

# Programmation C

## Chaînes de caractères

ING1-GI

CY Tech

2020-2021

# Introduction

# Introduction

## Présentation

- Type `char` : permet de stocker une lettre
- Chaîne de caractères : tableau de caractères
- Déjà utilisé : `printf("Hello World");`
- "Hello World" : Chaîne de 12 caractères (11 + `'\0'`)
- Déclaration : `char texte[12]`

# Introduction

## Entrée - Sorties

- Saisie au clavier : `scanf ("%s",texte);`
- Problème : saisie de l'espace difficile
  - ▶ Utilisation d'un autre format `"%[^\n]",texte`
  - ▶ Utilisation d'une autre fonction de saisie : `getchar`, `getline`, ...
- `texte` : tableau statique

# Introduction

## Autres "connaissances"

- Fonction `int main (int argc, char** argv)`
- `argv` : Tableau de chaîne de caractères
  - ▶ `argv` : Tableau de `char*`
  - ▶ `char*` : Chaîne de caractères
  - ▶ d'où `argv` : Tableau de chaîne de caractères
  - ▶ Contient la liste des arguments de la ligne de commande
  - ▶ `argv[0]` : Nom du programme
  - ▶ `argv[n*]` : Paramètre du programme (ligne de commande)
- `argc` : Nombre d'argument de la ligne de commande

# Manipulation

- Comment saisir une chaîne de caractères sans avoir à préciser la taille

### Listing 1 – Problème de saisie

```
char str_nom[31]; // Declaration
char* str_prenom; // Declaration
... = scanf("%s",str_nom); // Ok
... / ...
... = scanf("%s",str_prenom); // Probleme!
```

- `str_prenom` : pointeur non alloué
- Besoin d'allouer avant la saisie ???

# Manipulation

## getline

- Utilisation de la fonction `getline`
- Définie dans `stdio.h`
- Prototype : `ssize_t getline(char **lineptr, size_t *n, FILE *stream);`
- Si `*lineptr == NULL`  $\Leftrightarrow$  `getline` réalise l'allocation
- Sinon, reallocation possible



# Manipulation

## Exemple

```
#include <stdio.h>

int main(int argc, char** argv) {
    char* line; // Ligne a lire
    size_t nb1; // Nb de caracteres alloues
    int nb2;    // Taille reelle de la chaine

    printf("%d_\n", (int)nb1);
    line = NULL;
    nb2 = getline(&line, &nb1, stdin);
    printf("%d_\t%d_\t%s_\n", (int)nb1, nb2, line);

    return (0);
}
```



# Manipulation

## Manipulation de chaîne

- Inclusion de la bibliothèque : `string.h`
- Copie :
  - ▶ `strcpy` : copie vers une chaîne, une chaîne
  - ▶ `strncpy` : copie vers une chaîne, une chaîne d'au plus  $n$  caractères
  - ▶ `strcpy(texte, "Elisabeth Ranisavljevic");` : permet l'initialisation
  - ▶ `strncpy(texte, "Elisabeth Ranisavljevic", 9);` : copie les 9 premiers caractères
  - ▶ Attention au marqueur de fin de chaîne

# Manipulation

## Manipulation de chaîne

- Inclusion de la bibliothèque : `string.h`
- Longueur :
  - ▶ `strlen` : Calcul la longueur de la chaîne sans compter le marqueur de fin
  - ▶ `strlen("Elisabeth Ranisavljevic") : 23`
- Concaténation :
  - ▶ `strcat` : Ajoute à la fin d'une chaîne une autre chaîne
  - ▶ `strncat` : Idem mais  $n$  caractères ajoutés
  - ▶ `strcat("Elisabeth", " Ranisavljevic")`
  - ▶ Attention à la taille : risque de débordement ...

# Manipulation

## Manipulation de chaîne

- Inclusion de la bibliothèque : `string.h`
- Comparaison :
  - ▶ `strcmp` : Compare deux chaînes
  - ▶ `strcmp("Elisabeth", "Hervé")`
  - ▶ Renvoie 0 si les deux sont égales
  - ▶ Renvoie un nombre négatif si "Elisabeth" est avant "Hervé"
  - ▶ Renvoie un nombre positif si "Hervé" est avant "Elisabeth"
  - ▶ Ordre lexico-graphique : Attention majuscule/minuscule

# Manipulation

## Manipulation de chaîne

- Inclusion de la bibliothèque : `string.h`
- Recherche de caractères :
  - ▶ `strchr` : Recherche la première occurrence d'un caractère dans une chaîne
  - ▶ `strrchr` : Recherche la dernière occurrence d'un caractère dans une chaîne
  - ▶ `strstr` : Recherche une chaîne dans une chaîne
  - ▶ `strcasestr` : Recherche une chaîne dans une chaîne en ignorant la casse
  - ▶ `strtok` : Séparation d'une chaîne en fonction d'un délimiteur (cf `split.c`)

# Manipulation

## Manipulation de chaîne

- Inclusion de la bibliothèque : `string.h`
- Création d'une chaîne :
  - ▶ `sprintf` : Permet de créer une chaîne de caractères en utilisant le formalisme de `printf`
- Changement de casse : utilisation de la bibliothèque `ctype.h` :
  - ▶ `toupper` : Majuscule d'un caractère
  - ▶ `tolower` : Minuscule d'un caractère

# Conversion

# Conversion

## Conversion d'un nombre vers une chaîne

- Le plus simple : `sprintf`
- Permet de convertir n'importe quoi vers une chaîne de caractères



# Conversion

## Conversion d'une chaîne vers un nombre

- Avec gestion d'erreurs :
  - ▶ `strtol` : converti une chaîne vers un long int, possibilité de faire une conversion de base
  - ▶ `strtod` : converti une chaîne vers un double
  - ▶ `strtof` : converti une chaîne vers un float
  - ▶ .../..
- Ne pas utiliser `atoi` et `constr` : Ne détecte pas les erreurs !

# Lecture correcte

# Lecture correcte

## Problématique

- Lecture d'un entier : pose problème avec `scanf`
- Que se passe-t-il si l'utilisateur entre 1.22 ?
- Deux possibilités
  - ▶ Ne pas utiliser `scanf` mais `getline`
  - ▶ Ou alors vérifier qu'il ne reste pas de caractères dans le buffer d'entrée après la lecture de `scanf`  $\Leftrightarrow$  peut être fait avec `getchar`