1** ) **fer**

imatge.matriuimatge[j][k] := **256** / n \* ( imatge->matriuimatge[j][k] div (256 / n));

**fper fper**

**fsi**

**retorna** ( divisor256 **i** imatge.totcorrecte ) ;

**fi ffunció**

**$ FUNCIONS EXTRES $**

**$ capgira la imatge, per fer-ho intercambiare les files inicials per les finals $**

**funció** capgira\_imatge ( imatge: imatgepgm\_t ) **retorna booleà és var**

n, m, aux: **enter** ;

**fvar inici**

**si** ( imatge.totcorrecte ) **fer**

**$ intercambiare la meitat de la files perque si no la imatge quedaria igual $**

**per** ( n := **0** ; n < imatge.files / **2** ; n := n + **1** ) **fer per** ( m := **0** ; m < imatge.columnes; m := m + **1** ) **fer**

aux := imatge.matriuimatge[imatge.files - n - 1][m]; imatge.matriuimatge[imatge.files - n - 1][m] := imatge.matriuimatge[n][m]; imatge.matriuimatge[n][m] := aux;

**fper fper**

**fsi**

**retorna** imatge.totcorrecte;

**fi ffunció**

**$ miralla la imatge, per fer-ho intercambiare les columnes inicials per les finals $**

**funció** mirall\_imatge ( **var** imatge: imatgepgm\_t ) **retorna booleà és var**

n, m, aux: **enter** ;

**fvar inici**

**si** ( imatge.totcorrecte ) **fer**

**per** ( n := **0** ; n < imatge.files; n := n + **1** ) **fer**

**$ intercambiare les columnes fins a la meitat perque si no la imatge quedaria igual $**

**per** ( m := **0** ; m < imatge.columnes / 2; m := m + **1** ) **fer**

aux := imatge.matriuimatge[n][imatge.columnes - m - 1]; imatge.matriuimatge[n][imatge.columnes - m - 1] := imatge.matriuimatge[n][m]; imatge.matriuimatge[n][m] := aux;

**fper fper**

**fsi**

**retorna** imatge.totcorrecte;

**fi funció**

**$ aixafa la imatge, per fer-ho reassignaré 1 cada ratioaplastar files a les files inicials de la imatge $**

**$ i després reduiré imatge->files a imatge->files div ratioaplastar $**

**funció** aixafa\_imatge ( **var** imatge: imatgepgm\_t, ratioaplastar: **enter** ) **retorna booleà és var**

n, m: **enter** ; **$ enters per recorrer la matriu $**

filaaplastada: **enter** ; **$ filaaplastada contador de les files a re assignar de la imatge $**

ratiocorrecte: **booleà** ; **$ comprbar que el ratio sigui correcte (0, imatge.files] $**

**fvar inici**

**$ inicialitzar $**

filaaplastada := **0** ;

ratiocorrecte := ( ratioaplastar > **0 i** ratioaplastar <= imatge.files ) ;

**$ codi $**

**si** ( imatge.totcorrecte **i** ratiocorrecte ) **fer per** ( n := **0** ; n < imatge.files; n := n + **1** ) **fer**

**per** ( m := **0** ; m < imatge.columnes; m := m + **1** ) **fer**

**$ nomes reassigno 1 cada ratioaplastar files $**

**si** ( n mod ratioaplastar = **0** ) **fer**

imatge.matriuimatge[filaaplastada][m] := imatge.matriuimatge[n][m];

**fsi fper**

**si** ( n mod ratioaplastar = **0** ) **fer**

filaaplastada := filaaplastada + **1** ;

**fsi fper**

imatge->files := imatge->files div ratioaplastar;

**fsi**

**retorna** imatge.totcorrecte **i** ratiocorrecte;

**fi ffunció**

**$ comprova si el nom del arxiu passat és correcte, és a dir que acabi amb .pgm $ funció** comprovar\_nom\_arxiu\_correcte ( nomarxiu: **taula[] caràcters** ) **retorna booleà és var**

n: **enter** ; nomcorrecte: **booleà** ;

**fvar inici**

**$ inicialitzar $** nomcorrecte := **fals** ; n := **0** ;

**$ codi $**

**mentre** ( nomarxiu[n] != '\0' ) **fer si** ( nomarxiu[n] = '.' ) **fer**

n := n + **1** ;

**si** ( nomarxiu[n] = 'p' ) **fer**

n := n + **1** ;

**si** ( nomarxiu[n] = 'g' ) **fer**

n := n + **1** ;

**si** ( nomarxiu[n] = 'm' ) **fer**

n := n + **1** ;

**si** ( nomarxiu[n] = '\0' ) **fer**

nomcorrecte := **cert** ;

**sino**

n := n + **1** ;

**fsi sino**

n := n + **1** ;

**fsi sino**

n := n + **1** ;

**fsi sino**

n := n + **1** ;

**fsi sino**

n := n + **1** ;

**fsi fmentre**

**retorna** nomcorrecte;

**fi ffunció**

**$ escriu les opcions de la pràctica $**

**acció** escriureopcions () **és inici**

**escriure** ("1. Carregar imatge.");

**escriure** ("2. Desar imatge carregada");

**escriure** ("3. Mostrar mitjana pixels de l'imatge carregada.")

**escriure** ("4. Binaritzar imatge carregada."); **escriure** ("5. Negativitzar imatge carregada."); **escriure** ("6. Reduir nivells de l'imatge carregada."); **escriure** ("7. Capgirar imatge carregada.");

**escriure** ("8. Mirallar imatge carregada."); **escriure** ("9. Aplastar imatge carregada."); **escriure** ("10. Puzlejar imatge carregada."); **escriure** ("0. Sortir.");

**fi facció**

**$ demana el nom del arxiu i el demana fins que l'usuari introdueixi un nom correcte $**

**acció** demanar\_nom\_arxiu ( **var** nomarxiu: **taula[] de caràcters** ) **és var**

c: **caràcter** ; n: **enter** ;

**fvar inici**

**fer**

**escriure** ( "Quin nom te/tindra l'arxiu?" );

**llegir** ( c ); n := **0** ;

**mentre** ( c != '\n' ) **fer**

nomarxiu[i] := c; n := n + 1;

**llegir** ( c ); **fmentre** nomarxiu[n] := '\0';

**si** ( **no** comprovar\_nom\_arxiu\_correcte ) **fer**

**escriure** ( "El nom del arxiu ", nomarxiu," no es correcte el nom ha de acabar amb .pgm." ) ;

**fsi**

**mentre** ( **no** comprovar\_nom\_arxiu\_correcte ( nomarxiu ));

**fi facció**

**$ combina dues cadenes en una sola $**

**acció** combinarpath ( nomarxiu: **taula[] de caràcters** , pathrelatiu: **taula[] de caràcters** , **var** pathfinal: **taula[] de caràcters** ) **és**

**var**

n, m: **enter** ;

**fvar inici**

**per** ( n := **0** ; pathrelatiu[n] != '\0'; n := n + **1** ) **fer**

pathfinal[n] := pathrelatiu[n];

**fper**

**per** ( m := n; nomarxiu[n - m] != '\0'; n := n + **1** ) **fer**

pathfinal[n] := nomarxiu[n - m];

**fper**

pathfinal[n] = '\0';

**fi facció**

**$ transforma la imatge carregada en un puzle $**

**funció** puzle\_imatge ( **var** imatge: imatgepgm\_t, **var** dfil: **enter** , **var** dcol: **enter** ) **retorna booleà és var**

a, b, c, d, aux: **enter** ; posicio, aleatori: **enter** ;

**fvar inici**

dfil = MINIMDIVISIONS; mcol = MINIMDIVISIONS;

**$ trobo el primer multiple de cada dimensio més gran o igual a MINIMDIVISIONS $**

**mentre** ( imatge.files mod dfil != **0 i** dfil <= imatge.files / **2** ) **fer**

dfil := dfil + **1** ;

**fmentre**

**mentre** ( imatge.columnes mod dcol != **0 i** dcol <= imatge.columnes / **2** ) **fer**

dcol := dcol + **1** ;

**fmentre**

**$ si hi ha una imatge carregada i existeix un divisor més gran o igual al MINIMDIVISIONS es pot puzzlejar $**

**si** ( imatge.totcorrecte **i** dfil <= imatge.files / **2 i** dcol <= imatge.columnes / **2 i** MINIMDIVISIONS != **1** ) **fer**

**$ Per fer-ho transformaré la imatge en peces de imatge.columnes / dcol x imatge.files / dfil píxels de $**

**$ resolució. A cadascuna de aquestes peces li correspon un nombre, a la primera (dalt a la esquerra) el 0 $**

**$ i a l'última (sota a la dreta) el valor de dcol \* dfil - 1. Intercambiare els píxels de cada peça en ordre $**

**$ ([0, dfil \* dcol)) amb una altra peça escollida aleatoriament. $**

**per** ( posicio := **0** ; posicio < dfil \* dcol; posicio := posicio + **1** ) **fer**

aleatori := aleatori() mod ( dfil \* dcol ) ;

**mentre** ( aleatori mod ( mfil \* dcol ) = posicio ) **fer**

aleatori := aleatori() mod ( dfil \* dcol ) ;

**fmentre**

a := imatge.files / dfil \* ( posicio div dcol ) ; b := imatge.files / dfil \* ( aleatori div dcol ) ;

**mentre** ( a < imatge.files / dfil \* ( posicio div dcol + **1** ) **fer**

c := imatge.columnes / dcol \* ( posicio mod dcol ) ; d := imatge.columnes / dcol \* ( aleatori mod dcol ) ;

**mentre** ( c < imatge.columnes / dcol \* (posicio mod dcol + **1** ) **fer**

aux := imatge.matriuimatge[a][b]; imatge.matriuimatge[a][b] := imatge.matriuimatge[c][d]; imatge.matriuimatge[c][d] := aux;

c := c + **1** ; d := d + **1** ;

**fmentre**

a := a + **1** ; b := b + **1** ;

**fmentre**

**fsi**

**retorna** ( imatge.totcorrecte **i** dfil <= imatge.files / **2 i** dcol <= imatge.columnes / **2 i** MINIMDIVISIONS != **1** ) ;

**fi**

**ffunció**

**algorisme** Pràctica2 : "qüestió d'imatge" **és var**

nomarxiu, nompgmbase, pathfinal: taula[CADENAMAX] **de caràcters** ; pathrelatiuimatges, pathrelatiuimatgesMOD: **taula[] de caràcters** ; imatge: imatgepgm\_t;

opcio, llindar, n, dfil, dcol: **enter** ;

procedimentcorrecte, primeratriada: **booleà** ; mitjana: **real** ;

**fvar inici**

**$ inicialitzar $**

pathrelatiuimatges := ".\\imatges\\"; pathrelatiuimatgesMOD := ".\\imatgesModificades"; imatge.totcorrecte := **fals** ;

primeratriada := **fals** ; nomarxiu[0] := '\0'; ini\_llavor();

n := **0** ;

**$ codi $**

**escriure** ( "Benvingut a PGM EDITOR!" ) ;

**fer**

**si** ( primeratriada ) **fer**

**si** ( opcionestriades[0] != CENTINELLA ) **fer**

**escriure** ("Opcions introduïdes fins al moment: ", opcionestriades[0] ) ; n := **1** ;

**mentre** ( opcionestriades[n] != CENTINELLA **i** n < MAXOPCIONS ) **fer escriure** ( ", " opcionestriades[n] ) ;

n := n + **1** ;

**fper**

**si** ( n = MAXOPCIONS ) **fer**

n := MAXOPCIONS - **1** ;

**fsi**

**$ si nomarxiu s'ha actualitzat nomarxiu[0] != '\0' $**

**si** ( nomarxiu[0] != '\0' ) **fer**

**escriure** ( ". Imatge base actual: ", nompgmbase, "." ) ;

**si** ( comprovar\_nom\_arxiu\_correcte(nompgmbase) ) **fer**

**escriure** ( " Dimensions actuals: ", imatge.columnes,"x", imatge.files );

**fsi sino**

**escriure** ( ". NO hi ha cop imatge carregada." ) ;

**fsi sino**

**escriure** ( "Encara no has triat cap opcio valida" ) ;

**fsi fsi**

escriureopcions();

**escriure** ( "Quina opcio vols triar? " ); opcio := **0** ;

**llegir** ( opcio ) ;

**opció** ( opcio ) **fer**

**cas 1** :

**escriure** ( "Has triat carregar imatge" ) ; demanar\_nom\_arxiu(nomarxiu);

combinarpath(nomarxiu, pathrelatiuimatges, pathfinal); procedimentcorrecte := carrega\_pgm(pathfinal, imatge);

**si** ( procedimentcorrecte ) **fer**

**escriure** ( "S'ha carregat correctament l'arxiu ", nomarxiu," del directori ", pathrelatiuimatges ) ; combinarpath(nomarxiu, "", nompgmbase);

**sino**

**escriure** ( "Algo ha sortit malament, potser l'arxiu no existeix o esta buit o parcialment buit" ) ; combinarpath("La carregar ha sortit malament", "", nompgmbase);

**fsi cas 2** :

**escriure** ( "Has triat desar imatge" ) ; demanar\_nom\_arxiu(nomarxiu);

combinarpath(nomarxiu, pathrelatiuimatges, pathfinal); procedimentcorrecte := desar\_pgm(pathfinal, imatge); **si** ( procedimentcorrecte ) **fer**

**escriure** ( "S'ha desat correctament l'arxiu ", nomarxiu," del directori ", pathrelatiuimatgesMOD ) ;

**sino**

**escriure** ( "Algo ha sortit malament, has carregat una imatge abans?" ) ;

**fsi cas 3** :

**escriure** ( "Has triat mostrat mitjana imatge" ) ; mitjana = mitjana\_pixels(imatge);

**si** ( mitjana != **-1** ) **fer**

**escriure** ( "La imatge carregada té una mitjana de píxels de ", mitjana ) ;

**sino**

**escriure** ( "Algo ha sortit malament, has carregat una imatge abans?" ) ;

**fsi cas 4** :

**escriure** ( "Has triat binaritzar imatge" ) ;

**escriure** ( "Quin es el valor llindar pel cual vols binaritzar? " ) ;

**llegir** ( llindar ) ;

procedimentcorrecte := binaritza\_imatge(imatge, llindar);

**si** ( procedimentcorrecte ) **fer**

**escriure** ( "S'ha binaritzat correctament l'imatge." ) ;

**sino**

**escriure** ( "Algo ha sortit malament, has carregat una imatge abans? Tambe pot ser que el valor llindar que no sigui correcte." ) ;

**fsi cas 5** :

**escriure** ( "Has triat negativitzar imatge" ) ; procedimentcorrecte := negativa\_imatge(imatge); **si** ( procedimentcorrecte ) **fer**

**escriure** ( "La imatge ha sigut negativitzada." ) ;

**sino**

**escriure** ( "Algo ha sortit malament, has carregat una imatge abans?" ) ;

**fsi cas 6** :

**escriure** ( "Has triat reduir nivells de l'imatge" ) ;

**escriure** ( "Per quin valor vols reduir els nivells de l'imatge? " ) ;

**llegir** ( llindar ) ;

procedimentcorrecte := redueix\_nivells(imatge, llindar);

**si** ( procedimentcorrecte ) **fer**

**escriure** ( "S'ha reduit correctament l'imatge." ) ;

**sino**

**escriure** ( "Algo ha sortit malament, has carregat una imatge abans? Tambe pot ser que el valor no sigui correcte ha de ser divisor de 256." ) ;

**fsi cas 7** :

**escriure** ( "Has triat capgirar imatge" ) ; procedimentcorrecte := capgira\_imatge(imatge); **si** ( procedimentcorrecte ) **fer**

**escriure** ( "La imatge ha sigut capgirada." ) ;

**sino**

**escriure** ( "Algo ha sortit malament, has carregat una imatge abans?" ) ;

**fsi cas 8** :

**escriure** ( "Has triat mirallar imatge" ) ; procedimentcorrecte := mirall\_imatge(imatge); **si** ( procedimentcorrecte ) **fer**

**escriure** ( "La imatge ha sigut mirallada." ) ;

**sino**

**escriure** ( "Algo ha sortit malament, has carregat una imatge abans?" ) ;

**fsi cas 9** :

**escriure** ( "Has triat aplastar l'imatge" ) ;

**escriure** ( "Per quin ratio vols aplastar l'imatge? " ) ;

**llegir** ( llindar ) ;

procedimentcorrecte := aplasta\_imatge(imatge, llindar);

**si** ( procedimentcorrecte ) **fer**

**escriure** ( "La imatge carregada s'ha aixafat i te una altura final de ", imatge.files ) ;

**sino**

**escriure** ("Algo ha sortit malament, has carregat una imatge abans? Tambe pot ser que el ratio escollit no estigui dins del rang ((0, ", imatge.files,"]).") ;

**fsi**

**cas 10** :

**escriure** ( "Has triat puzle de la imatge carregada" ) ; procedimentcorrecte := puzle\_imatge(imatge, dfil, dcol); **si** ( procedimentcorrecte ) **fer**

**escriure** ("La imatge carregada ha sigut puzlejada i te ", dfil \* dcol, " peces.") ;

**sino**

**escriure** ("Algo ha sortit malament, t'has asegurat de carregar una imatge abans? Tambe pot ser que alguna dimensio de la imatge sigui un nombre primer.") ;

**escriure** ("O que el minim de divisions sigui més gran que un divisor d'alguna dimensio de la

imatge.") ;

**fsi**

**cas 0** :

**escriure** ("Sortint de PGM EDITOR...") ;

**altre cas** :

**escriure** ( opcio," no és una opció valida.") ; opcio := CENTINELLA;

**fopció**

primeratriada := **cert** ;

**si** ( opcio != **0** ) **fer**

opcionstriades[n] := opcio;

**si** ( opcio != CENTINELLA **i** n < MAXOPCIONS - **1** ) **fer**

opcionstriades[n + 1] := CENTINELLA

**fsi fsi**

**mentre** ( opcio != **0** ) ;

**fi**

**falgorisme**

# Joc de proves

## Procediments base:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Condicions inicials | Resultat esperat | Resultat real | Check |
| S’introdueix un caràcter. | Sortir del programa. |  |  |
| Carregar totes les imatges i mostrar les mitjanes de cadascuna. | Mitjanes correctes. |  |  |
| Desar una imatge amb el nom tomate.pgm. | Arxiu .pgm amb el nom tomate guardat a imatgesModificades. |  |  |
| Binaritzar imatge carregada per un valor llindar de 125. | Imatge bicolor on tots els píxels valen o 0 o 255. |  |  |
| Negativitzar imatge carregada. | Imatge negativa. |  |  |
| Reduir nivells de la imatge carregada per 8. | Canviar tots els píxels de la imatge fins a tenir un total de 8 tonalitats.  {0, 32, 64, 96, 128, 160,  192, 224} |  |  |

## Procediments extres:



|  |  |  |  |
| --- | --- | --- | --- |
| Condicions inicials | Resultat esperat | Resultat real | Check |
| Capgirar imatge carregada. | Imatge boca avall. |  |  |
| *Mirallar* imatge carregada. | Imatge vista des de un mirall. |  |  |
| Aixafar la imatge carregada per un rati de 8. | Baboon.pgm és de 256x256 per tant esperem una resolució final de 256x32. | Alçada de baboon inicial (al word) 2,8cm, alçada final 0,34 cm. 2,8/0,34 = 8. Per tant, s’ha aixafat per un rati de 8. |  |
| Aixafar la imatge carregada per un rati de 500. | Baboon.pgm és de 256x256 Així doncs, si l’aixafem per una ràtio de 500  significaria que l’alçada final seria de 0 per tant, no s’aixafa (per excedir el rang). | Zoom: |  |
| *Puzlejar* imatge “lenna.pgm” amb MINIMDIVISIO NS := 3. | *Puzlejar* imatge amb un mínim de peces de 9 (3\*3). | 16 peces és més gran que el mínim teòric de 9 peces. |  |
| *Puzlejar* imatge “velas.pgm” amb MINIMDIVISIO NS := 3. | Error, ja que les dimensions de “velas.pgm” és de 256x211  i 211 és un nombre prim | Zoom: |  |

## Procediments en profunditat:



**carregar\_pgm()**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Condicions inicials | Resultat esperat | Resultat real | | Check |
| Arxiu buit. | Error. | Zoom: | |  |
| Arxiu parcialment buit. | Error. | Zoom: | |  |
| Primera línia amb més informació que el format. | Error. |  | |  |
|  | Zoom: |
|  | |
| No existeix el directori on es desen. | Error. |  | |  |

**desar\_pgm()**

|  |  |  |  |
| --- | --- | --- | --- |
| Condicions inicials | Resultat esperat | Resultat real | Check |
| No existeix el directori on es desen. | Error. |  |  |
| Desar en format correcte la imatge carregada amb el nom probaC. | A la primera línia “P2”, a la segona les dimensions, a la tercera el valor màxim del píxel. A la resta els píxels. |  |  |
| Provar de escriure un nom sense .pgm. | Bucle infinit ja que no ho permet-ho. |  |  |

**mitjana\_pixels()**

|  |  |  |  |
| --- | --- | --- | --- |
| Condicions inicials | Resultat esperat | Resultat real | Check |
| Mitjana de proba.pgm | 255 + 255  4  = 127.5 |  |  |
| Mitjana de parcialmentbuit.pgm | Error. La imatge no és correcta. |  |  |

**binaritza\_imatge()**

|  |  |  |  |
| --- | --- | --- | --- |
| Condicions inicials | Resultat esperat | Resultat real | Check |
| Binaritzar proba.pgm per 150. La guardaré en probabin.pgm | 255 255  0 0 |  |  |
| Binaritzar proba.pgm per 0 i per 256. | Error. El valor llindar està fora del rang. |  |  |

**negativa\_imatge()**

|  |  |  |  |
| --- | --- | --- | --- |
| Condicions inicials | Resultat esperat | Resultat real | Check |
| Negativitzar proba.pgm.  La guardaré en probaneg.pgm | 105 0 155 106 |  |  |
| Negativitzar parcialmentbuit.pgm | Error. La imatge no és correcta. |  |  |

**redueix\_nivells()**

|  |  |  |  |
| --- | --- | --- | --- |
| Condicions inicials | Resultat esperat | Resultat real | Check |
| Reduir nivells proba.pgm per 4. La guardaré en probaredu.pgm. | 128 192  64 128 |  |  |
| Reduir nivells de proba.pgm per un valor NO divisor de 256. | Error. El valor no és divisor. |  |  |
| Reduir nivells proba.pgm per 1. La guardaré en probaredu.pgm. | La imatge quedarà en negre, ja que reduir la imatge de 256 tonalitats a 1, fa que l’únic valor possible sigui 0.  [0, 256/1[ -> 0 |  |  |
| Reduir parcialmentbuit.pgm per 4. | Error. La imatge no és correcta |  |  |

**capgira\_imatge() i miralla\_imatge()**

|  |  |  |  |
| --- | --- | --- | --- |
| Condicions inicials | Resultat  esperat | Resultat real | Check |
| Capgirar proba.pgm.  La guardaré en probacap.pgm. | 100 149  150 255 |  |  |
| Mirallar proba.pgm.  La guardaré en probamir.pgm. | 255 150  149 100 |  |  |
| Capgirar o mirallar parcialmentbuit.pgm | Error. La imatge no és correcta. |  |  |

**aixafa\_imatge()**

|  |  |  |  |
| --- | --- | --- | --- |
| Condicions inicials | Resultat esperat | Resultat real | Check |
| Aixafar proba.pgm per 2.  La guardaré en probaaix.pgm. | P2 2 1  150 255 |  |  |
| Aixafar proba.pgm per 0, 1, o 3. | Error. Està fora del rang  ]1, imatge.files] |  |  |
| Aixafar parcialmentbuit.pgm | Error. La imatge no és correcta. |  |  |

**puzle\_imatge()**



|  |  |  |  |
| --- | --- | --- | --- |
| Condicions inicials | Resultat esperat | Resultat real | Check |
| Puzlejar proba.pgm. | Error. 2 és un nombre prim. | Zoom: |  |
| Puzlejar baboon.pgm amb un MINIMDIVISIONS de 129. | Error. Està fora del rang  ]1,  min(imatge.files, imatge.columnes  )/2] ->  ]1, 128] | “O que, MINIMDIVISIONS (129) sigui 1 o més gran que tots els divisors d’alguna dimensió de la imatge excloent-hi la dimensio mateixa.” |  |
| Puzlejar parcialmentbuit.pgm | Error. La imatge no és correcta. |  |  |

# Conclusions

La dificultat de les pràctiques en general, en el meu cas, és preparar la documentació, trigo molt més en explicar que en fer la pràctica. Em costa explicar en forma escrita, és una limitació de les persones que ens agraden més les matemàtiques. És una de les coses que crec que he de superar-me, donat que m’agradaria ser en un futur professor.

Concretament amb la pràctica, no he tingut cap problema, fins i tot, la vaig acabar el mateix dia.

Donat que vaig fer procediment extres, he tingut un problema per fer la funció puzle\_imatge. Havia de relacionar cada peça amb els seu corresponent número amb la posició del seu píxel inicial i final. Esbrinant, i pensant vaig trobar la fórmula que relaciona el número de la peça amb les seves files inicials inicial (imatge.files

/ dfil \* (posició div dcol)) i finals ((imatge.files / dfil \* (posició div dcol + 1) - 1). Poc després vaig trobar la que relacionava el número de la peça amb les seves columnes inicials (imatge.columnes / dcol \* (posició mod dcol)) i finals ((imatge->columnes / dcol \* (posició mod dcol + 1)) - 1).

Per finalitzar, cal destacar que m’ha agradat molt la pràctica i m’ha resultat molt interessant.

# Checklist

|  |  |
| --- | --- |
| Requisits | Check |
| 1. En l'execució del programa no es queda penjat i realitza les accions demandes. |  |
| 2. El codi font és correcte (no genera ni errors ni advertències). |  |
| 3. Tant el pseudocodi com codi font són llegibles (estan indentats i amb comentaris). |  |
| 4. Els noms de les variables i constants segueixen les convencions de l’assignatura. |  |
| 5. El document no té errades greus d’ortografia / expressió. |  |
| 6. El document respecta les extensions màximes. |  |
| 7. Es compleix el que diu a l'apartat "A tenir en compte" de document "De pseudocodi a C"); |  |

1. En un tauler de mxn la cantonada que podria tenir mes valors serà primer, la que sigui més llarga i una disposició de 10101... Tenint en compte això maxCombinations = (max(m, n) + max(m, n) % 2)/2 + 1. Per exemple: M, N = 5 -> 10101 -> (5+1)/2 || M, N = 4 ->1010 -> 4/2. Afegim un 1 per saber quan acaba aquella cantonada guardant un valor sentinella com a final. [↑](#footnote-ref-1)