TP6_INF203

Groupe: INF2

Binôme: Shaghayegh HAJMOHAMMADKASHI

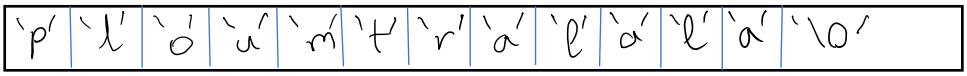
Kaiwen ZHENG

Terminal : cd INF203

./TP1/scripts/installeTP.sh 6

cd TP6

[a]:



[b] :

```c
#include <stdio.h>
#include <string.h>
#include "generer_entier.c"
#include "bouts_de_phrases.c"

/* longueur de la chaine passee en parametre */
/*1*/
unsigned long mon_strlen(char *ch) {
    int i=0 ;
    while (ch[i] != '\0') {
        i++;
    }
    return i ;
}

/*2*/
void mon_strcpy(char *destination, char *source){
    int i = 0;
    while (source[i] != '\0') {
        destination[i] = source[i];
        i++;
    }
    destination[i] = '\0';
}

/*3*/
void mon_strcat(char *destination, char *source) {
    // On se place au bout de la chaîne destination
    while (*destination != '\0') {
        destination++;
    }
```

```
    // On copie les caractères de la chaîne source à la suite de la chaîne
destination
    while (*source != '\0') {
        *destination = *source;
        destination++;
        source++;
    }
    // On ajoute le caractère de fin de chaîne
    *destination = '\0';
}

/*4
int mon_strcmp(char *chaine1, char *chaine2) {
    int i = 0;
    while (chaine1[i] != '\0' && chaine2[i] != '\0') {
        if (chaine1[i] != chaine2[i]) {
            return 1;
        }
        i++;
    }
    if (chaine1[i] != chaine2[i]) {
        return 1;
    }
    return 0;
} */

int mon_strcmp(char *chaine1, char *chaine2) {
    int i = 0;
    while (chaine1[i] != '\0' && chaine2[i] != '\0') {
        if (chaine1[i] < chaine2[i]) {
            return -1;
        } else if (chaine1[i] > chaine2[i]) {
            return 1;
        }
        i++;
    }
    if (chaine1[i] == '\0' && chaine2[i] == '\0') {
        return 0;
    } else if (chaine1[i] == '\0') {
        return -1;
    } else {
        return 1;
    }
}

/*1
int main() {
    char chaine[50] ;
```

```
    unsigned long mon_resultat ;

    printf("un mot (pas trop long !) à mesurer ?\n") ;
    scanf("%49s", chaine) ;
    mon_resultat=mon_strlen(chaine) ;
    if (mon_resultat == strlen(chaine) )
        printf("longueur de la chaine %s :%lu\n", chaine, mon_resultat) ;
    else
        printf("non, la longueur de '%s' n'est pas %lu\n", chaine,
mon_resultat) ;

    return 0 ;
}*/

/*2
int main() {
    char source[] = "Hello, world!";
    char destination[20];
    mon_strcpy(destination, source);
    printf("La chaine copiee est : %s\n", destination);
}*/

/*3
int main() {
    char chaine[20] = "ploum";
    mon_strcat(chaine, "tra");
    mon_strcat(chaine, "lala");
    printf("La chaine concaténée est : %s\n", chaine);
}*/

/*4
int main() {
    char chaine1[] = "Hello";
    char chaine2[] = "World";
    int resultat = mon_strcmp(chaine1, chaine2);
    if (resultat == 0) {
        printf("Les chaines sont identiques\n");
    } else {
        printf("Les chaines sont differentes\n");
    }
    return 0;
}*/

int main() {
    char chaine1[50], chaine2[50];
    int resultat;

    printf("Entrez la première chaîne : ");
```

```
    scanf("%s", chaine1);

    printf("Entrez la deuxième chaîne : ");
    scanf("%s", chaine2);

    resultat = mon_strcmp(chaine1, chaine2);

    if (resultat < 0) {
        printf("La première chaîne précède la deuxième dans l'ordre
lexicographique\n");
    } else if (resultat == 0) {
        printf("Les deux chaînes sont identiques\n");
    } else {
        printf("La deuxième chaîne précède la première dans l'ordre
lexicographique\n");
    }

    return 0;
}
```

[c] :

```
#include <stdio.h>
#include <string.h>
#include "generer_entier.c"
#include "bouts_de_phrases.c"

/* longueur de la chaine passee en parametre */
/*1*/
unsigned long mon_strlen(char *ch) {
    int i=0 ;
    while (ch[i] != '\0') {
        i++;
    }
    return i ;
}

/*2*/
void mon_strcpy(char *destination, char *source){
    int i = 0;
    while (source[i] != '\0') {
        destination[i] = source[i];
        i++;
    }
    destination[i] = '\0';
}

/*3*/
```

```c
void mon_strcat(char *destination, char *source) {
    // On se place au bout de la chaîne destination
    while (*destination != '\0') {
        destination++;
    }
    // On copie les caractères de la chaîne source à la suite de la chaîne
destination
    while (*source != '\0') {
        *destination = *source;
        destination++;
        source++;
    }
    // On ajoute le caractère de fin de chaîne
    *destination = '\0';
}

/*4*/
int mon_strcmp(char *chaine1, char *chaine2) {
    int i = 0;
    while (chaine1[i] != '\0' && chaine2[i] != '\0') {
        if (chaine1[i] < chaine2[i]) {
            return -1;
        } else if (chaine1[i] > chaine2[i]) {
            return 1;
        }
        i++;
    }
    if (chaine1[i] == '\0' && chaine2[i] == '\0') {
        return 0;
    } else if (chaine1[i] == '\0') {
        return -1;
    } else {
        return 1;
    }
}

int main() {
    char Sujet[50]="la petite souris" ;
    char Verbe[50]="mange" ;
    char Compl[50]="le gros chat" ;
    char Phrase[150];
    mon_strcat(Phrase, Sujet);
    mon_strcat(Phrase, " ");
    mon_strcat(Phrase, Verbe);
    mon_strcat(Phrase, " ");
    mon_strcat(Phrase, Compl);
    printf("La phrase est : %s\n", Phrase);
}
```

[d] :

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include "generer_entier.c"
#include "bouts_de_phrases.c"

/* longueur de la chaine passee en parametre */
/*1*/
unsigned long mon_strlen(char *ch) {
    int i=0 ;
    while (ch[i] != '\0') {
        i++;
    }
    return i ;
}

/*2*/
void mon_strcpy(char *destination, char *source){
    int i = 0;
    while (source[i] != '\0') {
        destination[i] = source[i];
        i++;
    }
    destination[i] = '\0';
}

/*3*/
void mon_strcat(char *destination, char *source) {
    // On se place au bout de la chaîne destination
    while (*destination != '\0') {
        destination++;
    }
    // On copie les caractères de la chaîne source à la suite de la chaîne
destination
    while (*source != '\0') {
        *destination = *source;
        destination++;
        source++;
    }
    // On ajoute le caractère de fin de chaîne
    *destination = '\0';
}

/*4*/
int mon_strcmp(char *chaine1, char *chaine2) {
    int i = 0;
```

```c
    while (chaine1[i] != '\0' && chaine2[i] != '\0') {
        if (chaine1[i] < chaine2[i]) {
            return -1;
        } else if (chaine1[i] > chaine2[i]) {
            return 1;
        }
        i++;
    }
    if (chaine1[i] == '\0' && chaine2[i] == '\0') {
        return 0;
    } else if (chaine1[i] == '\0') {
        return -1;
    } else {
        return 1;
    }
}

// Fonction qui compte le nombre de mots dans une phrase
int nb_mots(char* phrase, char separateur) {
    int nb_mots = 1;
    int i;
    for (i = 0; i < strlen(phrase); i++) {
        if (phrase[i] == separateur) {
            nb_mots++;
        }
    }
    return nb_mots;
}

/*int main() {
    char Sujet[50]="la petite souris" ;
    char Verbe[50]="mange" ;
    char Compl[50]="le gros chat" ;
    char Phrase[150];
    mon_strcat(Phrase, Sujet);
    mon_strcat(Phrase, " ");
    mon_strcat(Phrase, Verbe);
    mon_strcat(Phrase, " ");
    mon_strcat(Phrase, Compl);
    printf("La phrase est : %s\n", Phrase);
}*/

int main() {
    // Initialisation du générateur de nombres aléatoires
    srand(time(NULL));

    // Génération de 10 phrases aléatoires
    int i;
```

```
    for (i = 0; i < 10; i++) {
        char phrase[200];
        strcpy(phrase, sujet[generer_entier(10)]);
        strcat(phrase, " ");
        strcat(phrase, verbe[generer_entier(10)]);
        strcat(phrase, " ");
        strcat(phrase, complement[generer_entier(10)]);
        printf("%s\n", phrase);
        printf("Nombre de mots : %d\n", nb_mots(phrase, ' '));
    }

    return 0;
}
```

Dans ce main, la fonction nb_mots est utilisée pour compter le nombre de mots dans chaque phrase générée aléatoirement en utilisant l'espace comme séparateur.

[e] :

Le profil de la fonction ne peut pas être lire_joueur(joueur j) car si on passe une variable joueur à cette fonction, la fonction modifiera une copie locale de cette variable, plutôt que la variable originale. Ainsi, les modifications apportées à la copie locale ne seront pas répercutées sur la variable originale une fois que la fonction aura terminé son exécution.

C'est pourquoi, si on souhaite modifier directement une variable existante de type joueur, il faut passer un pointeur vers cette variable, comme c'est le cas dans la fonction lire_joueur(joueur* pj). De cette façon, les modifications apportées à la structure pointée par le pointeur seront répercutées sur la variable d'origine une fois que la fonction aura terminé son exécution.

[f] :

```
#include <stdio.h>

typedef struct {
    char pseudo[20];
    int nb_billes;
} joueur;

joueur atchoum = { "Atchoum", 42 };
joueur dormeur = { "Dormeur", 25 };
joueur grincheux = { "Grincheux", 3 };
joueur joyeux = { "Joyeux", 100 };
joueur prof = { "Prof", 2 };
joueur simplet = { "Simplet", 0 };
joueur timide = { "Timide", 12 };

void afficher_joueur_V2(joueur* pj) {
    printf("%s a %d billes\n", pj->pseudo , pj->nb_billes);
}
```

```c
void afficher_joueur_V1(joueur j) {
    printf("%s a %d billes\n", j.pseudo, j.nb_billes);
}


// void change_bille(joueur* pj){
//     pj->nb_billes = 1000;
// }
// void change_bille_local(joueur j){
//     // Inutile ...
//     j.nb_billes = 2000;
// }

void lire_joueur(joueur* pj) {
    printf("Entrez le pseudo du joueur : ");
    scanf("%s", pj->pseudo);
    printf("Entrez le nombre de billes du joueur :");
    scanf("%d", &pj->nb_billes);
    afficher_joueur_V2(pj);
}

void transferer_V1(joueur j1, joueur j2, int nb) {
    j1.nb_billes -= nb;
    j2.nb_billes += nb;
    afficher_joueur_V1(j1);
    afficher_joueur_V1(j2);
}

void  transferer_V2(joueur* pj1, joueur* pj2, int nb) {
    pj1->nb_billes -= nb;
    pj2->nb_billes += nb;
    afficher_joueur_V2(pj1);
    afficher_joueur_V2(pj2);
}


int main() {

    afficher_joueur_V1(atchoum);
    joueur *pointeurversatchoum = &atchoum;
    afficher_joueur_V2(pointeurversatchoum);
    afficher_joueur_V2(&grincheux);
    printf("\n");
    // change_bille(pointeurversatchoum);
    // change_bille_local(atchoum);
    afficher_joueur_V2(pointeurversatchoum);
    joueur nouveau_joueur;
    // lire_joueur(&nouveau_joueur);
    transferer_V1(atchoum, dormeur, 2);
```

```
    transferer_V2(&atchoum, &dormeur, 2);
    return 0;
}
```

tr -s abc < exemples_tr

[g] :

```
grep -v '^$' fich1 > fich2
```

- grep est une commande qui permet de chercher des motifs dans des fichiers.
- -v est l'option qui inverse la recherche, c'est-à-dire qui affiche les lignes qui ne correspondent pas au motif.
- '^$' est un motif qui correspond aux lignes vides. ^ représente le début de la ligne et $ représente la fin de la ligne, donc '^$' correspond à une ligne qui ne contient rien entre le début et la fin.
- fich1 est le nom du fichier d'origine.
- > est un opérateur de redirection qui permet de rediriger la sortie de la commande vers un fichier plutôt que de l'afficher à l'écran.
- fich2 est le nom du fichier de destination.