Référence pour le langage de programmation C

N.K. “C Programming : A modern approach ”, Norton, 1996.

Aujourd’hui

?? Pointeurs et tableaux (Ref. King)

?? Chaînes de caractères (Ref. King)

?? Classes et objets en C++ (Ref. Folk, Zoellick et Riccardi, Section 1.5)

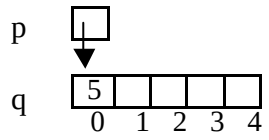
?? Un exemple utile pour les devoirs

Pointeurs et Tableaux

1. Les Pointeurs pointent sur les éléments des tableaux et ils sont utiles dans la manipulation des tableaux.

```
int a[5], *p, *q;

p = &a[o];   // permet à ‘p’ qu’elle pointe sur a[0]
*p = 5;      //  met 5 dans la variable pointée par p
```



L’arithmétique sur les pointeurs

```
p   = &a[2];   //  p pointe sur a[2]
p+ = 2 ;       //  p pointe sur a[4]
```

Ajouter 2 à ‘p’ l’avance 2 ‘int’ et pas 2 octets car ‘p’ est un pointeur sur ‘int’.

```
q  =  p – 3;
*q =  10;          //  place 10 dans a[1]
```

Le programme suivant somme les éléments d'un tableau en utilisant les pointeurs :

```
int a[10], sum, *p;
        .
        .
        .
sum = 0;
for (p = &a[0]; p < &a[10]; p++) {
        sum += *p;
}
```

2. Utilisant le nom du tableau comme un pointeur
   Le nom d'un tableau peut être utilisé comme un pointeur sur le premier élément du tableau.

```
int a[10];

*a = 7;            // affecte 7 à a[0]
*(a+1) = 12;       // affecte 12 à a[1]
```

La boucle 'for' dans l'exemple précédent peut être écrite de la façon suivante :

```
for (p = a; p < a+N; p++) {
        sum += *p;
}
```

Important : un tableau peut être utilisé comme un pointeur mais c'est impossible de l'affecter une valeur.

```
while (*a!=0) {
        a++;     // faux
}
```

La bonne façon de l'écrire est la suivante :

```
p = a;
while (*p!=0) {
        p++;
}
```

3. Les tableaux et les arguments
   Quand il est passé à une fonction, le nom du tableau est toujours considéré comme un pointeur.

```
int find_largest(int a[], int n) {
        int i, max;
        max = a[0];
        for (i = 1; i < n; i++) {
                if (a[i] > max) max = a[i];
        }
        return max;
}

void store_zeros (int a[], int n) {
        int i;
        for (i = 0; i<n; i++)
                a[i] = 0;
}
```

   Pour indiquer que le paramètre d'un tableau ne va pas être modifier (comme dans find_largest) on peut inclure const dans sa déclaration.

```
int find_largest(const int a[], int n) {
        …
}
```

   Les tableaux constants ne sont pas copiés quand ils sont passés comme des arguments à une fonction; seuls les pointeurs sont copiés.

   L'appel :

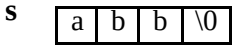
```
        Largest = find_largest(b,N);
```

   permet au pointeur ,qui pointe sur le premier élément de 'b' (un pointeur sur b[0]), d'être assigner à 'a'.

Les Chaînes de Caractères

- Il n'y a pas un type de base String en C
- Les tableaux de caractères peuvent être utilisés comme des chaînes de caractères.

La chaîne de caractères “abc” est sauvegardée dans un tableau ‘s’ contenant 4 caractères comme suit :



La chaîne de caractères vide “ ” est sauvegardée comme

On peut initialiser le tableau de char avec une de chaîne de caractères, mais on ne peut pas affecter une chaîne de caractères à un tableau.

```
char s[4] = “abc”;        // OK

char s[4] ;
s = “abc”;                // Faux

s[0] = ‘a’; s[1] = ‘b’; s[2] = ‘c’;
s[3] = ‘\0’;              // Juste
```

Pour bien manipuler les chaînes de caractères on a besoin de la librairie C :

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

Quelques fonctions utiles pour manipuler les chaînes de caractères :

Copier une chaîne de caractères (strcpy)

Prototype : char *strcpy(char *s1, const char *s2);

```
strcpy(s,“abc”);
strcpy(r,s);              // copie “abc” dans ‘r’
```

Si on fait :

```
r = s;
```

Le pointeur ‘s’ peut être copié dans ‘r’, mais l’affectation ne sera plus valable si ‘r’ est déclaré comme char r[4].

Autres fonctions utiles :

Concaténation des chaînes de caractères (strcat)

```
strcat(str1,str2);
```

Ajoute ‘str2’ à la fin de ‘str1’.

Comparaison des chaînes de caractères (strcmp)

```
Strcmp(str1,str2);
```

Retourne les valeurs : <0 si str1 < str2
=0 si str1 = str2
>0 si str1 > str2

Comparaison lexicographique :

```
“abcd” < “abce”
“abc” < “abcd”
```

Taille d’une chaîne de caractères (strlen)

strlen(str) retourne la taille de la chaîne de caractères ‘str’, tout en excluant le caractère nulle ‘\0’.

```
int len;
char str1[10];

len = strlen(“abc”);      // len est 3
len = strlen(“”);         // len est maintenant 0
strcpy(str1, “abc”);
len = strlen(str1);       // len devient 3 ici
```

Utiliser des Objets en C++

Référence : Folk, Zoellick and Riccardi. Sections 1.5.

- Pour plus de détail lire la section dans le livre.

Ci-dessous on trouve un exemple d'une très simple 'classe' en C++ :

```
class Personne
{ public :
        // les membres données
        char Nom[11], Prenom[11], Adresse[16];
        char Ville[16], Province[3], CodePostale[10];
        // méthode
        Personne();      // Constructeur par défaut
};
```

?? Nom, Prenom, … sont des membres

?? La déclaration de l'objet 'p' appartenant à la classe 'Personne' est :
Personne p;

?? p.Nom fait référence à son membre 'Nom'

?? Niveaux d'accès :

1. public
2. private
3. protected

Le C++ inclut des méthodes spéciales appelées 'constructeurs' qui garantissent la bonne initialisation des objets.

Un 'constructeur' n'a pas un type de retour et son nom est identique au nom de la 'classe' : Personne() dans l'exemple ci-dessus.

L'objet est crée de deux façons :

Par la déclaration du variable :

```
Personne p;      // création automatique
```

Par la déclaration d'un pointeur + création dynamique avec l'opérateur 'new' :

```
Personne *p-ptr = new Personne;
```

L'écriture suivante est valable :

```
Personne *p-ptr;
…
p-ptr = new Personne;
```

Dans ce cas, l'accès aux membres peut être effectué comme suit :

```
(*p-ptr).Nom        ou
p-ptr->Nom          ( la seconde façon est plus utilisée)
```

Les deux façons de créations des objets inclurent l'exécution du constructeur de 'Personne'.

?? Le symbole **::** est l'opérateur de résolution de portée, indique que Personne() est un membre de la classe Personne.

?? Noter qu'à l'intérieur du code membre, le membre peut être utilisé sans l'opérateur point (**.**) .

?? Chaque appel à une fonction membre a un argument caché qui est le pointeur à l'objet : **this** .
This->Nom est identique à Nom à l'intérieur du code méthode.

Le code du constructeur Peronne est le suivant :

```
Personne :: Personne()
{       // tous les champs sont initialisés à une chaîne de caractères vide
        Nom[0]='\0'; Prenom[0]='\0'; Adresse[0] = '\0';
        Ville[0]='\0'; Province[0]='\0'; CodePostale[0]='\0';
}
```

Exemple : manipuler des enregistrement C++ à taille fixe

Le programme suivant est discuté dans ce tutorial/lab. Il contient des idées/matériels utiles pour le devoir #1.

```
// lireenr.cpp
#include <fstream.h>
#include <string.h>


#define MAX 100

// une version simplifiée de la classe Personne
class Personne {
        public :
        char Nom[6];
        char Prenom[6];
        char Province[3];
        Personne();
};


Personne :: Personne() {
        Nom[0]='\0'; Prenom[0]='\0'; Province[0] = '\0';
}

//      Lire du stream un enregistrement de taille fixe et le placer dans 'p'
//      Les champs ont les tailles : 5, 5 et 2 respectivement

int LireEnrPersonne(fstream & stream, Personne & p) {
        stream.getline(p.Nom,6);        // lire 5 caractères ou le défaut
        stream.clear();                 // délimiteur '\n'
                                        // si le délimiteur n'est pas trouvé le flag
                                        // est initialisé à "fail", 'clear' efface le flag "fail "

        if (strlen(p.Nom) == 0)  return 0;
        stream.getline(p.Prenom,6); stream.clear();
        stream.getline(p.Province,3); stream.clear();
        return 1;
}
```

```
// Ecrire la donnée dans p au stream
int EcrEnrPersonne(fstream &stream, Personne &p) {
        stream << p.Nom << p.Prenom << p.Province << endl;
}

// Lire les enregistrements du fichier "in.txt" et écrire dans "out.txt" dans l'ordre inverse
// (le  premier enregistrement le dernier, le dernier enregistrement le premier)
int main() {
        fstream infile;
        fstream outfile;

        infile.open("in.txt", ios ::in);
        outfile.open("in.txt", ios  ::in);

        Personne gens[MAX]; int i, n=0;

        if (infile.fail()) {            // si le fichier n'existe pas, abandonner le programme
                cerr << "l'ouverture du fichier a échouée"
                return 0;
        }

        while ((LireEnrPersonne(infile, gens[n]) != 0)  &&  (n<MAX))
                n++;

        for (int i=n-1; i>=0; i--)
                EcrEnrPersonne(outfile, gens[i]);

        infile.close();
        outfile.close();

        return 1;
}
```

Comprendre cet exemple et puis vous pouvez se référer aux appendices : D.5, D.6, D.7, D.8 pour voir comment on fait des lecture/écriture similaires avec l'opérateur de 'surcharge' >> et l'opérateur <<.