Initiation au C

Cours 2

David Simplot

Exercices

1. Écrire une fonction “ordonne” qui prend en entrée deux entiers, les ordonne et retourne le plus grand.

2. Écrire une fonction “permute” qui prend en entrée deux entiers et qui les permute. Réécrire “ordonne” en utilisant “permute”.

ex13.c : Constantes

```
* calcul de factorielle */
include <stdio.h>

define MIN 0
define MAX 100

nt main(void)

 int n;
 printf("Valeur de n \
entre %d et %d) ? ", MIN, MAX);
 scanf("%d", &n);
 while ( n<MIN || n>MAX )
   scanf("%d", &n);
 printf("n=%d\n", n);
 return(0);
```

- Syntaxe :

  `#define` *nom_constante valeur*

- Il n'y a pas de ";"...
- Permet des modifications plus faciles du code.

ex14.c : Macros

- Macros simples : `#define delta(a,b) a-b`
- Macros condionnelles :

```
/* calcul de factorielle */
#include <stdio.h>

#define max(a,b) ( a > b ? a : b)
```

```
int main(void)
{
  int x, y;
  printf("x et y ?\n");
  scanf("%d %d", &x, &y);
  printf("max=%d\n",
         max(x,y));
  return(0);
}
```

  - Syntaxe :

`#define` *macro*(*param*) (*condition* ? *alors* : *sinon*)

Types simples

`char`	un octet	8 bits -128, 127 ou 0, 255
`int`	entier	32 bits -2 147 483 648, 2 147 483 647
`short int`	entier	16 bits -32 768, 32 767
`long int`	entier	32 bits
`float`	flottant simple précision	32 bits
`double`	flottant double précision	64 bits
`long double`	flottant d-préc. étendu	96 bits

On peut mettre le "tag" `signed` ou `unsigned`.

Entrées/Sorties Standards

La fonction (`stdio.h`)

```
int printf(const char *format, ...);
```

permet d'écrire sur la sortie standard.

- La chaîne de caractères `format` contient des caractères à écrire directement (dont les caractères d'échappement) ainsi que des spécifications de conversion commençant par %.
- Retourne le nombre de caractères écrits ou un nombre négatif en cas d'erreur.
- Pour les autres possibilités, voir `man 3 printf`.

Entrées/Sorties Standards (suite)

Exemple

```
int n = 0;
char car = 'a';
printf("Voici un caractère : %c,\n et un entier \
%d\n",car,n);
```

```
{---} a.out
Voici un caractère :  a,
et un entier 0
{---}
```

Saisie "clavier"

La fonction (`stdio.h`)

```
int scanf( const char *format, ...);
```

permet de lire sur l'entrée standard, elle :

- lit des données sur l'entrée standart conformément aux spécifications données par `format` ;
- affecte les valeurs lues aux arguments suivant `format` qui doivent être des pointeurs (voir plus loin) ;
- retourne le nombre d'objects lus et affectés si la lecture s'est deroulée normalement, `0` s'il reste des caractères à lire mais que ceux-ci ne correspondent pas au format demandé, ... ;
- pour plus de détails, voir `man scanf`.

Entrées/Sorties Standards (suite)

Exemple.

```
int n;
char c;
int result;
result = scanf ("%d %c",&n,&c);
printf ("Saisies : %d et %c, resultat %d", n,c,result);
```

Caractères d'échappement.

`\b`	backspace
`\f`	saut de page
`\n`	fin de ligne
`\r`	retour chariot
`\t`	tabulation horizontale
`\v`	tabulation verticale
...	

Caractères de conversion

Présentation TRÈS simplifiée...

d	entier sous forme décimale
i	entier (sous forme octale si précédé de `0` ou hexadécimale (si précédé de `0x` ou `0X`
c	`scanf` : les caractères suivants (par défaut 1) sont placés dans le tableau, caractères d'espacement compris `printf` : affichage d'un caractère
s	`scanf` : chaîne de caractères – sans caractères d'espacement – `'\0'` est ajouté en fin `printf` affiche les caractères jusqu'à `'\0'` (sauf si précision indiquée)
...	

Tableaux

- Un tableau contient des valeurs d'un même type.
- Les tableaux sont toujours indicés par des valeurs entières de `0` à `longueur-1`.
- Il n'y a pas de marque de fin de tableau.

Déclaration :

`int tent[10];`	tableau de 10 entiers
`char tcar[8];`	tableau de 8 caractères

Indexation :

`tent[3] = 1;`	$4^{\text{ème}}$ case du tableau
`tcar[6+1] = 'a';`	dernière case du tableau

Tableaux – Exemple I

```
int main (void) {
  int t[10], i;
  for (i = 0; i < 10; i++)
    t[i] = 10+i;
  for (i = 0; i < 10; i++)
    printf("%d ",t[i]);
}
```

Affichage :

```
{---} a.out
10 11 12 13 14 15 16 17 18 19
```

```
  for (i = 0; i <= 10; i++)
    printf("%d ",t[i]);
```

↑

incorrect !!

↓

```
{---} a.out
10 11 12 13 14 15 16 17 18
19 -1073742824
```

Tableaux – Exemple II

pas bon

```
nt main(void)
{
  int i, a[10];

  for ( i= 0 ; i<10 ; i++)
    a[i] = 2*i+1;

  ...

  return(0);
}
```

bon !!!

```
#define MAX 10
int main(void)
{
  int i, a[MAX];

  for ( i= 0 ; i<MAX ; i++)
    a[i] = 2*i+1;

  ...

  return(0);
}
```

- `a` est un tableau de 10 entiers
- `a[0]` = premier élément, ... , `a[9]` = 10ème élément

`a`	pointeur vers le tableau adresse du 1er élt	type `int*`
`a[x]`	valeur du x+1 ème élt	type `int`
`&(a[x])`	adresse mémoire x+1 ème élt	type `int*`
`a+x`	idem	type `int*`

Généralités sur les Tableaux

- Attention : `sizeof(a)`$\neq$`sizeof(int *)` :

```
int a[10], *b;
printf("%p %p\n", a, b);
printf("%d %d\n", sizeof(a), sizeof(b));
{---} a.out
0xbfffef30 0x4001eca0
40 4
```

- Attention : `b` est non initialisé $\Rightarrow$ n'appartient pas obligatoirement au programme...

```
*b=0     -> segmentation fault
b=a      OK
a=b      ne fonctionne pas          type réel de a : const int *
```

Chaînes de caractères

- Les chaînes de caractères sont des tableaux de caractères.
- Par CONVENTION, la fin d'une chaîne est marquée par le caractère `'\0'` (caractère de code ASCII 0).
- Les fonctions d'entrées/sorties utilisent cette convention.

```
int main (void) {
 char machaine[10];
 int i;
 for (i = 0; i < 5; i++)
   machaine[i] = 'a' + i;
 machaine[5] = '\0';
 printf("%s!\n",machaine);
}
```

Exécution :

```
{---} a.out
abcde!
```

Exemple : parcours de chaîne

```
int main (void) {

 char machaine[10];
 int i=0, cpt = 0;

 scanf("%s",machaine);
 while (machaine[i] != '\0') {
   if (machaine[i] == 'a')
      cpt ++;
   i++;
 }
 printf("nb de 'a' dans %s : %d\n",machaine,cpt);
}
```

Exécution :

```
{---} a.out
atlas
nb de 'a' dans atlas :  2
```

Exemple d'E/S avec des chaînes de caractères

```
char chaine[] = "azertyuiop";
char saisie[20];
```

Instructions	Affichages
• `printf("%s\n",chaine);`	`azertyuiop`
`printf("%.5s\n",chaine);`	`azert`
`printf("%c\n",chaine[0]);`	`a`
• `scanf("%s",saisie);`	`a z e`
`printf("%s\n", saisie);`	`a`
• `scanf("%5c", saisie);`	`aaaaa`
`printf("%s\n",saisie);`	`aaaaaDüÿ>`

- `scanf("%5c", saisie);` a z e r t y

  `saisie[3] = '\0';`

  `printf("%s\n", saisie);` a z e

Paramètre tableau

C'est l'adresse du premier élément qui est passée.

Exemple.

```
int strlen (char *s) {
   int n;
   for (n = 0; *s != '\0'; s++) n++;
   return n;
}
```

ou

```
int strlen (char s[]) {
```

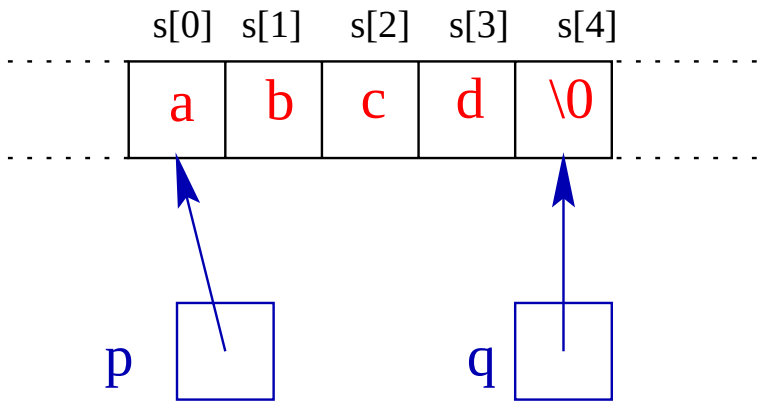
s est passé par valeur !

`char s[]` et `char *s` sont équivalents comme paramètres formels.

Calculs sur les pointeurs

- entier {+, -} pointeur ;
- pointeur - pointeur ;
- comparaison de deux pointeurs : ==,!=,<,<=,>,>= ;
- par convention, on affecte 0 ou NULL à un pointeur qui ne pointe rien.

Exemple.




```
int strlen (char *s) {
    char *p =s;
    while (*p != '\0') p++;
    return p-s;
}
```

Chaînes de caractères et pointeurs

Les chaînes de caractères peuvent être manipulées avec des pointeurs.

Exemples. Copie d'une chaîne dans une autre.

```
void strcpy (char *d, char *s) {
  int i = 0;
  while ((d[i] = s[i]) != '\0') i++;
}

void strcpy (char *d, char *s) {
  while ((*d = *s) != '\0') {d++; s++;}
}

void strcpy (char *d, char *s) {
  while ((*d++ = *s++) != '\0');
}

void strcpy (char *d, char *s) {while (*d++ = *s++);}
```

Chaînes constantes

- Les constantes de type chaîne sont exprimées entre guillemets : `"bonjour"`.
- Attention, `"a"` et `'a'` sont totalement différents.
- Les caractères d'échappement peuvent être utilisés dans une chaîne : `"bonjour, \n \t tout le monde"`.

Exemples.

`char machaine[] = "bonjour";` déclaration d'un tableau de caractères de aille nécessaire pour contenir la chaîne avec lequel le tableau est initialisé. Il st possible de modifier le contenu du tableau mais PAS l'espace pointé.

`char *machaine = "bonjour";` déclaration d'un pointeur de caractère nitialisé en pointant sur la chaîne constante `"bonjour"`. Il est possible de

modifier le pointeur pour le faire pointer sur une autre zone mémoire mais il
n'est pas possible de modifier le contenu de la chaîne constante.

Pointeurs : récapitulation

Déclaration	Affectation

`int *p1;`
`int *p2;`
`char * pc;`
`float * pf;`
	`int i;`
	`char c;`
	`p1 = &i;`
	`p2 = p1;`
	`pc = &c;`

Accès à la valeur pointée	**Arithmétique**

`c = 'a';`
`*p1 = 1;`
`printf("%c %d",*pc, i);`
 	
Affichage : a1.
	`int t[3] = {10,11,12};`
	`p2 = t;`
	`p2 = p2 +2;`
	`printf("%d", *p2);`
	
	Affichage : 12.

Exemples

`int pi[3] = {0,2,3};`	déclaration d'un tableau
`int *qi;`	déclaration d'un pointeur
`int y;`	déclaration d'un entier
`*pi = 1;`	utilisation du tableau en pointeur
`y = *pi + 1;`	contenu impliqué dans un calcul
`*pi = *pi + 10;`	idem + affectation
`*pi += 1;`	idem
`qi =pi;`	qi pointe sur la $1^{\text{ère}}$ case
`*(++qi);`	contenu de la $2^{\text{ème}}$ case
`*qi++;`	contenu de la $2^{\text{ème}}$ case
	qi pointe sur la $3^{\text{ème}}$ case