

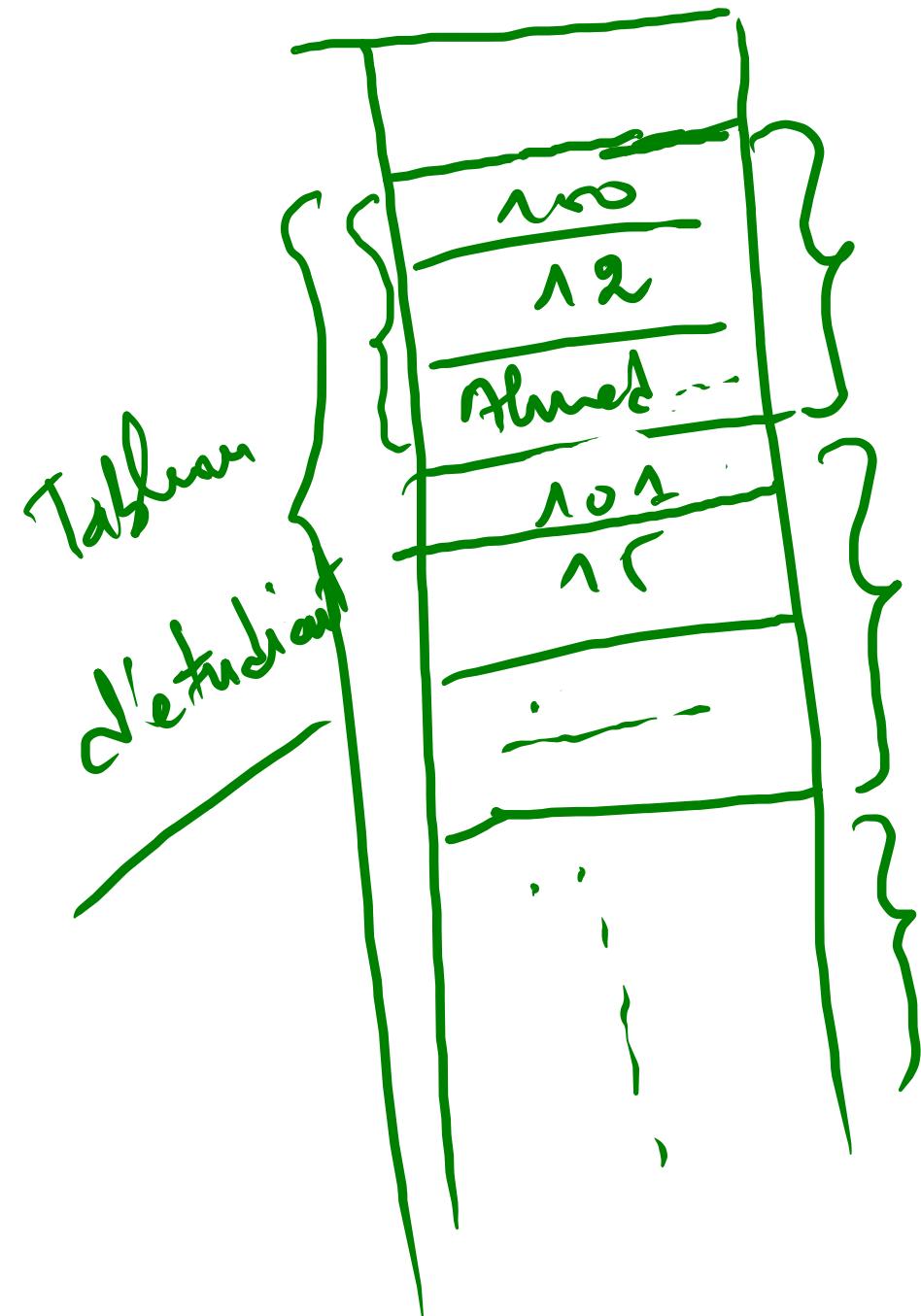
- Solution qui m'a proposé :

→ {Code [0..Max] : entier
 → {note [0 .. Max] : réel

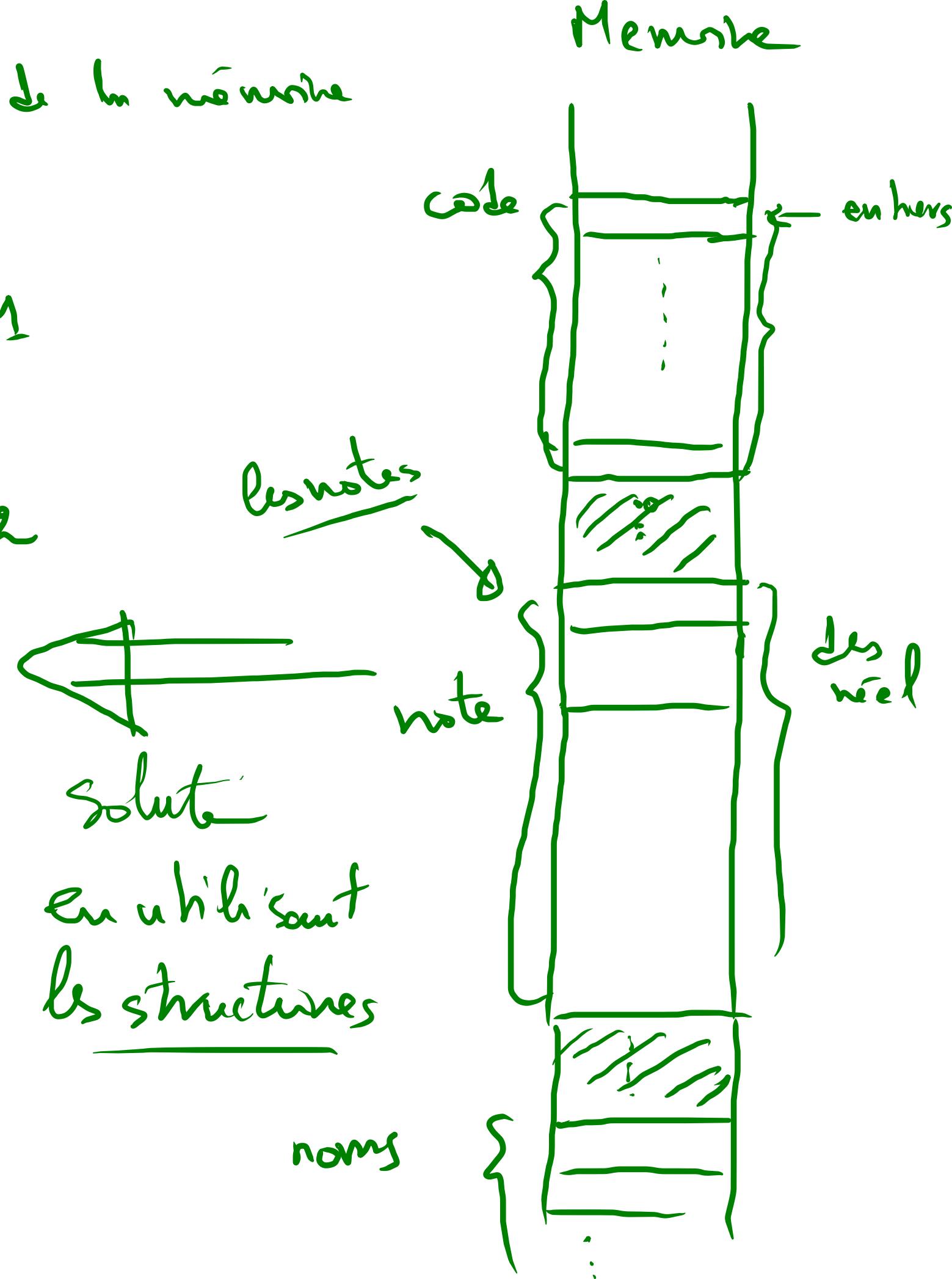
- utiliser en parallèle des deux tableaux

| toutes nos fonctions utilisent à la fois les deux tableaux
 - recherche, tri

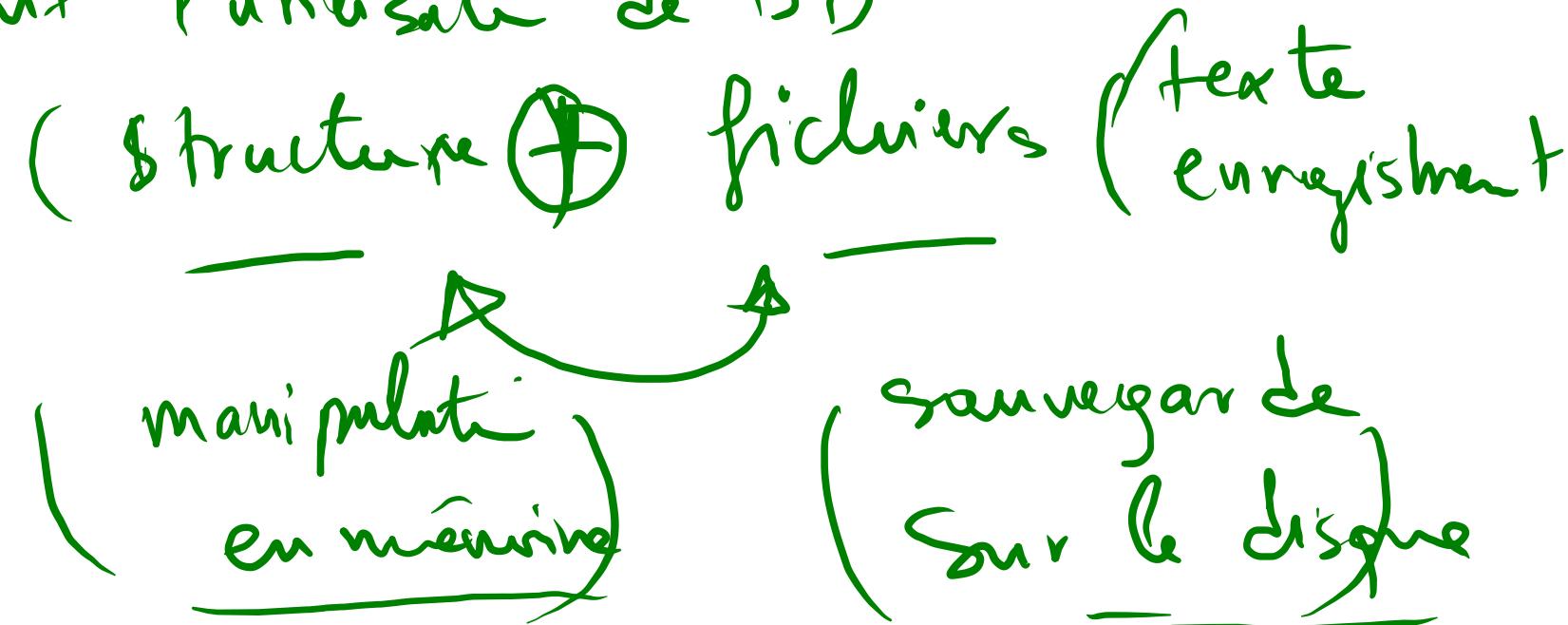
} cas de concepte
 d'app.
 info utiles



au niveau de la mémoire



avant l'utilisation de BD



Syntaxe:

```
type NomDuType = structure  
    ch1 : type1
```

```
    ch2 : type2
```

```
fin structure
```

e : etudiant ; une variable
{ e.code = 100; de type etudiant

e.note = 12;
 e.nom = "Ahmed -"

etudiant.dat

classe.dat

machine.dat

- - - .txt

exemples ↴

type etudiant = structure

code : entier

note : réel

nom : chaîne

fin structure

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
- Soustraction
- Division
- Multiplication
- affichage
- Simplification

Déclaration de la structure

```
type fraction = structure  
    n : entier  
    d : entier
```

fin structure

f₁ : fraction
f₁.n ← 5
f₁.d ← 2) 5/2
affichage

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
- Soustraction
- Division
- Multiplication
- affichage
- Simplification

Déclaration de la structure

```
typedef struct {  
    int n;  
    int d;  
} fraction;
```

fraction f1;
 $f1.n = 5;)$ 5/2
 $f1.d = 2;)$

affichage

5/2

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
- Soustraction
- Division
- Multiplication
- affichage
- Simplification

Déclaration de la structure

```
type fraction = structure  
    n : entier  
    d : entier
```

fin structure

n/d

```
procédure affichage (f1 : fraction)
```

```
procédure simplification (e/s f : fraction)
```

fonction simplification2 (e f : fraction) : fraction

fraction somme (E f1, f2 : fraction) :

fraction

fraction soustraire (E f1, f2 : fraction) :

fraction

division et multiplication

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
- Soustraction
- Division
- Multiplication
- affichage
- Simplification

procédure affichage (f : fraction)
Debut
| ecrive (f.n, ' / ', f.d);
fin

en C :

void affichage (fraction f) {

} printf (" %d / %d ", f.n, f.d);
}

f.n = 2
f.d = 5
nxd ↓
[2/5]

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
- Soustraction
- Division
- Multiplication
- affichage
- Simplification

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + cb}{bd}$$

Fonction Somme ($E : F_1, F_2 : \text{Fraction}$) : Fraction

$S : \text{Fraction}$

Début

$$S.n \leftarrow F_1.n * F_2.d + F_2.n * F_1.d$$

$$S.d \leftarrow F_1.d * F_2.d$$

Retourner S;

Fin

Utilisati

$f_1, f_2 : \text{frac}$

$f_1.n \leftarrow 2; f_1.d \leftarrow 5; // \frac{2}{5}$

$f_2.n \leftarrow 1; f_2.d \leftarrow 3; // \frac{1}{3}$

affichage (f_1); affichage (f_2);

$$\frac{2}{5} + \frac{1}{3} = \frac{2 \cdot 3 + 1 \cdot 5}{5 \cdot 3} = \frac{11}{15}$$

$s_1 : \text{frac}$ f_2, f_1
 $s_1 \leftarrow \text{somme}(f_1, f_2);$
 $\text{affichage}(s_1); // \frac{11}{15}$

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
- Soustraction
- Division
- Multiplication
- affichage
- Simplification

```
fraction somme(frac1, frac2) {  
    fraction s;  
    s.n = f1.n * f2.d + f1.d * f2.n;  
    s.d = f1.d * f2.d;  
    return s;  
}
```

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
- Soustraction
- Division
- Multiplication
- affichage
- Simplification

fonction soustraction (f_1, f_2 : fraction) : fraction

s : fraction

début

$$s.n \leftarrow f_1.n * f_2.d - f_2.n * f_1.d$$

$$s.d \leftarrow f_1.d * f_2.d$$

return s

Fin

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
- Soustraction
- Division
- Multiplication
- affichage
- Simplification

en C°

```
fracti soustraction ( fracti f1, fracti f2 ) {  
    fraction s ;  
    S.n = f1.n * f2.d - (f2.n * f1.d) ;  
    S.d = f1.d * f2.d ;  
    return S ;  
}
```

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
 - Soustraction
 - Division
 - Multiplication
 - affichage
 - Simplification

```

fonction Division (E f1, f2 : fraction) : fraction
    résultat : fraction
    début
        résultat . m ← f1 . m * f2 . d
        résultat . d ← f1 . d * f2 . m
    retourner résultat
fin

```

function Multiplication (E. f₁·f₂ : fraction) : fraction

H : fraction

Jébut

$$n \leftarrow f_1 \circ n + f_2 \circ n$$

$$n \cdot d \leftarrow f_1 \cdot d * f_2 \cdot e$$

reformer M

10

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
- Soustraction
- - Division
- Multiplication
- affichage
- Simplification

```
fract division (fract f1, fract f2) {  
    fract resultat;  
    resultat.n = f1.n * f2.d ;  
    resultat.d = f1.d * f2.n ;  
    return resultat;  
}  
  
fract multiply (---) {  
    fract M;  
    M.n = f1.n * f2.n ;  
    M.d = f1.d * f2.d ;  
    return M;  
}
```

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
- Soustraction
- Division
- Multiplication
- affichage
- Simplification



Fonction Simplification(E: F: Fraction):Fraction

Diviseur : Entier

S: Fraction

Début

Diviseur $\leftarrow \text{pgcd}(F.n, F.d);$

S.n $\leftarrow F.n / \text{Diviseur};$

S.d $\leftarrow F.d / \text{Diviseur};$

Retourner S;

Fin

on considère la fonction :

fonction pgcd(E m, n : entier) : entier

p : entier

Début

: ?

retourner p

fin

appel :

d $\leftarrow \text{pgcd}(3, 24)$, // $\Rightarrow d = 3$

d $\leftarrow \text{pgcd}(15, 3)$; $\Rightarrow d = 3$

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

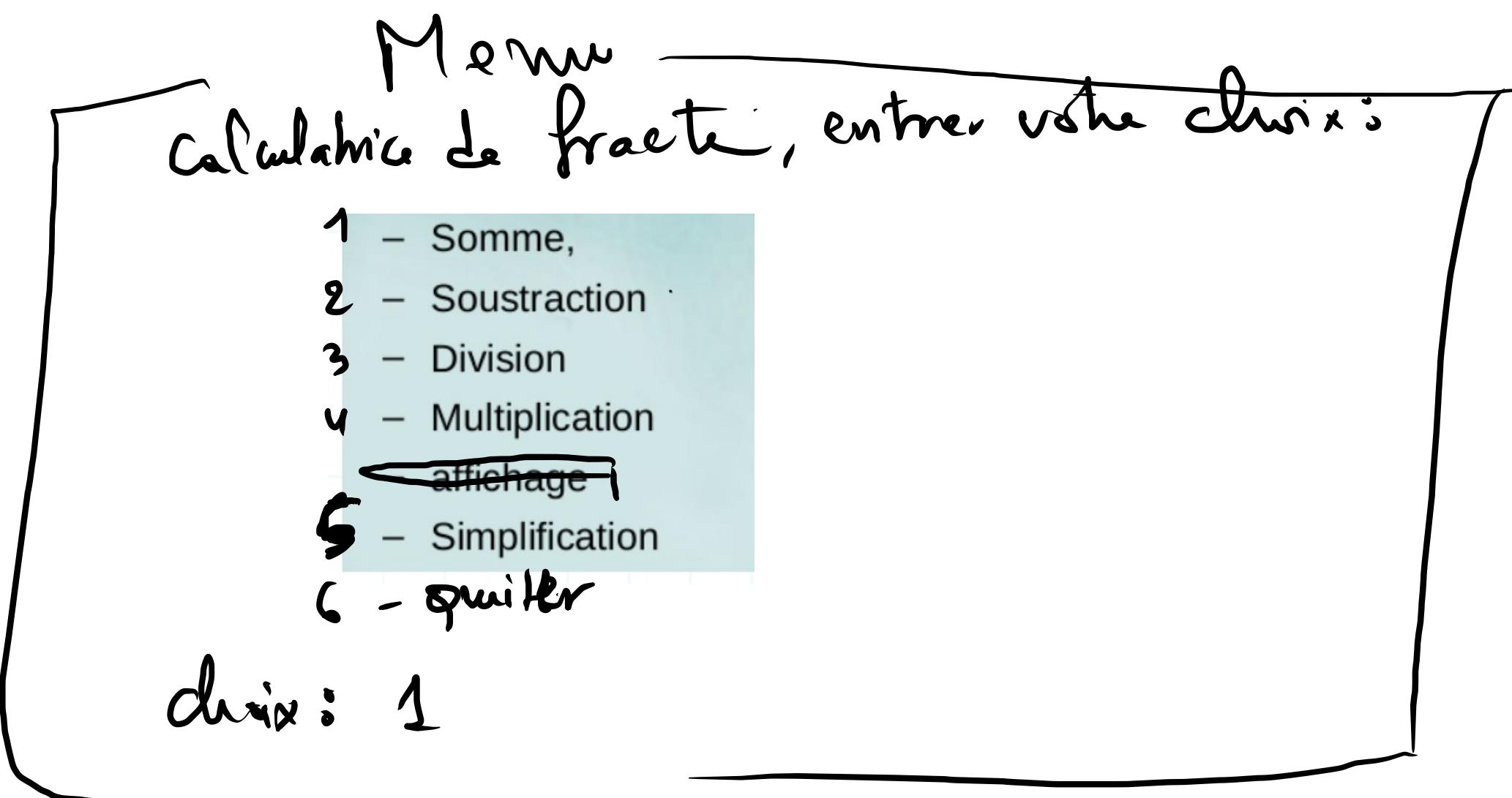
- Somme,
- Soustraction
- Division
- Multiplication
- affichage
- Simplification



```
fracte simplificate (fracte f) {  
    int diviseur = pgcd(f.n, f.d); //  
    fracte r;  
    r.n = f.n / diviseur;  
    r.d = f.d / diviseur;  
    return r;  
}  
f = simplificate(some(f1, f2));  
f = simplificate(some(f1, f2));  
f1 = simplificate(f);  
f2 = simplificate(f);
```

Ecrire un algorithme puis un programme qui permet de manipuler les fractions, on doit alors développer les fonctions suivantes:

- Somme,
- Soustraction
- Division
- Multiplication
- affichage
- Simplification



procédure Menu ()

choix : entier

f_1, f_2 : fraction

debut

{ répéter
 ecrив ("Calculatrice des fractions");
 ecrив ("1 - somme");

 |
 ecrив ("6 - quitter");
 ecrив ("Voulez vous continuer ?");

 line (choix);

cas où choix :

 cas 1 : lecture(f_1); lecture(f_2);
 ecrив ("la somme est :"); affichage(somme(f_1, f_2));

 cas 2 : lecture(f_1); lecture(f_2);
 ecrив ("la soustraction est :"); affichage(soustraction(f_1, f_2))

Jusqu'à ce que (choix = 6)

fin

```
fracte    lecturefracte () {  
    fracte f;  
    printf ("(n/d) enter n: ");  
    scanf ("%d", &f.n);  
    printf ("(%d/d) enter d: ", f.n);  
    scanf ("%d", &f.d);  
    return f;  
}
```

$f_n = \text{lecturefracte}();$

$f_r = \text{lecturefracte}();$

```
void main() {  
    fraction f1, f2;  
    f1 = lecturefraction();  
    f2 = lecturefraction();  
    printf("ln somme : \n");  
    affichage(somme(f1, f2));  
    printf("ln soustraction : \n");  
    affichage(soustraction(f1, f2));  
    printf("ln multiblent ln");  
    affichage(multiplication(f1, f2));  
}
```