Langages de programmation

Cours 6

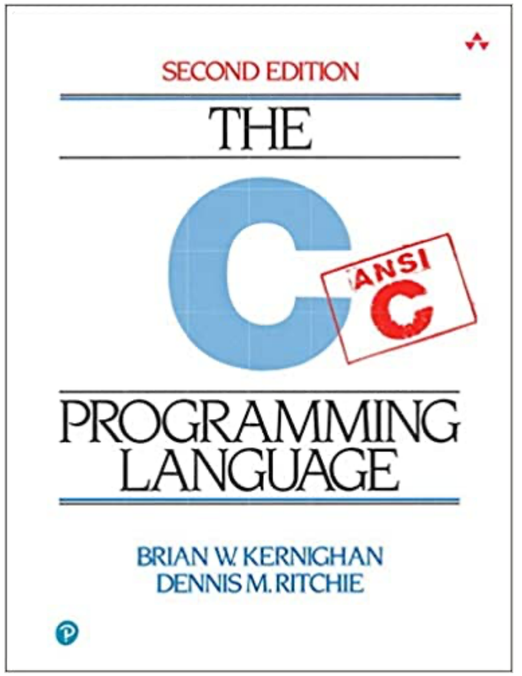
Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/lp-prog

Plan

- allocation mémoire
- chaînes de caractères
- arithmétique des pointeurs
- booléens

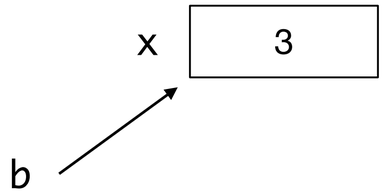


A Lire: tutorial sur le langage C `http://www.programiz.com/c-programming`

Tableaux

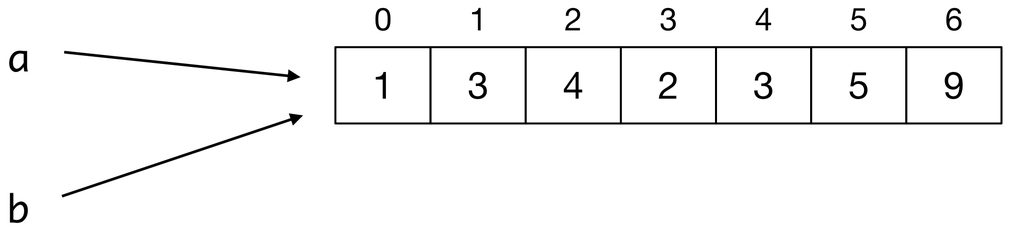
- adresse, pointeur (aussi appelé référence)

```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]`

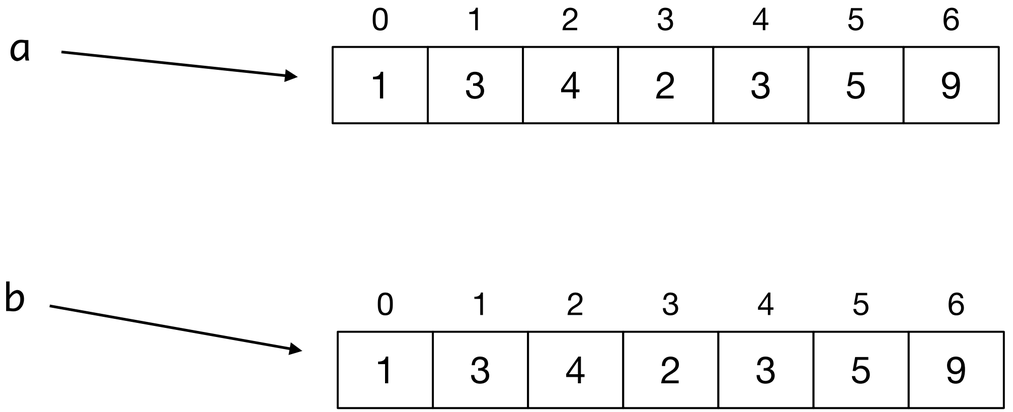
```c
int main() {
   int a[] = {1, 3, 4, 2, 3, 5, 9};
   print_array (a, 7);
   int *b = a;
   print_array (b, 7);
}
```



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;
}
```

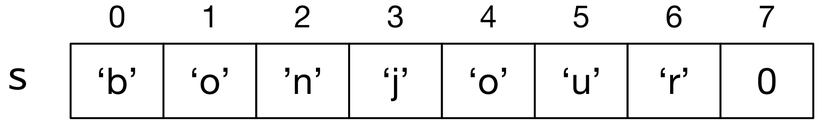
```c
int main() {
   int a[] = {1, 3, 4, 2, 3, 5, 9};
   print_array (a, 10);
   int *b = copy_array (a, 10);
   print_array (b, 10);
}
```

Chaînes de caractères

- les chaînes de caractères sont des tableaux de caractères se finissant par 0 ou encore ‘\0’

```c
#include <string.h>

int main() {
    char s[] = "bonjour";
    printf ("%s, longueur = %lu\n", s, strlen(s));
}
```

```
mac$ ./prog2
bonjour, longueur = 7
```

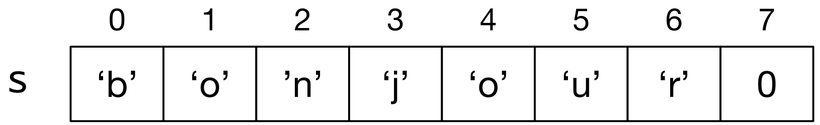



```c
#include <string.h>
#include <stdbool.h>

bool is_palindrome (char s[]) {
    int n = strlen(s);
    for (int i=0; i < n; ++i)
        if (s[i] != s[n-1-i])
            return false;
    return true;
}

int main() {
    char s[] = "kayak";
    if (is_palindrome(s))
        printf ("%s est un palindrome", s);
    else
        printf ("%s n'est pas un palindrome", s);
}
```

Chaînes de caractères

- les chaînes de caractères sont des tableaux de caractères se finissant par 0

```c
#include <string.h>

int main() {
    char s[] = "bonjour";
    printf ("%s, longueur = %lu\n", s, strlen(s));
}
```




```
mac$ ./prog2
bonjour, longueur = 7
```

```c
#include <string.h>
#include <stdbool.h>

bool is_palindrome (char s[]) {
    int n = strlen(s);
    for (int i=0; i < n; ++i)
        if (s[i] != s[n-1-i])
            return false;
    return true;
}

int main() {
    char s[] = "kayak";
    if (is_palindrome(s))
        printf ("%s est un palindrome", s);
    else
        printf ("%s n'est pas un palindrome", s);
}
```

Chaînes de caractères

Exercice Écrire la fonction `strlen (char a[])` qui retourne
la longueur de la chaîne *a*

Exercice Écrire la fonction `strmem (char a[], char c)` qui teste
si le caractère `c` apparait dans la chaîne *a*

Exercice Écrire la fonction `strcat (char a[], char b[])` qui met
la chaîne *b* au bout de la chaîne *a*

Exercice Écrire la fonction `strcp (char a[])` qui retourne
une copie de la chaîne *a*

Exercice Écrire la fonction `str_catenate (char a[], char b[])`
qui retourne une nouvelle chaîne concaténant *a* et *b*

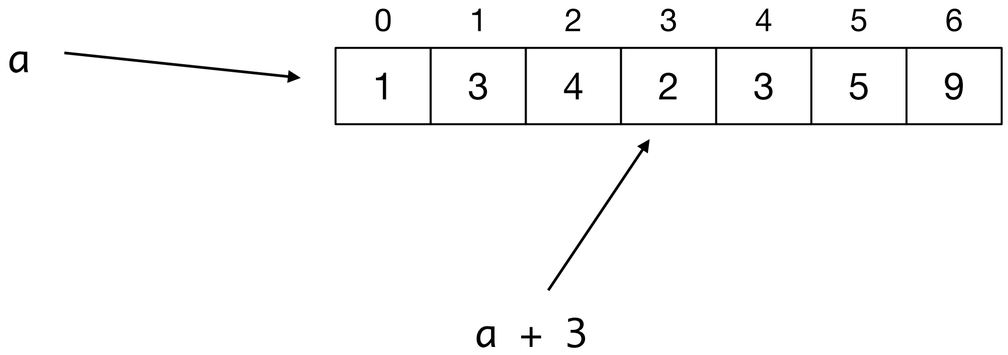
Arithmétique des pointeurs

- adresse, pointeur (aussi appelé référence)

```
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]`

- arithmétique des pointeurs: `a + i == &a[i]`




```
int main() {
   int a[] = {3, 2, 7, 8, 1, 12, 30, 4, 2, 12};
   print_array (a, 10);
   int *b = copy_array (a+3, 6);
   print_array (b, 6);
}
```

Arithmétique des pointeurs

Exercice Écrire la fonction `mystrlen (char a[])` qui retourne la longueur de la chaîne `a`

```c
int mystrlen (char a[]) {
   int r = 0;
   for (int i=0; a[i] != 0; ++i)
     ++r;
   return r;
}
```

```c
int mystrlen1 (char a[]) {
   int r = 0;
   for (char *p = a; *p != 0; ++p)
      ++r;
   return r;
```

Booléens

- librairie `stdbool` des booléens

```c
#include <stdbool.h>

int main(){
   printf ("%d et %d \n", true, false);
}
```

ne pas confondre = et ==

DANGER !

- mais aussi , on a 0 pour Faux, et toute valeur non nulle pour Vrai

```c
int x = 4;

int main() {
   if (x = 3)
     printf ("x vaut 3");
   else
     printf ("x ne vaut pas 3");
}
```

ERREUR

```c
int x = 4;

int main() {
   if (x == 3)
     printf ("x vaut 3");
   else
     printf ("x ne vaut pas 3");
}
```

CORRECT

Booléens

- on a 0 pour Faux, et toute valeur non nulle pour Vrai

```c
int mystrlen2 (char a[]) {
   int r = 0; char *p = a;
   while (*p) { p++; r++; }
   return r;
}
```

```c
int mystrlen2 (char a[]) {
   int r = 0; char *p = a;
   while (*p++) ++r;
   return r;
}
```

style de la librairie C

C++

C++

Stroustrup

=

C

programmation système

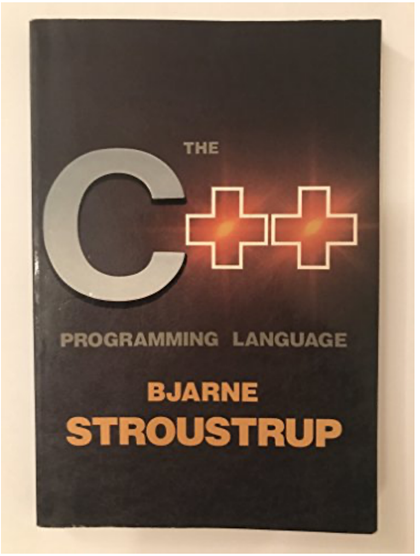
Ritchie

+

Simula

programmation objet

Nygaard



- Smalltalk aussi ancêtre de la programmation objet

Débuts en C++

- le suffixe des noms de fichiers est `.C` au lieu de `.c`
- C++ a les mêmes types de base et opérateurs que C
- les entrées-sorties sont différentes

flux de sortie

```
#include <iostream>
using namespace std;

int main() {
    cout << "Bonjour tout le monde" << endl;
    return 0;
}
```

flux d'entrée

```
int main() {
    char a;
    int num;
    cout << "Entrez caractère et nombre: " ;
    cin >> a >> num;
    cout << "a = " << a << "; ";
    cout << "num = " << num << endl;
    return 0;
}
```

end line

[on n'a pas à spécifier le type du format comme dans `printf` (merci à la programmation objet !) — voir plus tard]

Débuts en C++

- l'allocation mémoire se fait avec la primitive `new` (plus besoin de `malloc` et `sizeof` !)

```
int main() {
    int *a = new int[10];
    for (int i = 0; i < 10; ++i) a[i] = i*i;

    cout << a << " " << endl;
    for (int i = 0; i < 10; ++i)
       cout << a[i] << " ";
    return 0;
}
```

ne pas oublier d'initialiser le tableau

- la dé-allocation mémoire se fait avec la primitive `free` (qui existe aussi en C)

[très peu utilisé à cause des alias — problème général du ramasse-miettes (*garbage collector*)]

Surcharge des fonctions

- les fonctions peuvent être surchargées (*overloading*)
- elles peuvent avoir un code différent selon le type ou le nombre de leurs arguments

```
int absolu (int x) {
    if (x < 0)
      return -x;
    else
      return x;
}

double absolu (double x) {
    if (x < 0)
      return -x;
    else
      return x;
}

int main() {
    cout << absolu(22) << " " << absolu (-42) << endl;
    cout << absolu(3.14) << " " << absolu (-1.28) << endl;
}
```

résultat entier

résultat réel flottant 64bits

Prochain cours

- un bon tutorial C++: `http://www.programiz.com/cpp-programming`
- enregistrements, classes, objets
- structures de données dynamiques
- principes de la programmation objet