Université de Xidian
Polytech'Sophia

Université de Xidian
2020–2021

Examen de Langage C *(V. Granet)*

Durée : 1h30
Aucun document autorisé
Mobiles interdits

► 1. **Sans utiliser** les fonctions standard de `string.h`, et uniquement à l'aide de la notation de pointeur (donc pas de tableau), écrivez la procédure `inverser` qui inverse une chaîne de caractères. Cette procédure a un paramètre « donnée » qui est la chaîne à inverser, et un paramètre « résultat » qui est la chaîne inversée. L'en-tête de cette procédure est le suivant :

```
// antécédent : à compléter
// conséquent : à compléter
void inverser(const char *s1, char *s2)
```

Par exemple, si `s1` est égale à *"hellow"*, après inversion, `s2` est égale à *"wolleh"*.

```
// antécédent : s1 chaîne de caractères à inverser
// conséquent : s2 chaîne de caractères inversion de s1
void inverser(const char *s1, char *s2) {

  const char *p = s1;
  // se placer à la fin de s1
  while (*s1) s1++;
  // parcourir la chaîne en sens inverse et copier
  // ses caractères dans s2
  while (--s1>=p)
    *s2++ = *s1;
  // écrire le caractères de fin de chaîne dans s2
  *s2 = '\0';
}
```

.....

► 2. Toujours sans utiliser les fonctions standard de `string.h`, et uniquement à l'aide de la notation de pointeur, on veut maintenant récrire la procédure `inverser` précédente sous forme d'une fonction, c'est-à-dire qui renvoie la chaîne inversée. Il faudra donc créer une nouvelle chaîne (`malloc`). Écrivez la fonction `inverser` dont l'en-tête est le suivant :

```
// antécédent : à compléter
// conséquent : à compléter
char *inverser(const char *s)
```

```
// antécédent : s chaîne de caractères à inverser
// rôle : renvoie l'inverse de s
char *inverser(const char *s) {
  int lg = 0;
  const char *p = s;
  // calculer la longueur de s
  while (*s++) lg++;
  // allouer la nouvelle chaîne de longueur lg
  char *si = malloc(lg+1);
  char *q = si;
```



```
  // parcourir la chaîne en sens inverse et copier
  // ses caractères dans si
  s--;
  while (--s>=p)
    *si++ = *s;
  // écrire le caractères de fin de chaîne dans si
  *si = '\0';
  return q;
}
```

Il est également possible d'utiliser la première version de `inverser` comme suit :

```
// antécédent : s chaîne de caractères à inverser
// rôle : renvoie l'inverse de s
char *inverser2(const char *s) {
  int lg = 0;
  const char *p = s;
  // calculer la longueur de s
  while (*s++) lg++;
  // allouer la nouvelle chaîne de longueur lg
  char *si = malloc(lg+1);
  // inverser p
  inverser(p, si);
  return si;
}
```

.....

► 3. Écrivez un programme qui prend en paramètre deux noms de fichier. Le premier nom correspond à un fichier de texte qui contient une suite (éventuellement vide) de mots. Les mots ont une longueur maximale de 10 caractères, et sont placés un par ligne.

Votre programme devra lire tous les mots dans le premier fichier, et les écrire de façon inversée (à l'aide de la fonction `inverser` précédente) dans un second fichier de texte dont le nom est le second paramètre programme. Par exemple, si un fichier `f1` contient :

```
aert
hello
bonjour
maitenant
```

L'exécution du programme `inverserFich f1 f2` produira le fichier f2 :

```
trea
olleh
ruojnob
tnanetiam
```

Note : votre programme devra faire toutes les vérifications nécessaires.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

#define LGMAXMOT 10

int main(int argc, char *argv[])
{
  // vérifier le bon nombre de paramètres programme
```

```
  if (argc!=3) {
    fprintf(stderr, "Usage : %s f1 f2\n", argv[0]);
    exit(EXIT_FAILURE);
  }
  // le nombre de paramètres est correct
  // ouvrir le fichier de mots argv[1] en lecture
  FILE *in;
  if ((in = fopen(argv[1], "r")) == NULL) {
    perror(argv[1]);
    exit(errno);
  }
  // ouvrir le fichier de mots inversés argv[2] en écriture
  FILE *out;
  if ((out = fopen(argv[2], "w")) == NULL) {
    perror(argv[2]);
    exit(errno);
  }
  // lire le fichier de mots in et écrire les mots de façon inversée dans out
  char mot[LGMAXMOT+1];
  while (fscanf(in, "%s", mot)>0)
    fprintf(out, "%s\n", inverser(mot));
  // fermer les fichiers
  fclose(in);
  fclose(out);

  return EXIT_SUCCESS;
}
```