07/12/23 Malloc allone dynamiquement la mémoire d'une taille donnée.

P[7] = *(P+7); int *p1 = malloc(50 * sizeof(int))
P[0] == *P *p = p1[1]

P+1 P+2 P+3 P+4
p -> P[0] P[1] P[2] P[3] a=2; *b=&a; **c=&b;

**p = malloc(50 * sizeof(int*)); Case des lignes contient plusieurs colonne
for(int i=0; i<50; ++i){ le nombres de lignes
  p[i] = malloc(20 * sizeof); le nombres de colonnes
} a = P[5][8]
  *P = adresse colonne **P = valeurs

Sanitizer, -g

14/11/23 File *fp = fopen(filename, mode); filename = string, chemin (si invalide pas retour null)
mode = "r", "w";
Check si fichier est ok
fprintf(stderr, "Can't open output file %s!\n", filename);
↑ destination

dernier fermer EX: FILE *fp = fopen("mon_fichier", "w");
fclose(fp); -> garantit l'écriture en vidant la mémoire toujours

File *fp = fopen("test.txt", "r");
char buffer[100]; -> malloc, calloc (sauf pour remplacer type de fichier)
fgets(buffer, 37, fp);
fclose(fp);

bytes

fread(void *ptr, size, nmemb, FILE *stream)

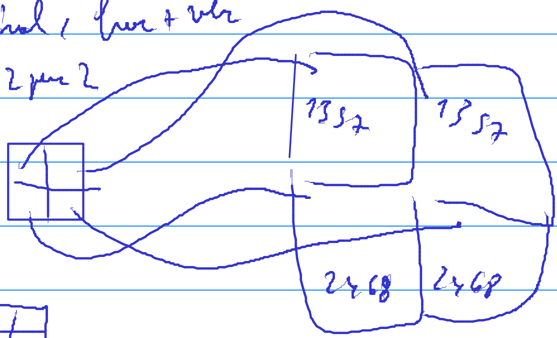
```
FILE *fp = fopen("doubles.bin", "rb");
double buffer[100];
size_t num = fread(buffer, sizeof(double), 4, fp);
 fclose(fp);
```

```
int fprintf(FILE *stream, const char *format, ...);
} FILE * : stdin
           stdout
           stderr
```

---

pgm-result

1. lire l'image pgm binaire (à vérifier) l;c ; nbcolor (max, min); niveaux de gris
2.
3. inverser les valeurs, 255 - val pixel
4. Symétrie : hori, vertical, hori + vert
5. Pixelisation : 2x2, on avance 2 par 2 ligne et colonne



5. Rogner
6. Convolution :

7	9	12
2	40	21
1	0	21

→
