Introduction aux algorithmes

Algorithmique et structures de données, 2022-2023

P. Albuquerque (B410), P. Künzli et O. Malaspinas (A401), ISC, HEPIA

2022-10-19

En partie inspirés des supports de cours de P. Albuquerque

Rappel

Quel est l'algorithme du tri par sélection?

Rappel

Quel est l'algorithme du tri par sélection?

1. Soit un tableau d'entiers, `tab[0:SIZE-1]` et `i=0`.
2. Trouver l'indice, `j`, de `tab[i:SIZE-2]` où la valeur est minimale.
3. Échanger `tab[i]` et `tab[j]`.
4. `i+=1` et revenir à 2, tant que `j < SIZE-2`.

Un type de tableau particulier

Les chaînes de caractères

```
string = tableau + char + magie noire
```

Le type `char`

- Le type `char` est utilisé pour représenter un caractère.
- C'est un entier 8 bits signé.
- En particulier:
  - Écrire

    ```
    char c = 'A';
    ```

  - Est équivalent à:

    ```
    char c = 65;
    ```

- Les fonctions d'affichage interprètent le nombre comme sa valeur ASCII.

Chaînes de caractères (strings)

- Chaîne de caractère == tableau de caractères **terminé par la valeur** `'\0'` ou `0`.

Exemple

```
char *str  = "HELLO !";
char str[] = "HELLO !";
```

Est représenté par

char	H	E	L	L	O		!	\0
ASCII	72	69	76	76	79	32	33	0

Chaînes de caractères (strings)

- Chaîne de caractère == tableau de caractères **terminé par la valeur** `'\0'` ou `0`.

Exemple

```
char *str  = "HELLO !";
char str[] = "HELLO !";
```

Est représenté par

char	H	E	L	L	O		!	\0
ASCII	72	69	76	76	79	32	33	0

A quoi sert le `\0`?

Chaînes de caractères (strings)

- Chaîne de caractère == tableau de caractères **terminé par la valeur** `'\0'` ou `0`.

Exemple

```
char *str  = "HELLO !";
char str[] = "HELLO !";
```

Est représenté par

char	H	E	L	L	O		!	\0
ASCII	72	69	76	76	79	32	33	0

A quoi sert le `\0`?

Permet de connaître la fin de la chaîne de caractères (pas le cas des autres sortes de tableaux).

Syntaxe

```
char name[5];
name[0] = 'P';  // = 70;
name[1] = 'a';  // = 97;
name[2] = 'u';  // = 117;
name[3] = 'l';  // = 108;
name[4] = '\0'; // = 0;
char name[] = {'P', 'a', 'u', 'l', '\0'};
char name[5] = "Paul";
char name[] = "Paul";
char name[100] = "Paul is not 100 characters long.";
```

Fonctions

- Il existe une grande quantités de fonction pour la manipulation de chaînes de caractères dans `string.h`.
- Fonctions principales:

```c
// longueur de la chaîne (sans le \0)
size_t strlen(char *str);
// copie jusqu'à un \0
char *strcpy(char *dest, const char *src);
 // copie len char
char *strncpy(char *dest, const char *src, size_t len);
// compare len chars
int strncmp(char *str1, char *str2, size_t len);
// compare jusqu'à un \0
int strcmp(char *str1, char *str2);
```

- Pour avoir la liste complète: `man string`.

Fonctions

- Il existe une grande quantités de fonction pour la manipulation de chaînes de caractères dans `string.h`.
- Fonctions principales:

```c
// longueur de la chaîne (sans le \0)
size_t strlen(char *str);
// copie jusqu'à un \0
char *strcpy(char *dest, const char *src);
 // copie len char
char *strncpy(char *dest, const char *src, size_t len);
// compare len chars
int strncmp(char *str1, char *str2, size_t len);
// compare jusqu'à un \0
int strcmp(char *str1, char *str2);
```

- Pour avoir la liste complète: `man string`.

Quels problèmes peuvent se produire avec `strlen`, `strcpy`, `strcmp`?

Les anagrammes

Définition

Deux mots sont des anagrammes l'un de l'autre quand ils contiennent les mêmes lettres mais dans un ordre différent.

Exemple

t	u	t	u	t	\0
t	u	t	t	u	\0

Problème: Trouvez un algorithme pour déterminer si deux mots sont des anagrammes.

Les anagrammes

Il suffit de:

1. Trier les deux mots.
2. Vérifier s'ils contiennent les mêmes lettres.

Implémentation ensemble

```
int main() { // pseudo C
    tri(mot1);
    tri(mot2);
    if egalite(mot1, mot2) {
        // anagrammes
    } else {
        // pas anagrammes
    }
}
```

Les palindromes

Mot qui se lit pareil de droite à gauche que de gauche à droite:

Les palindromes

Mot qui se lit pareil de droite à gauche que de gauche à droite:

- rotor, kayak, ressasser, ...

Problème: proposer un algorithme pour détecter un palindrome

Les palindromes

Mot qui se lit pareil de droite à gauche que de gauche à droite:

- rotor, kayak, ressasser, ...

Problème: proposer un algorithme pour détecter un palindrome

Solution 1

```
while (first_index < last_index {
    if (mot[first_index] != mot [last_index]) {
        return false;
    }
    first_index += 1;
    last_index -= 1;
}
return true;
```

Les palindromes

Mot qui se lit pareil de droite à gauche que de gauche à droite:

- rotor, kayak, ressasser, ...

Problème: proposer un algorithme pour détecter un palindrome

Solution 1

```
while (first_index < last_index {
    if (mot[first_index] != mot [last_index]) {
        return false;
    }
    first_index += 1;
    last_index -= 1;
}
return true;
```

Solution 2

```
mot_tmp = revert(mot);
return mot == mot_tmp;
```

Crible d'Ératosthène

Algorithme de génération de nombres premiers.

Exercice

- À l'aide d'un tableau de booléens,
- Générer les nombres premiers plus petits qu'un nombre N

Pseudo-code

- Par groupe de trois, réfléchir à un algorithme.

Programme en C

- Implémenter l'algorithme et le poster sur le salon `Element`.

Crible d'Ératosthène: solution

```c
#include <stdio.h>
#include <stdbool.h>
#define SIZE 51
int main() {
   bool tab[SIZE];
   for (int i=0;i<SIZE;i++) {
      tab[i] = true;
   }
   for (int i = 2; i < SIZE; i++) {
      if (tab[i]) {
         printf("%d ", i);
         int j = i;
         while (j < SIZE) {
            j += i;
            tab[j] = false;
         }
      }
   }
   printf("\n");
}
```

Réusinage de code (refactoring)

Le réusinage est?

Réusinage de code (refactoring)

Le réusinage est?

- le processus de restructuration d'un programme:
  - en modifiant son design,
  - en modifiant sa structure,
  - en modifiant ses algorithmes
  - mais en **conservant ses fonctionalités**.

Réusinage de code (refactoring)

Le réusinage est?

- le processus de restructuration d'un programme:
  - en modifiant son design,
  - en modifiant sa structure,
  - en modifiant ses algorithmes
  - mais en **conservant ses fonctionalités**.

Avantages?

Réusinage de code (refactoring)

Le réusinage est?

- le processus de restructuration d'un programme:
  - en modifiant son design,
  - en modifiant sa structure,
  - en modifiant ses algorithmes
  - mais en **conservant ses fonctionalités**.

Avantages?

- Amélioration de la lisibilité,
- Amélioration de la maintenabilité,
- Réduction de la complexité.

Réusinage de code (refactoring)

Le réusinage est?

- le processus de restructuration d'un programme:
  - en modifiant son design,
  - en modifiant sa structure,
  - en modifiant ses algorithmes
  - mais en **conservant ses fonctionalités**.

Avantages?

- Amélioration de la lisibilité,
- Amélioration de la maintenabilité,
- Réduction de la complexité.

"Make it work, make it nice, make it fast", Kent Beck.

Réusinage de code (refactoring)

Le réusinage est?

- le processus de restructuration d'un programme:
  - en modifiant son design,
  - en modifiant sa structure,
  - en modifiant ses algorithmes
  - mais en **conservant ses fonctionalités**.

Avantages?

- Amélioration de la lisibilité,
- Amélioration de la maintenabilité,
- Réduction de la complexité.

"Make it work, make it nice, make it fast", Kent Beck.

Exercice:

- Réusiner le code se trouvant sur Cyberlearn.

Tableau à deux dimensions (1/4)

Mais qu'est-ce donc?

Tableau à deux dimensions (1/4)

Mais qu'est-ce donc?

- Un tableau où chaque cellule est un tableau.

Quels cas d'utilisation?

Tableau à deux dimensions (1/4)

Mais qu'est-ce donc?

- Un tableau où chaque cellule est un tableau.

Quels cas d'utilisation?

- Tableau à double entrée;
- Image;
- Écran (pixels);
- Matrice (mathématique);

Tableau à deux dimensions (2/4)

Exemple: tableau à 3 lignes et 4 colonnes d'entiers

indices	0	1	2	3
0	7	4	7	3
1	2	2	9	2
2	4	8	8	9

Syntaxe

```
int tab[3][4]; // déclaration d'un tableau 4x3
tab[2][1]; // accès à la case 2, 1
tab[2][1] = 14; // assignation de 14 à la position 2, 1
```

Tableau à deux dimensions (3/4)

Exercice: déclarer et initialiser aléatoirement un tableau `50x100`

Tableau à deux dimensions (3/4)

Exercice: déclarer et initialiser aléatoirement un tableau `50x100`

```
#define NX 50
#define NY 100
int tab[NX][NY];
for (int i = 0; i < NX; ++i) {
    for (int j = 0; j < NY; ++j) {
        tab[i][j] = rand() % 256; // 256 niveaux de gris
    }
}
```

Exercice: afficher le tableau

Tableau à deux dimensions (3/4)

Exercice: déclarer et initialiser aléatoirement un tableau 50x100

```
#define NX 50
#define NY 100
int tab[NX][NY];
for (int i = 0; i < NX; ++i) {
    for (int j = 0; j < NY; ++j) {
        tab[i][j] = rand() % 256; // 256 niveaux de gris
    }
}
```

Exercice: afficher le tableau

```
for (int i = 0; i < NX; ++i) {
    for (int j = 0; j < NY; ++j) {
        printf("%d ", tab[i][j]);
    }
    printf("\n");
}
```

Tableau à deux dimensions (4/4)

Attention

- Les éléments ne sont **jamais** initialisés.
- Les bornes ne sont **jamais** vérifiées.

```
int tab[3][2] = { {1, 2}, {3, 4}, {5, 6} };
printf("%d\n", tab[2][1]);  // affiche?
printf("%d\n", tab[10][9]); // affiche?
printf("%d\n", tab[3][1]);  // affiche?
```

La couverture de la reine

- Aux échecs la reine peut se déplacer horizontalement et verticalement
- Pour un échiquier `5x6`, elle *couvre* les cases comme ci-dessous

```
   0  1  2  3  4  5
0  *     *     *
1     *  *  *
2  *  *  R  *  *  *
3     *  *  *
4  *     *     *
```

Exercice

- En utilisant les conditions, les tableaux à deux dimensions, et des `char` uniquement.
- Implémenter un programme qui demande à l'utilisateur d'entrer les coordonnées de la reine et affiche un tableau comme ci-dessus pour un échiquier `8x8`.

Types énumérés (1/2)

- Un **type énuméré**: ensemble de *variantes* (valeurs constantes).
- En C les variantes sont des entiers numérotés à partir de 0.

```c
enum days {
    monday, tuesday, wednesday,
    thursday, friday, saturday, sunday
};
```

- On peut aussi donner des valeurs "custom"

```c
enum days {
    monday = 2, tuesday = 8, wednesday = -2,
    thursday = 1, friday = 3, saturday = 12, sunday = 9
};
```

- S'utilise comme un type standard et un entier

```c
enum days d = monday;
(d + 2) == tuesday + tuesday; // true
```

Types énumérés (2/2)

- Très utile dans les `switch ... case`

```
enum days d = monday;
switch (d) {
    case monday:
        // trucs
        break;
    case tuesday:
        printf("0 ou 1\n");
        break;
}
```

- Le compilateur vous prévient qu'il en manque!

Utilisation des types énumérés

Réusiner votre couverture de la reine avec des `enum`

Représentation des nombres (1/2)

- Le nombre 247.

Nombres décimaux: Les nombres en base 10

10^2	10^1	10^0
2	4	7

$$247 = 2 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0.$$

Représentation des nombres (2/2)

- Le nombre 11110111.

Nombres binaires: Les nombres en base 2

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	1	0	1	1	1

$$1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

Représentation des nombres (2/2)

- Le nombre 11110111.

Nombres binaires: Les nombres en base 2

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	1	0	1	1	1

$$1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

$$= 247.$$

Conversion de décimal à binaire (1/2)

Convertir 11 en binaire?

Conversion de décimal à binaire (1/2)

Convertir 11 en binaire?

- On décompose en puissances de 2 en partant de la plus grande possible

```
11 / 8 = 1, 11 % 8 = 3
3  / 4 = 0,  3 % 4 = 3
3  / 2 = 1,  3 % 2 = 1
1  / 1 = 1,  1 % 1 = 0
```

- On a donc

$$1011 \Rightarrow 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 11.$$

Conversion de décimal à binaire (2/2)

Convertir un nombre arbitraire en binaire: 247?

- Par groupe établir un algorithme.

Conversion de décimal à binaire (2/2)

Convertir un nombre arbitraire en binaire: 247?

- Par groupe établir un algorithme.

Algorithme

1. Initialisation

```
num = 247
while (2^N < num) {
    N += 1
}
```

Conversion de décimal à binaire (2/2)

Convertir un nombre arbitraire en binaire: 247?

- Par groupe établir un algorithme.

Algorithme

1. Initialisation

```
num = 247
while (2^N < num) {
    N += 1
}
```

2. Boucle

```
while (N >= 0) {
    bit = num / 2^N
    num = num % 2^N
    N += 1
}
```

Les additions en binaire

Que donne l'addition 1101 avec 0110?

- L'addition est la même que dans le système décimal

```
   1101          8+4+0+1 = 13
+  0110    +     0+4+2+0 =  6
-------    ----------------
  10011       16+0+0+2+1 = 19
```

- Les entiers sur un ordinateur ont une précision **fixée** (ici 4 bits).
- Que se passe-t-il donc ici?

Les additions en binaire

Que donne l'addition `1101` avec `0110`?

- L'addition est la même que dans le système décimal

```
   1101          8+4+0+1 = 13
+  0110    +     0+4+2+0 =  6
-------    -----------------
  10011       16+0+0+2+1 = 19
```

- Les entiers sur un ordinateur ont une précision **fixée** (ici 4 bits).
- Que se passe-t-il donc ici?

Dépassement de capacité: le nombre est "tronqué"

- `10011 (19) -> 0011 (3)`.
- On fait "le tour"."

Entier non-signés minimal/maximal

- Quel est l'entier non-signé maximal représentable avec 4 bit?

Entier non-signés minimal/maximal

- Quel est l'entier non-signé maximal représentable avec 4 bit?

$$(1111)_2 = 8 + 4 + 2 + 1 = 15$$

- Quel est l'entier non-signé minimal représentable avec 4 bit?

Entier non-signés minimal/maximal

- Quel est l'entier non-signé maximal représentable avec 4 bit?

$$(1111)_2 = 8 + 4 + 2 + 1 = 15$$

- Quel est l'entier non-signé minimal représentable avec 4 bit?

$$(0000)_2 = 0 + 0 + 0 + 0 = 0$$

- Quel est l'entier non-signé min/max représentable avec N bit?

Entier non-signés minimal/maximal

- Quel est l'entier non-signé maximal représentable avec 4 bit?

$$(1111)_2 = 8 + 4 + 2 + 1 = 15$$

- Quel est l'entier non-signé minimal représentable avec 4 bit?

$$(0000)_2 = 0 + 0 + 0 + 0 = 0$$

- Quel est l'entier non-signé min/max représentable avec N bit?

$$0 \text{ et } 2^N - 1.$$

- Donc `uint32_t`? maximal est?

Entier non-signés minimal/maximal

- Quel est l'entier non-signé maximal représentable avec 4 bit?

$$(1111)_2 = 8 + 4 + 2 + 1 = 15$$

- Quel est l'entier non-signé minimal représentable avec 4 bit?

$$(0000)_2 = 0 + 0 + 0 + 0 = 0$$

- Quel est l'entier non-signé min/max représentable avec N bit?

$$0 \text{ et } 2^N - 1.$$

- Donc `uint32_t`? maximal est?

$$4294967295$$

Les multiplications en binaire (1/2)

Que donne la multiplication de 1101 avec 0110?

- L'addition est la même que dans le système décimal

```
     1101                  13
*    0110     *             6
---------     --------------
     0000                  78
    11010
   110100
+ 0000000
---------     --------------
  1001110     64+0+0+8+4+2+0
```

Les multiplications en binaire (2/2)

Que fait la multiplication par 2?

Les multiplications en binaire (2/2)

Que fait la multiplication par 2?

- Décalage de un bit vers la gauche!

```
      0110
*     0010
---------
      0000
+    01100
---------
     01100
```

Les multiplications en binaire (2/2)

Que fait la multiplication par 2?

- Décalage de un bit vers la gauche!

```
      0110
*     0010
---------
      0000
+    01100
---------
     01100
```

Que fait la multiplication par 2^N?

Les multiplications en binaire (2/2)

Que fait la multiplication par 2?

- Décalage de un bit vers la gauche!

```
      0110
*     0010
---------
      0000
+    01100
---------
     01100
```

Que fait la multiplication par 2^N?

- Décalade de N bits vers la gauche!

Entiers signés (1/2)

Pas de nombres négatifs encore...

Comment faire?

Entiers signés (1/2)

Pas de nombres négatifs encore...

Comment faire?

Solution naïve:

- On ajoute un bit de signe (le bit de poids fort):

```
00000010: +2
10000010: -2
```

Problèmes?

Entiers signés (1/2)

Pas de nombres négatifs encore...

Comment faire?

Solution naïve:

- On ajoute un bit de signe (le bit de poids fort):

```
00000010: +2
10000010: -2
```

Problèmes?

- Il y a deux zéros (pas trop grave): 10000000 et 00000000
- Les additions différentes que pour les non-signés (très grave)

```
  00000010              2
+ 10000100          + -4
----------          ----
  10000110 = -6  !=    -2
```

Entiers signés (2/2)

Beaucoup mieux

- Complément à un:
  - on inverse tous les bits: `1001 => 0110`.

Encore un peu mieux

- Complément à deux:
  - on inverse tous les bits,
  - on ajoute 1 (on ignore les dépassements).

Entiers signés (2/2)

Beaucoup mieux

- Complément à un:
  - on inverse tous les bits: `1001 => 0110`.

Encore un peu mieux

- Complément à deux:
  - on inverse tous les bits,
  - on ajoute 1 (on ignore les dépassements).
- Comment écrit-on -4 en 8 bits?

Entiers signés (2/2)

Beaucoup mieux

- Complément à un:
  - on inverse tous les bits: `1001 => 0110`.

Encore un peu mieux

- Complément à deux:
  - on inverse tous les bits,
  - on ajoute 1 (on ignore les dépassements).
- Comment écrit-on `-4` en 8 bits?

```
 4 =  00000100

        --------
-4 =>   00000100


       11111011
     + 00000001
     ----------
       11111100
```

Le complément à 2 (1/2)

Questions:

- Comment on écrit +0 et −0?
- Comment calcule-t-on 2 + (−4)?
- Quel est le complément à 2 de 1000 0000?

Le complément à 2 (1/2)

Questions:

- Comment on écrit +0 et -0?
- Comment calcule-t-on `2 + (-4)`?
- Quel est le complément à 2 de `1000 0000`?

Réponses

- Comment on écrit +0 et -0?

```
+0 = 00000000
-0 = 11111111 + 00000001 = 100000000 => 00000000
```

- Comment calcule-t-on `2 + (-4)`?

```
  00000010          2
+ 11111100        +  -4
----------        -----
  11111110           -2
```

- En effet

```
11111110 => 00000001 + 00000001 = 00000010 = 2.
```

Le complément à 2 (1/2)

Quels sont les entiers représentables en 8 bits?

Le complément à 2 (1/2)

Quels sont les entiers représentables en 8 bits?

```
01111111 =>  127
10000000 => -128 // par définition
```

Quels sont les entiers représentables sur N bits?

Le complément à 2 (1/2)

Quels sont les entiers représentables en 8 bits?

```
01111111 =>  127
10000000 => -128 // par définition
```

Quels sont les entiers représentables sur N bits?

$$-2^{N-1} ... 2^{N-1} - 1.$$

Remarque: dépassement de capacité en `C`

- Comportement indéfini!

Types composés: `struct` (1/6)

Fractions

- Numérateur: `int num;`
- Dénominateur: `int denom`.

Addition

```
int num1 = 1, denom1 = 2;
int num2 = 1, denom2 = 3;
int num3 = num1 * denom2 + num2 * denom1;
int denom3 = denom1 * denom2;
```

Pas super pratique....

Types composés: `struct` (2/6)

On peut faire mieux

- Plusieurs variables qu'on aimerait regrouper dans un seul type: `struct`.

```
struct fraction { // déclaration du type
    int32_t num, denom;
}

struct fraction frac; // déclaration de frac
```

Types composés: `struct` (3/6)

Simplifications

- `typedef` permet de définir un nouveau type.

```
typedef unsinged int uint;
typedef struct fraction fraction_t;
typedef struct fraction {
    int32_t num, denom;
} fraction_t;
```

- L'initialisation peut aussi se faire avec

```
fraction_t frac = {1, -2}; // num = 1, denom = -2
fraction_t frac = {.denom = 1, .num = -2};
fraction_t frac = {.denom = 1}; // argl! .num non initialisé
fraction_t frac2 = frac; // copie
```

Types composés: `struct` (4/6)

Pointeurs

- Comme pour tout type, on peut avoir des pointeurs vers un `struct`.
- Les champs sont accessible avec le sélecteur `->`

```
fraction_t *frac; // on crée un pointeur
frac->num = 1;    // seg fault...
frac->denom = -1; // mémoire pas allouée.
```

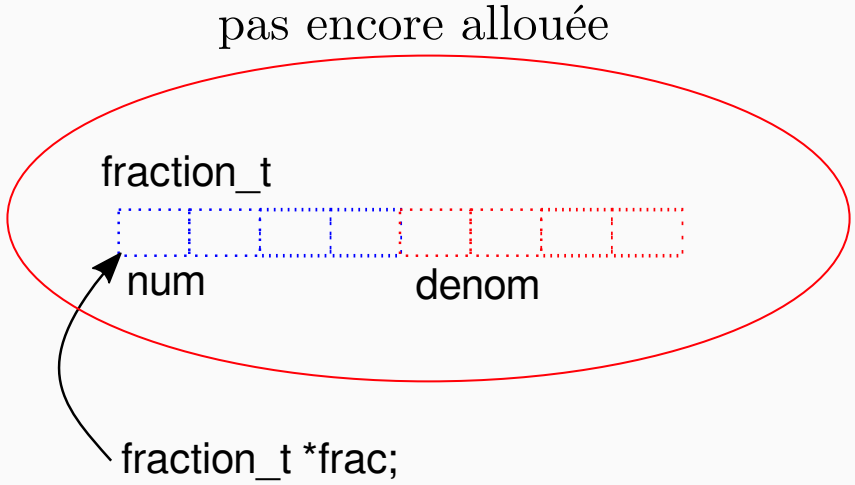

Figure 1: La représentation mémoire de `fraction_t`.

Types composés: `struct` (5/6)

Initialisation

- Avec le passage par **référence** on peut modifier un struct *en place*.
- Les champs sont accessible avec le sélecteur `->`

```
void fraction_init(fraction_t *frac,
                   int32_t re, int32_t im)
{
    // frac a déjà été allouée
    frac->num   = frac;
    frac->denom = denom;
}
int main() {
    fraction_t frac; // on alloue une fraction
    fraction_init(&frac, 2, -1); // on l'initialise
}
```

Types composés: `struct` (6/6)

Initialisation version copie

- On peut allouer une fraction, l'initialiser et le retourner.
- La valeur retournée peut être copiée dans une nouvelle structure.

```
fraction_t fraction_create(int32_t re, int32_t im) {
    fraction_t frac;
    frac.num = re;
    frac.denom = im;
    return frac;
}
int main() {
    // on crée une fraction et on l'initialise
    // en copiant la fraction créé par fraction_create
    // deux allocation et une copie
    fraction_t frac = fraction_create(2, -1);
}
```