Les chaînes de caractères en C pour les nuls

Module FAR

Polytech Montpellier – IG3

David Delahaye

Déclaration

- Les chaînes de caractères sont des tableaux de caractères (type char) :

  char *ma_chaine;

- La déclaration ne permet pas d'utiliser la chaîne de caractères, il faut lui allouer de la place en mémoire.
- Cette allocation est manuelle.

Allocation mémoire

- L'allocation version longue est la suivante :

```
char *ma_chaine;
ma_chaine = malloc(10 * sizeof(char));
```

- Alloue une chaîne de 10 caractères en mémoire.

Allocation mémoire

- L'allocation version courte est la suivante :

  char ma_chaine[10];

- Alloue une chaîne de 10 caractères en mémoire.
- Complètement équivalent à la version précédente.

Initialisation

- L'initialisation peut être faite manuellement :

```
char ma_chaine[10];
ma_chaine[0] = 'I';
ma_chaine[1] = 'G';
ma_chaine[2] = '3';
ma_chaine[3] = '\0';
printf("%s\n", ma_chaine);
```

- Les chaînes doivent être terminées par '\0'.
- Le '\n' vide le buffer de sortie.

Initialisation

- L'initialisation peut être faite de manière compacte :

```
char ma_chaine[10] = "IG3";
printf("%s\n", ma_chaine);
```

- Ne peut être fait qu'à l'initialisation :

```
char ma_chaine[10];
ma_chaine = "IG3";
```

Saisie

- La « bonne » façon de saisir une chaîne de caractères est d'utiliser la fonction « fgets » (et pas « scanf ») :

```
char ma_chaine[10];
fgets(ma_chaine, 10, stdin);
printf("Chaine = %s\n", ma_chaine);
```

- « fgets » ne lira que 9 caractères ici et placera le '\0' à la fin de la chaîne. Si la saisie est plus longue, la chaîne sera tronquée.

Saisie

- Si la chaîne fait moins de 9 caractères, le '\n' sera placé dans la chaîne (avant le '\0'). La plupart du temps, on souhaitera le retirer comme suit :

```
char ma_chaine[10];
fgets(ma_chaine, 10, stdin);
char *pos = strchr(ma_chaine, '\n');
*pos = '\0';
printf("Chaine = %s\n", ma_chaine);
```

Longueur

- La longueur d'une chaîne de caractères se calcule de la manière suivante :

  ```
  char ma_chaine[10] = "IG3";
  printf("Longueur = %d\n", (int)strlen(ma_chaine));
  ```

- La fonction strlent rend un objet de type size_t (d'où la nécessité de caster).

Copie

- La copie d'une chaîne de caractères se fait de la manière suivante :

```
char ma_chaine[10] = "IG3", copie[10];
strcpy(copie, ma_chaine);
printf("Copie = %s\n", copie);
```

- La chaîne « copie » doit être assez longue pour contenir la chaîne « ma_chaine » sous peine de débordement de mémoire.

Concaténation

- La concaténation de deux chaînes de caractères se fait de la manière suivante :

```
char ma_chaine[20] = "Bonjour ",
        ma_chaine2[10] = "IG3";
strcat(ma_chaine, ma_chaine2);
printf("Copie = %s\n", ma_chaine);
```

- La chaîne « ma_chaine » doit être assez longue pour contenir la concaténation des deux chaînes sous peine de débordement de mémoire.

Comparaison

- La comparaison de deux chaînes de caractères se fait de la manière suivante (pas avec « == »!) :

```
char ma_chaine[10] = "IG3",
        autre_chaine[10] = "IG3";
printf("Comparaison = %d\n",
          strcmp(ma_chaine, autre_chaine));
```

- Si « strcmp » rend 0, les chaînes sont identiques, sinon elles sont différentes.

Recherche de caractères

- La recherche d'un caractère dans une chaîne de caractères se fait de la manière suivante :

```
char ma_chaine[10] = "IG3";
char *pos = strchr(ma_chaine, 'G');
printf("Chaine = %s\n", pos);
```

- Recherche le caractère concerné et rend un pointeur sur ce caractère dans la chaîne initiale (rend NULL si le caractère n'est pas dans la chaîne).
- Le programme affiche « G3 ».

Recherche d'une chaîne de caractères dans une autre

- La recherche d'une chaîne de caractères dans une autre se fait de la manière suivante :

```
char ma_chaine[20] = "IG3 for ever";
char *pos = strstr(ma_chaine, "for");
printf("Chaine = %s\n", pos);
```

- Rend un pointeur sur la chaîne recherchée dans la chaîne initiale (rend NULL en cas d'échec).
- Le programme affiche « for ever ».

Découpage en « tokens »

- Le découpage en « tokens » d'une chaîne de caractères se faire de la manière suivante :

```
char ma_chaine[20] = "IG3 for ever";
char *tok = strtok(ma_chaine, " ");
printf("Token 1 = %s\n", tok);
tok = strtok(NULL, " ");
printf("Token 2 = %s\n", tok);
tok = strtok(NULL, " ");
printf("Token 3 = %s\n", tok);
```

Découpage en « tokens »

- Le premier argument est la chaîne à découper en « tokens ». Au premier appel, on mettra la chaîne, puis on mettra « NULL » pour continuer à découper la chaîne.
- Le deuxième argument est le séparateur.
- Affichage du programme :

Token 1 = IG3
Token 2 = for
Token 3 = ever

Écriture formatée dans une chaîne de caractères

- On peut former une chaîne de caractères avec des formats à la manière de « printf » comme suit :

```
char ma_chaine[20];
sprintf(ma_chaine, "IG3 %d !", 2016);
printf("%s\n", ma_chaine);
```

- La chaine « ma_chaine » doit être assez grande sous peine de débordement de mémoire.
- Le programme affiche « IG3 2016 ! ».