

Système d'exploitation

Cours no. 9

Jean-Sébastien Coron

Université du Luxembourg

November 8, 2009

- Les boucles for.
- Les tableaux.

- Une boucle For permet de répéter une instruction plusieurs fois, à l'aide d'un compteur.
- Syntaxe: `for(<init>;<test>;<gestion compteur>)`
- Exemple: afficher les nombres de 1 à 10.
 - `for (i=1;i<=10;i++) printf("%d\n",i);`
- `init`: permet d'initialiser le compteur.
- `test`: permet de tester la valeur du compteur.
- `gestion compteur`: permet d'incrémenter le compteur.

Exemple

- Le programme suivant calcule 2^n , pour un entier n donné:

```
int c=1;
int i;
for(i=0;i<n;i++)
{
    c=c*2;
}
// c contient  $2^n$ .
```

Calcul de $c = a^b$

- ```
#include <stdio.h>
int main()
{
 int a=2; int b=8;
 int i;

 int c=1;
 for(i=0;i<b;i++)
 {
 c=c*a;
 }
 printf("%d\n",c);
}
```

- Calcul de  $n! = n \cdot (n - 1) \dots 3 \cdot 2 \cdot 1$ .

```
#include <stdio.h>
int main()
{
 int n,i,c=1;
 scanf("%d",&n);

 for(i=2;i<=n;i++)
 {
 c=c*i;
 }
 printf("%d\n",c);
}
```

- Les tableaux permettent de définir un groupe de cases mémoires de même type.
  - Par exemple, si on veut stocker les notes d'un élève dans un tableau, on peut définir:

```
int notes[5]; \\ on déclare un tableau de 5 entiers
notes[0]=15; \\ première notes
notes[1]=8;
notes[2]=16;
notes[3]=17;
notes[4]=9; \\ cinquième note
```

- On a ici un tableau d'entiers.

# Différents types de tableaux

- Les différents types possibles:
  - `float tabf[5]`: un tableau de 5 `float`.
  - `double tabd[10]`: un tableau de 10 `double`.
  - `int tabi[7]`: un tableau de 7 `int`.
- Attention:
  - Un tableau à  $n$  éléments est indexé de 0 à  $n - 1$ :
    - `int tabi[7]`.
      - Les cases du tableau vont de `tabi[0]` à `tabi[6]`.



- La taille d'un tableau doit être constante.
  - Cette taille doit être soit écrite explicitement dans le programme, par ex. `int tab[10]`
  - Ou bien on peut utiliser l'instruction `#define`:

```
#include <stdio.h>
#define N 10 \\ on définit la constante N=10
int main()
{
 int tab[N];
 int autretab[5];
}
```

- Avantage: il est plus facile de modifier la taille du tableau.

- Les caractères sont stockés sur un octet (8 bits), comme des entiers non signés.
  - Les caractères sont codés suivant le code ASCII
    - 'A' → 65, 'B' → 66,...
    - '0' → 48,...
- Afficher un caractère:

```
char x;
x='A';
printf("%c",x);
```

- Un tableau de caractère constitue une chaîne de caractère.
  - `char ch[10]="hello";` crée un tableau de caractère tel que
  - `ch[0]='h'`, `ch[1]='e'`, `ch[2]='l'`, `ch[3]='l'`, `ch[4]='o'`
  - `ch[5]='\0'`, qui est le caractère de fin de chaîne.
  - Les autres caractères ne sont pas initialisés.
- Affichage d'une chaîne de caractère:
  - `printf("%s",ch);`

# Initialisation d'un tableau

- On peut initialiser un tableau à l'aide d'une boucle for:

```
#include <stdio.h>
#define N 10
int main()
{
 int tab[N];
 int i;
 for(i=0;i<N;i++)
 {
 tab[i]=0;
 }
}
```

- Tous les entiers du tableau sont initialisés à zéro.

# Exemple

- Calcul de factoriel à l'aide d'un tableau:
  - $n! = n \cdot (n - 1) \dots 2 \cdot 1$

```
#include <stdio.h>
#define N 10
int main()
{
 int fac[N];
 int i;
 tab[0]=1;
 for(i=1;i<N;i++)
 {
 fac[i]=fac[i-1]*i;
 }
}
```

# Tableaux à deux dimensions

- Il est possible de définir des tableaux à deux dimensions ou plus.
  - `int tab[4][3]`; déclare un tableau d'entier de taille 4\*3.
- Initialisation d'un tableau à deux dimensions:

```
#include <stdio.h>
#define M 10
#define N 5
int main()
{
 int tab[M][N];
 int i,j;
 for(i=0;i<M;i++)
 for(j=0;j<N;j++)
 tab[i][j]=0;
```

# Génération de nombres aléatoires

- La fonction `rand()` renvoie un entier aléatoire entre 0 et `RAND_MAX`.
  - Sous `gcc/Cygwin`, `RAND_MAX=231`.
- Renvoyer un entier entre 0 et  $n$ :
  - Utiliser `rand() % n`.
- Initialisation du générateur aléatoire.
  - Fonction `srand()`.
  - Généralement initialisée avec le temps `time(0)`.
  - `#include <stdlib.h>`

# Exemple

- Affiche un nombre aléatoire entre 0 et 99:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
 srand(time(0));
 int x;
 x=rand() % 100;
 printf("%d\n",x);
}
```



# Arguments de la commande

- Il est possible d'obtenir les arguments de la commande du programme.
  - Par exemple, pour un programme `fact` qui calcule une factorielle, on voudrait pouvoir taper

```
$ fact 5
120
```
- Avantage:
  - Cela évite d'écrire `int n=5` dans le programme (il faut recompiler lorsqu'on modifie cette valeur).
  - Cela évite de demander l'entier  $n$  à l'utilisateur.

- Les mots de la ligne de commande sont stockés dans le tableau argv.
- La variable argc contient la taille du tableau (=le nombre de mots de la ligne de commande).

```
#include <stdio.h>
int main(int argc,char *argv[])
{
 int i;
 for(i=0;i<argc;i++) // boucle sur tous le tableau
 {
 printf("%s\n",argv[i]); // affiche les mots
 }
}
```

- Si le nom du programme précédent est affiche, alors
  - `$ affiche hello world 2`  
affiche  
hello  
world  
2
- La taille du tableau est `argc=4`.

# Conversion chaîne entier

- La fonction `int atoi()` permet de convertir une chaîne de caractère en un entier.
  - Utile pour convertir les mots `argv[i]` qui sont des chaînes de caractères.
  - Exemple: affiche le carré d'un nombre.

```
#include <stdio.h>
int main(int argc, char *argv[])
{
 int a=atoi(argv[1]); // conversion
 printf("%d\n", a*a);
}
```

- \$ carre 3  
9