INF155 — SÉANCE 8
LES CHAINES DE CARACTÈRES

Anis Boubaker, Ph.D.
Maître d'enseignement
École de Technologie Supérieure

PLAN DE LA SÉANCE

- Chaine de caractères - Présentation
- Chaine de caractères = tableau = pointeur
- Initialisation d'une chaine de caractères
- Les fonctions de la librairie standard

CARACTÈRES

- Le caractère est l'un des types de base du langage C.
- Un caractère est un entier codé sur un octet
- Afficher un caractère revient à afficher le caractère correspondant dans la table ASCII:

```
char un_char = 65;
printf("%c", un_char);  //Affiche A
```

CARACTÈRES

- Nous assignons généralement une valeur caractère à une variable de type char, plutôt qu'un nombre:

```
char un_char = 'a'; //Utilisation d'apostrophes!
```

- Cependant, c'est le code ASCII du caractère qui est réellement stocké en mémoire:

```
char un_char = 'a';
printf("%d", a);  //Affiche 97
```

CHAINE DE CARACTÈRES

- Une chaine de caractères est un regroupement de caractères

- Permet de stocker des mots, phrases, paragraphes, etc.

- Exemple:

  INF155

  Allo le monde!

LES CHAINES DE CARACTÈRES EN C

- En C, une chaine de caractères n'est qu'un tableau de caractères.

- Pour déclarer une chaine de caractères:

Déclaration:

```
char identifiant[taille];
```

- Exemple:

```
char une_chaine[100]; //Chaine de caractères pouvant
                      //contenir 100 caractères max.
```

INITIALISATION

- Pour initialiser une chaine de caractères, on peut utiliser la méthode d'initialisation d'un tableau:

```
char une_chaine[]={'C',' ','i','s',' ','F','u','n','!'};

// Rappel! c'est équivalent à:  char une_chaine[9]= … ;
```

- Pour afficher la chaine de caractères, on utilise le **code de formatage %s**:

```
printf("%s", une_chaine);
```

- Malheureusement, ça ne fonctionne pas tel que prévu ☹

ZÉRO BINAIRE TERMINAL

- En C, toute chaine de caractères doit se terminer par un zéro binaire, qui correspond au caractère '\0'
- Ce zéro binaire permet d'identifier la fin de la chaine de caractères
- Cela remplace le paramètre sur le nombre d'éléments effectifs du tableau.

```
char une_chaine[10]={'C',' ','i','s',' ','F','u','n','!','\0'};
```

INITIALISATION – PRISE 2

- Initialiser une chaine de caractères comme on initialise un tableau est pénible!

- Il existe une façon d'initialiser une chaine de caractères de façon littérale, avec les guillemets (" … ") :

```
char une_chaine[] = "Allo le monde!";
```

- Cette méthode ajoute le zéro binaire terminal automatiquement.

MODIFIER UNE CHAINE DE CARACTÈRES

- Pour modifier une chaine de caractères, il suffit de modifier les caractères à l'indice voulu.

```
char une_chaine[] = "Allo le monde!”;

une_chaine[4] = '-’;
une_chaine[7] = '-’;

printf(“%s”, une_chaine); //Allo-le-monde;
```

MODIFIER UNE CHAINE DE CARACTÈRES

- **Problème:** En déclarant notre chaine de caractères comme un tableau, le compilateur a initialisé la mémoire à la taille de la chaine et le pointeur vers le début de cette zone mémoire est fixe (constant).

- Donc, il n'est pas possible de faire ceci:

```
char une_chaine[] = "Allo le monde!";
une_chaine = "Comment ça va?";
```

INITIALISATION – PRISE 3

- Nous savons également qu'un tableau est un pointeur!
- Nous pouvons donc déclarer une chaine de caractères comme un pointeur de caractère:

```
char *une_chaine = "Allo le monde!";
```

- Nous pouvons donc faire ceci:

une_chaine = "Comment ça va?";

CHAINE DE CARACTÈRES COMME POINTEUR

- **Problème:** En initialisant une chaine de caractères déclaré comme un pointeur:
  - Nous pouvons changer la chaine de caractères,
  - MAIS, les chaines ne sont pas modifiables;

```
char *une_chaine = "Allo le monde!”;

une_chaine[4]=‘-’; //Erreur!
*(une_chaine+4) = ‘-’; //Erreur!
```

SAISIE D'UNE CHAINE DE CARACTÈRES

- Pour lire une chaine de caractères depuis la console, on peut utiliser la fonction scanf. Le code de formatage est le %s:

```
char chaine_a_lire[100];

scanf("%s",chaine_a_lire);
```

- **Attention:**
  - Le scanf s'arrêtera au première caractère blanc (espace, retour à la ligne)
  - Le scanf ne vérifie pas la taille de la chaine saisie (possible dépassement de mémoire!)
  - Le scanf ajoute automatiquement le zéro binaire à la fin.

SAISIE D'UNE CHAINE DE CARACTÈRES

- Attention – Ceci cause une erreur!

```
char *chaine_a_lire;

scanf("%s",chaine_a_lire);
```

- Sauriez-vous l'expliquer?

SAISIE D'UNE CHAINE DE CARACTÈRES

- La fonction fgets permet de lire une chaine de caractères en spécifiant la taille maximale:

```
char *fgets ( char * str, int num, FILE * stream );
```

- Exemple:

```
char une_chaine[100];
fgets(une_chaine, 100, stdin);
```

- Le fgets s'arrête quand un retour chariot est saisi
- La retour chariot fait partie de la chaine de caractères! ☹

17

FONCTIONS DE LA LIBRAIRIE STANDARD POUR LES CHAINES

LIBRAIRIE STRING.H

- La librairie <string.h> contient plusieurs fonctions très utiles pour la manipulation des chaines de caractères.

- Parmi les opérations que l'on peut réaliser grâce à ces fonctions:
  - Copier une chaine de caractères dans une autre
  - Comparer des chaines de caractères
  - Concaténer des chaines de caractères
  - etc.

STRLEN

- La fonction **strlen** permet d'obtenir le nombre de caractères d'une chaine de caractères

```
char une_chaine[100]="Une chaine de 27 caractères";
int nb_chars;

nb_chars = strlen(une_chaine);
```

STRCPY

- La fonction **strcpy** copie une chaine dans une autre.

char * strcpy (char * destination, const char * source);

- Exemple:

```
char *chaine_source = “Salut le monde!”;
char la_copie[100];

strcpy(la_copie, chaine_source);
```

STRNCPY

- La fonction strcpy copie une chaine source vers une chaine destination, en supposant que la chaine destination est assez grande pour contenir la source.
- **strncpy** permet de spécifier le nombre maximal de caractères à copier dans la destination:

char * strncpy (char * destination, const char * source, size_t num);

- Exemple:

```
char *chaine_source = “Salut le monde!”;
char la_copie[100];

strncpy(la_copie, chaine_source, 100);
```

STRCAT

- La fonction **strcat** concatène deux chaines de caractères (colle une chaine à une autre)

char * strcat (char * destination, const char * source);

- La chaine ***source*** est ajoutée à la fin de la chaine ***destination***.
- Exemple:

```
char chaine1[100] = “Allo”;
char *chaine2=“ le monde!”;

strcat(chaine1, chaine2);
printf(“%s”, chaine1); //Allo le monde!
```

SPRINTF

- La fonction **sprintf** permet de générer une chaine de caractères en utilisant des codes de formatage (à la printf)

int sprintf(**char** * destination, **const char** * format, …);

- Exemple:

```
char resultat[255];
char ch1[100] = “Allo”;
char *ch2=“le monde”;

sprintf(resultat, "%s tout %s %d fois!",ch1,ch2,10);
printf(“%s”,resultat); //Allo tout le monde 10 fois!
```

STRCMP

- La fonction **strcmp** compare deux chaines de caractères:

int strcmp (const char * str1, const char * str2);

- La fonction renvoie une valeur:
  - nulle (==0) si les deux chaines de caractères sont identiques
  - positive si str1 > str2
  - négative si str1 < str2

CONVERSION DE CHAINE EN NOMBRE

- Une chaine de caractère contenant des caractères numériques (ex.:“1234”) peut être convertie en nombre entier ou réel.
- Deux fonctions sont proposées:
  - atoi : convertit une chaine en un entier:

```
char *une_chaine = “123”
int c;

c = atoi(une_chaine);

printf(“c = %d”, c); //Affiche c = 123
```

  - atof: convertit une chaine en un nombre réel.

L'INTRA 2 LA SEMAINE PROCHAINE!