Chaînes de caractères

Déclaration d'une chaîne de caractères

Les chaînes de caractères sont des constantes du langage :

```
"bonjour"
```

ou

```
"bon\
 jour"
```

Déclaration d'une chaîne de caractères

Les chaînes de caractères sont des constantes du langage :

```
"bonjour"
```

ou

```
"bon\
 jour"
```

En pratique, une chaîne de caratère est convertie en tableau de caractères qui termine par le caractère nul `\0`.
Pour déclarer une variable de ce type :

```
char str[256];
char* str;
```

Initialisation des chaînes de caractères

`char str[8] = "bonjour"` (prévoir la place pour le caractère nul) ou
`char str[] = "bonjour"` ou `char* str = "bonjour"`

Initialisation des chaînes de caractères

`char str[8] = "bonjour"` (prévoir la place pour le caractère nul) ou
`char str[] = "bonjour"` ou `char* str = "bonjour"`

La dernière déclaration est spécifique aux chaînes de caractères : le compilateur alloue un tableau de 8 caractères et une variable de type pointeur qui contient l'adresse du premier élément du tableau.

Exemple 1

```
char* str1 = "bonjour\n";
printf("%s", str1);
str1 = "coucou\n";
printf("%s", str1);
```

Exemple 1

```
char* str1 = "bonjour\n";
printf("%s", str1);
str1 = "coucou\n";
printf("%s", str1);


/* code erroné
char str2[] = "bonjour\n";
printf("%s", str2);
str2 = "coucou\n";
printf("%s", str2);
*/
```

Exemple 2

```
char* adr;
adr = "bonjour";
while(*adr){
    printf("%c", *adr);
    adr++;
}
```

Arguments de la ligne de commande

Un programme peut être exécuté avec des arguments :

```
./addition 2 5
```

Arguments de la fonction main

La fonction `main` peut avoir 2 arguments :

- `argc` : compte le nombre de chaînes de caractères séparées par des blancs lors de l'exécution du programme (au moins 1)

Arguments de la fonction main

La fonction `main` peut avoir 2 arguments :

- `argc` : compte le nombre de chaînes de caractères séparées par des blancs lors de l'exécution du programme (au moins 1)
- `argv` : tableau de taille `argc` qui contient, dans l'ordre, ces chaînes de caractères.

Arguments de la fonction main

La fonction `main` peut avoir 2 arguments :

- `argc` : compte le nombre de chaînes de caractères séparées par des blancs lors de l'exécution du programme (au moins 1)
- `argv` : tableau de taille `argc` qui contient, dans l'ordre, ces chaînes de caractères.

Arguments de la fonction main

La fonction `main` peut avoir 2 arguments :

- `argc` : compte le nombre de chaînes de caractères séparées par des blancs lors de l'exécution du programme (au moins 1)
- `argv` : tableau de taille `argc` qui contient, dans l'ordre, ces chaînes de caractères.

Si le programme `addition` contient la ligne suivante

```
int main(int argc, char** argv)
```

alors l'appel `./addition 2 5` fait que

- `argc` vaut 3
- `argv[0]` est la chaîne `"addition"`
- `argv[1]` est la chaîne `"2"`
- `argv[2]` est la chaîne `"5"`

Fonctions de conversion (librairie stdlib.h)

- `int atoi(char*)` convertit une chaîne en entier

Fonctions de conversion (librairie stdlib.h)

- `int atoi(char*)` convertit une chaîne en entier
- `double atof(char*)` convertit une chaîne en double

Fonctions de conversion (librairie stdlib.h)

- `int atoi(char*)` convertit une chaîne en entier
- `double atof(char*)` convertit une chaîne en double

Fonctions de conversion (librairie stdlib.h)

- `int atoi(char*)` convertit une chaîne en entier
- `double atof(char*)` convertit une chaîne en double

```
//programme addition
int main(int argc, char** argv){
    int a,b;
    a = atoi(argv[1]);
    b = atoi(argv[2]);
    printf("%d\n", a+b);
    return 0;
}
```

Fonctions de la librairie string.h

- copier : memcpy, memmove, strcpy, strncpy ;
- concaténer : strcat, strncat ;
- comparer : memcmp, strcmp, strcoll, strncmp ;
- rechercher : memchr, strchr, strcspn, strpbrk, strrchr, strspn, strstr, strtok ;
- initialiser : memset ;
- calculer la longueur : strlen ;

- `int strlen(char *s)` : retourne la taille de la chaîne (le caractère nul n'est pas compté) ;

- `int strlen(char *s)` : retourne la taille de la chaîne (le caractère nul n'est pas compté) ;
- `int strcmp(char *s1, char *s2)` : retourne une valeur positive, nulle ou négative selon que `s1` est plus grand, égal ou plus petit que `s2` (ordre lexicographique) ;

- `int strlen(char *s)` : retourne la taille de la chaîne (le caractère nul n'est pas compté) ;
- `int strcmp(char *s1, char *s2)` : retourne une valeur positive, nulle ou négative selon que `s1` est plus grand, égal ou plus petit que `s2` (ordre lexicographique) ;
- `char* strcat(char *dest, char *src)` : ajoute la chaîne `src` à la fin de `dest` ; les chaînes doivent être disjointes et la mémoire disponible ; renvoie un pointeur sur `dest` si succès, `NULL` sinon ;

- `void* memcpy(void* dest, void* src, int n)` : copie `n` octets de l'adresse `src` à l'adresse `dest` ; renvoie un pointeur sur `dest` ;

- `void* memcpy(void* dest, void* src, int n)` : copie `n` octets de l'adresse `src` à l'adresse `dest` ; renvoie un pointeur sur `dest` ;
- `char* strstr(char* haystack, char* needle)` : cherche la première occurence de `needle` dans la chaîne `haystack`, le caractère nul n'est pas comparé ; retourne l'adresse de la première occurence dans `haystack`, et `NULL` si elle n'existe pas ;