Cours de Langage C
Chaînes de caractères

Les variables de type caractère

- En C, un caractère est une variable de type « *caractères* » à savoir de type **char** : type occupant 1 octet (8 bits)

  - **Le premier usage (peu fréquent) d'une variable de type char est de stocker une variable numérique sur 8 bits.**

Exemple :

```
char a,b,c;
c = a + b ;
```

  - **Le second usage (très fréquent) d'une variable de type char est de stocker un caractère.**

Donc : 256 caractères possibles !

Les variables de type caractère

Exemple :

- Les lettres et les chiffres :
  - a,b,c, … , z ou A,B,C, … ,Z ou 1,2,3, … , 9
- des caractères spéciaux :
  - \n (retour chariot) , \t (tabulation)
- les caractères de ponctuation :
  - … : , ;
- les opérations :
  - + - / *
- etc.
- et aussi : des caractères non affichables :
  - EOF (End of File)

Les variables de type caractère

Pour qu'une variable de type numérique puisse stocker des caractères, on dispose d'une table de transcodage : un code (entre 0 et 255) correspond à un caractère.
Il s'agit de la **table ASCII** (*American Standard Code for Information Interchange*)

Exemple (on a noté les caractères entre ' ') :

'a' → $61_h = 97_d$!! Aucun intérêt à le **mémoriser**

'z' → $7A_h = 122_d = 61_h + 25_d$

'A' → $41_h = 65_d$

'Z' → $5A_h = 90_d$

'0' → $30_h = 48_d$!! Le caractère 0 n'a pas le code 0

'9' → $39_h = 57_d = 48_d + 9_d$

C'est le compilateur qui utilisera les codes Ascii Le programmeur n'a presque jamais à les connaître.

« Qui » se sert de ces codes ? → Essentiellement les fonctions printf , scanf, …

Les variables de type caractère

- **Déclaration / Initialisation/ Utilisation des variables caractères**

Déclaration :

```
char Caract ;  // RIEN SUR LE CONTENU : caractère
               // ou bien nombre sur 8 bits
```

Déclaration & Initialisation :

```
char Caract = 'A' ; // LE CONTENU de la variable est
                    // code Ascii de la lettre A
```

Utilisation :

```
Caract ='A' ;
printf("%c", Caract) ; /* Le printf formaté utilise
                          Caract comme un code Ascii
                       pour afficher la lettre A */
```

Les variables de type caractère

Remarque 1

On peut afficher le code Ascii d'un caractère :

```
printf("%d" , Caract) ; /* Le printf formaté affiche la valeur du code Ascii du caractère A contenu dans la variable Caract donc affiche 65 en décimal sur la console */
```

Remarque 2

Les ‘ ’ sont indispensables pour indiquer au système qu'il s'agit d'un caractère.

```
char Caract ='A' ; // Caract <- code Ascii de
                   // la lettre A
char Caract = A ; // Caract <- valeur de la
                  // variable (ou constante
          // symbolique) A
```

Remarque 3

On peut faire des calculs sur les variables caractères :

→***Exercice : Comment convertir les majuscules en minuscules ?***

Les chaînes de caractères

En C, une chaîne de caractères est un **tableau** de caractères (donc de type char) dont le dernier élément est le caractère nul, noté ' \0 '.

Déclaration : Comment disposer d'un emplacement pour y ranger une chaîne ?

Statiquement :

```
char chaine1[21] ;
```

Permet de ranger une chaîne d'au plus 20 caractères + son '\0 'de fin

Dynamiquement :

```
char *chaine2;
int n = 20 ;
Chaine2 = malloc(n+1) ;
```

A l'adresse chaine2, on pourra ranger une chaîne d'au plus 20 caractères compte tenu de son zéro de fin

Les chaînes de caractères

Initialisation : Comment initialiser une chaîne lors de sa réservation ?

```
char chaine0[20] ;
chaine0= ''bonjour '' ;   INTERDIT

char chaine1[20] = " bonjour "; AUTORISÉ
```

Attention : " " et non pas ''

→ Réservation de 20 octets à partir de l'adresse ***chaine1***

→ C'est le compilateur (et surtout pas le programmeur) qui rajoutera le caractère '\0' en fin de tableau chaîne de caractères.

→ Le programmeur a « juste » à réserver l'espace nécessaire à la chaîne

Initialisation : Comment initialiser une chaîne en cours de programme ?

→ Voir un peu plus loin …

Fonctions manipulant des chaînes

Utilisation : Pour copier le contenu de chaine2 dans l'emplacement pointé par chaine1, il faut utiliser la fonction **strcpy** et surtout pas le symbole d'affectation =

```
#include <string.h>  /* FONCTIONS de traitement de chaînes
                        de caractères */
char    chaine1[100]="abc";
char    chaine2[100]="def";
strcpy(chaine1,chaine2); // SURTOUT pas chaine1 = chaine2
printf("%s\n%s\n",chaine1,chaine2);
```

→ Le printf formaté utilise le format **%s** pour les chaines de caractères (« *string* »)

→ Il affichera caractère après caractère jusqu'au symbole de fin de chaîne : **'\0'**

Il existe de nombreuses fonctions permettant de manipuler les chaînes de caractères. Elles se trouvent dans la bibliothèque <string.h>

Fonctions manipulant des chaînes

Utilisation : Exemples de fonctions de la bibliothèque <string.h>

- **`strlen`** : longueur d'une chaîne de caractères
- **`strcmp`** : comparaison de 2 chaînes
- **`strchr`** : recherche d'un caractère dans une chaîne
- **`strcat`** : concaténation de chaînes de caractères
- **`sprintf, sscanf`** : équivalents de printf et scanf

} voir poly

→ ***Quel est l'algorithme de calcul de la longueur d'une chaîne de caractères ?***

Exemple : construction d'une chaîne de caractères

```
char    nom[100],chemin[100],racine[100],extension[100];

scanf("%s",chemin);
scanf("%s",racine);
scanf("%s",extension);

strcpy(nom,chemin);

strcat(nom,"/");
strcat(nom,racine);
strcat(nom,".");
strcat(nom,extension);
printf("%s\n",nom);

strcpy(nom,"\0"); printf("%s\n",nom);

sprintf(nom,"%s/%s.%s\n",chemin,racine,extension);
printf("%s\n",nom);
```

Exemple

```
char    nom[100],chemin[100],racine[100],extension[100];
char   *ad;

sprintf(nom,"%s/%s\.%s\n", chemin,racine,extension);
printf("%s\n", nom);

printf("%c\n", *nom);
printf("%c\n", *(nom+2));

ad = strrchr(nom,'.');

printf("%s",ad);
printf("%s",ad+1);

*ad = '\0';
printf("%s\n",nom);
```