Langages de programmation

Cours 5

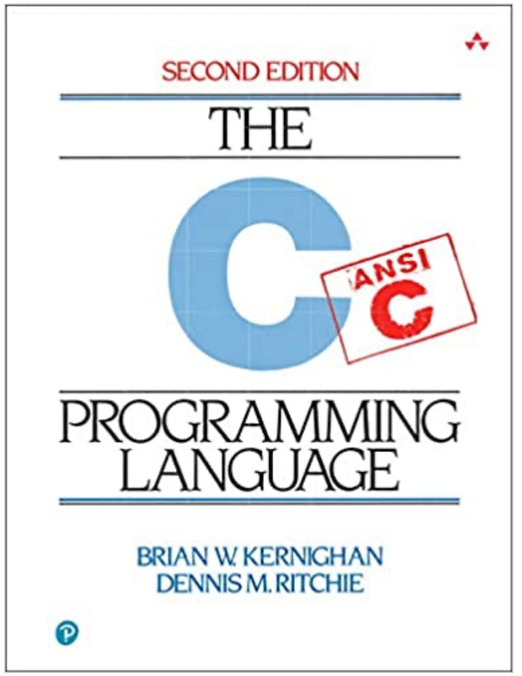
Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/lp-prog

Plan

- principes de base du langage C
- révision des programmes du cours 3
- alias et données modifiables
- gestion de la mémoire

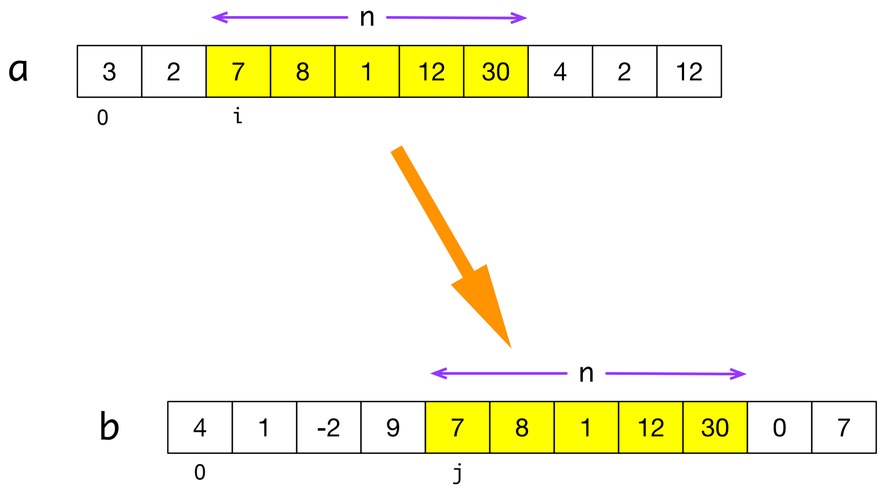


A Lire: tutorial sur le langage C `http://www.programiz.com/c-programming`

Solutions des exercices (1/3)

Exercice Écrire la fonction `copy_sublist (a, i, b, j, n)` qui copie les n éléments de *a* à partir de l'indice `i` dans le tableau b à partir de l'indice `j`

```
def copy_sublist (a, i, b, j, n) :
    if not (0 <= i <= i+n < len (a) and
            0 <= j <= j+n < len (b)):
        print ("erreur!"); return
    if not (a is b and i < j) :
        for k in range (n):
            b[j + k] = a[i + k]
    else :
        for k in range (n):
            b[j + n-1-k] = a[i + n-1-k]
```



Solutions des exercices (2/3)

Exercice Les variations du cours de l'action Google en bourse est affichée tous les jours de l'année 2020 dans le tableau g de 366 éléments (positifs ou négatifs)

Ecrire un programme pour trouver les jours où on aurait dû acheter et vendre cette action pour avoir obtenu un gain maximum ?


```
def max_win (g) :
    n = len (g)
    r = 0.0
    for i in range(n):
        for j in range (i, n):
            gain = 0.0
            for k in range (i, j+1):
               gain = gain + g[k]
               r = max (r, gain)
    return r
```

```
def max_win1 (g) :
    n = len (g)
    r = 0.0; gain_du_jour = 0.0
    for dx in g:
       gain_du_jour = max (gain_du_jour + dx, 0)
       r = max (r, gain_du_jour)
    return r
```

Solutions des exercices (3/3)

Exercice Les enfants font une ronde en se tenant par la main. Chacun note sur un bout de papier son nom et celui de son voisin de droite.

La récréation arrive. Les bouts de papier sont collectés dans une boîte.

Ecrire un programme pour remettre tous les élèves en place.

```
tickets = [['paul', 'marie'],['marie', 'charles'],['charles', 'bob'],
      ['bob', 'alice'],['alice', 'romain'],['romain', 'laurent'],
      ['laurent', 'paul']]
```

```
def reorder (tickets) :
    n = len (tickets)
    tri_paires (tickets, DROITE)
    for i in range (n):
        tickets[i][DROITE] = i
    tri_paires (tickets, GAUCHE)
    if n > 0 :
        t = tickets[0]
        for i in range (n):
            print(t[GAUCHE])
            t = tickets[t[DROITE]]
```

```
GAUCHE = 0; DROITE = 1

def tri_paires (a, col) :
    n = len (a)
    for i in range (1, n) :
        v = a[i]; j = i
        while j > 0 and a[j-1][col] > v[col] :
            a[j] = a[j-1];
            j = j-1
        a[j] = v
```

Hello world!

- premier programme dans un fichier `hello.c`

```c
#include <stdio.h>

int main(){
    printf ("Bonjour tout le monde!");
}
```

stdio.h est la bibliothèque standard des entrées/sorties

main est le début du programme, main retourne un entier

impression formattée

- et on l'exécute comme avec Python

  différence: on compile pour obtenir un fichier binaire `hello` qui est exécuté

- si on veut rajouter un retour à la ligne suivante

```c
int main(){
    printf ("Bonjour tout le monde!\n");
}
```

Hello world!

- premier programme dans un fichier `hello.c`

```c
#include <stdio.h>

char s[] = "Bonjour les amis!" ;

int main()
{
    printf ("%s\n", s);
}
```

chaîne de caractères

impression formattée

- format: `%d`, `%f`, `%c`, `%s` pour nombres entiers, flottant, caractères, chaînes de caractères

- les variables sont toutes déclarées avec leur **type**:

```c
int x = 42;
float r = 3.14;
double r1 = 3.14116;
char c = 'B';
char s[] = "Bonjour";
```

Données de base

- en C, on peut spécifier la taille exacte de toutes les données et variables

```
char x = 126;
short y = 4201;
int z = 14201;
long t = 10298309;
```

entier signé sur 8, 16, 32, 64 bits

```
float r = 3.14;
double r1 = 3.14116;
```

flottant sur 32 ou 64 bits

- il existe aussi les entiers (positifs) non-signés

```
unsigned int z1;
```

Tableaux

- les tableaux initialisés

```
char s[] = {'a', 'b', 'e', 'z', 'A', ':', 17};
int a[] = {3, 2, 7, 8, 1, 12, 30, 4, 2, 12};
float r[] = {3.14, 2.71828};
```

a

0	1	2	3	4	5	6	7	8	9
3	2	7	8	1	12	30	4	2	12

- les tableaux non initialisés

```
int a[100];

const int N = 100;
int b[N];
```

constante entière

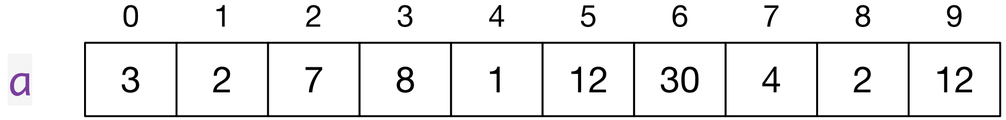
on déclare la longueur du tableau

tableaux non initialisés
=
DANGER !

Tableaux

- on gère soi-même la longueur d'un tableau

```
int main(){
   int a[] = {3, 2, 7, 8, 1, 12, 30, 4, 2, 12};
   for (int i=0; i < 10; ++i)
      printf ("%d ", a[i]);
   printf ("\n");
}
```




```
mac$ ./prog1
3 2 7 8 1 12 30 4 2 12
```

- attention à la longueur du tableau

```
int main(){
   int a[] = {3, 2, 7, 8, 1, 12, 30, 4, 2, 12};
   for (int i=0; i < 20; ++i)
      printf ("%d ", a[i]);
   printf ("\n");
}
```

débordement de tableau non contrôlé = DANGER !

```
mac$ ./prog1
3 2 7 8 1 12 30 4 2 12 -495779747 1512067678 -469809976 32766 542434849 32767 542434849 32767 0 0
```

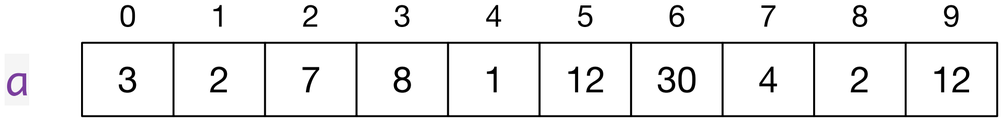
Tableaux

- on gère soi-même la longueur d'un tableau

```
void print_array (int a[], int n) {
   for (int i=0; i < n; ++i)
      printf ("%d ", a[i]);
   printf ("\n");
}

int main() {
   int a[] = {3, 2, 7, 8, 1, 12, 30, 4, 2, 12};
   print_array (a, 10);
}
```

```
mac$ ./prog1
3 2 7 8 1 12 30 4 2 12
```



Tableaux

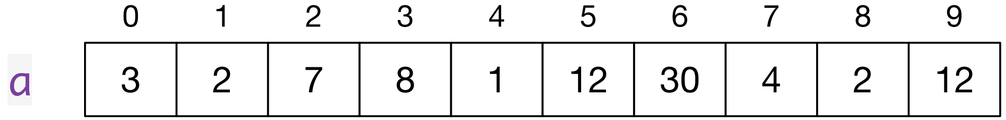
- on gère soi-même la longueur d'un tableau

```
#include <limits.h>

int max_of (int a[], int n) {
   int r = INT_MIN;
   for (int i=0; i < n; ++i)
      if (r < a[i])
         r = a[i];
   return r;
}

int main() {
   int a[] = {3, 2, 7, 8, 1, 12, 30, 4, 2, 12};
   printf("%d\n", max_of (a, 10));
}
```

```
mac$ ./prog1
30
```




```
int min_of (int a[], int n) {
   int r = INT_MAX;
   for (int i=0; i < n; ++i)
      if (r > a[i])
         r = a[i];
   return r;
}
```

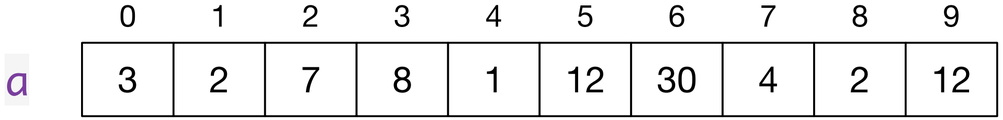
Tableaux

- on gère soi-même la longueur d'un tableau

```
int index_max_of (int a[], int n) {
   int r = INT_MIN;
   int imax = -1;
   for (int i=0; i < n; ++i)
      if (r < a[i]) {
         r = a[i]; imax = i;
      }
   return imax;
}

int main() {
   int a[] = {3, 2, 7, 8, 1, 12, 30, 4, 2, 12};
   printf("%d\n", index_max_of (a, 10));
}
```

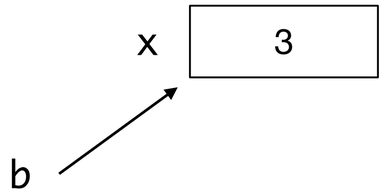
```
mac$ ./prog1
6
```



Tableaux

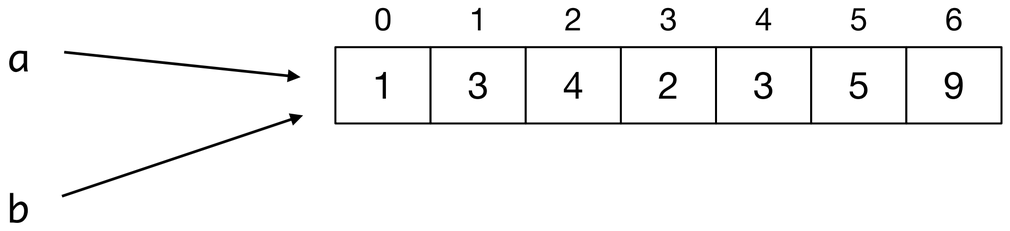
- adresse, pointeur (aussi appelé référence)

```c
int main() {
   int x = 3;
   int *b = &x;
   printf("%d\n", *b);
}
```



- en C, pour tout tableau *a*, on a l'équation: `a == &a[0]`

```c
int main() {
   int a[] = {1, 3, 4, 2, 3, 5, 9};
   print_array (a, 7);
   int *b = a;
   print_array (b, 7);
}
```

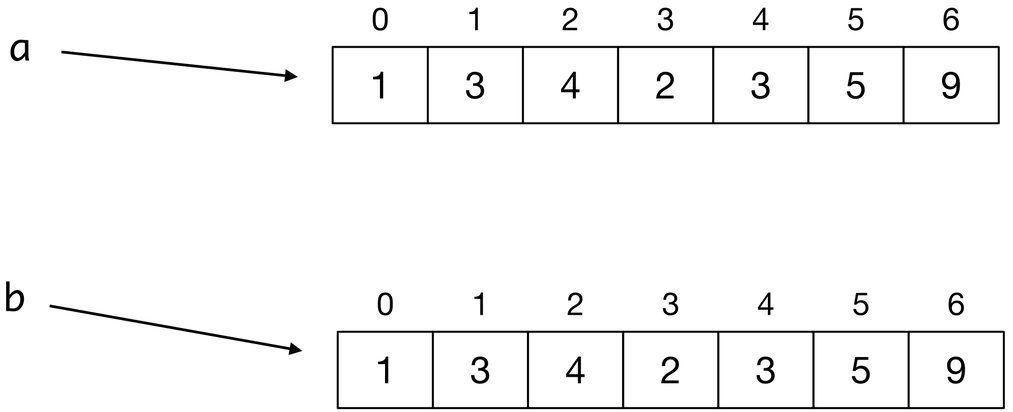


Tableaux

- allocation mémoire (*memory alloc*)

```c
#include <stdlib.h>

int *copy_array (int a[], int n) {
   int *r = malloc (n * sizeof(int));
   for (int i=0; i<n; ++i)
      r[i] = a[i];
   return r;
}
```




```c
int *new_array (int a[], int n, int v) {
   int *r = malloc (n * sizeof(int));
   for (int i=0; i<n; ++i)
      r[i] = v;
   return r;
}
```

```c
int main() {
   int a[] = {1, 3, 4, 2, 3, 5, 9};
   print_array (a, 10);
   int *b = copy_array (a, 10);
   print_array (b, 10);
}
```

Prochain cours

- un bon tutorial Python: `http://www.programiz.com/python-programming`
- un peu plus de C
- débuts en C++
- principes de la programmation objet
- un peu plus d’algorithmique