

Contrôle continu Langage C

Durée : 1h30

Barème final

Les copies de transparents du cours, le photocopié de L. Granvilliers et les notes manuscrites personnelles sont les seuls documents autorisés.

Exercice 1 (5 pts. (0.5 pt. par erreur))

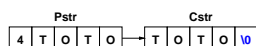
Indiquez les erreurs se trouvant dans le programme ci-dessous et corrigez-les :

```

1. include "stdio.h"
2.
3. void main(void) {
4.   char *chaine;
5.
6.   printf("Votre nom ? ");
7.   scanf("%c",&chaine);
8.
9.   printf("Votre nom à l'envers : ");
10.  for (int i = strlen(chaine); i >=0; --i) {
11.    printf("%c",chaine[i]);
12.  }
13. }
```

Exercice 2 (9 pts. (3 pts. par question))

- Afin de pouvoir manipuler des chaînes de caractères contenant des caractères non imprimables, on a défini un format de chaîne **Pstr** « à la Pascal » où le premier caractère correspond en fait à la taille de la chaîne (voir figure ci-dessous).



- Écrire la fonction `char *pstr2cstr(char *cstr, char *pstr, unsigned int maxsz)` prenant en entrée une chaîne de caractères au format **Pstr** et stockant au plus **maxsz** caractères (y compris le terminateur `'\0'`) dans **cstr** en utilisant le format de chaîne standard de C. La fonction retournera un pointeur sur **cstr**. On suppose que l'espace pointé par **cstr** a été alloué correctement avant l'appel. La fonction devra remplacer chaque caractère non imprimable par '?' (On utilisera la fonction `int isprint(char)` dont la déclaration se trouve dans `ctype.h` pour ce test);
- On souhaite écrire la fonction `mysqrt()` calculant la racine carrée d'un nombre de type **double** en utilisant la formule :

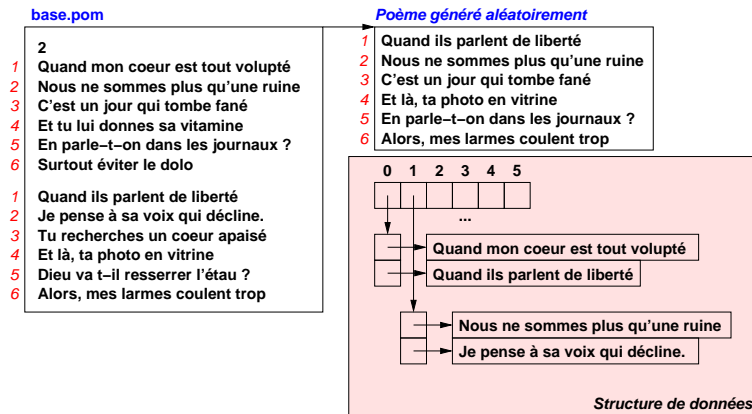
$$\sqrt{a} = U_{\infty}, \quad \text{avec} \quad \begin{cases} U_0 = \begin{cases} 0 & \text{si } a = 0 \\ \frac{a^3 + 3a^2}{3a^2 + a} & \text{sinon} \end{cases} \\ U_n = \frac{U_{n-1}^3 + 3aU_{n-1}}{3U_{n-1}^2 + a} \end{cases}$$

- Écrire la fonction `mysqrt()` en utilisant un nombre d'itérations fixe $n = 4$;

- (b) Écrire le programme stockant dans un fichier `mysqrt.dat` l'erreur absolue Δ de la fonction `mysqrt` définie par $\Delta = |\text{sqrt}(a) - \text{mysqrt}(a)|$ pour a variant de 0 à 100 par incrément de 0.0125. Le fichier devra comporter deux colonnes : en première colonne, la valeur de a et en deuxième colonne, l'erreur absolue correspondante. On affichera un message d'erreur sur le canal d'erreur standard et l'on sortira du programme avec le code d'erreur 1 si le fichier `mysqrt.dat` ne peut être ouvert en écriture.
- Rappel : la fonction « valeur absolue » est `fabs()`.

Exercice 3 (7 pts. (1 pt. par question))

On dispose dans le fichier texte `base.pom` d'une base de poèmes composés de six vers. La première ligne du fichier contient le nombre de poèmes présents. Chaque poème est séparé du suivant par une ligne vide. On veut écrire un programme affichant un poème de six vers créé aléatoirement en piochant le premier vers parmi les premiers vers des poèmes de la base, le deuxième vers parmi les deuxièmes vers, ... comme illustré par la figure ci-dessous.



On décide de lire la base de poèmes et de la stocker en mémoire de façon à minimiser la place requise. Aussi, on allouera dynamiquement le nombre exact de caractères nécessaires pour stocker chaque vers en mémoire.

1. Définir le type `poeme_t` représentant un poème de six vers ;
2. Définir le type `base_vers` représentant une base de vers. Aide : on représentera la base en s'inspirant de la représentation donnée dans la figure ci-dessus ;
3. Écrire la fonction `creer_base()` prenant en argument un `FILE*` et retournant une base de vers de type `base_vers` ;
4. Écrire la fonction `destruire_base()` prenant en entrée une base de vers et libérant toute la mémoire allouée dynamiquement lors de sa création ;
5. Écrire la fonction `creer_poeme()` prenant en entrée une base de vers et retournant un objet de type `poeme_t` contenant un poème créé aléatoirement ;
6. Écrire la fonction `afficher_poeme()` prenant en entrée un poème (type `poeme_t`) et un flux `f` (type `FILE*`) et écrivant le poème dans le flux ;
7. Écrire un programme lisant la base de poèmes `base.pom` et affichant un poème créé aléatoirement à l'écran.

Contrôle continu Langage C

Durée : 1h30

— CORRECTION —

Corrigé exercice 1 (5 pts. (0.5 pt. par erreur))

Les erreurs

1. Ligne 1 : il manque le # devant `include`
2. Ligne 1 : `stdio.h` entre guillemets au lieu de chevrons
3. Ligne 2 : il manque `#include <string.h>` pour déclarer `strlen()`
4. Ligne 3 : `main()` déclaré comme retournant un `void` au lieu de `int`
5. Ligne 7 : `chaine` n'a pas d'espace alloué
6. Ligne 7 : Il faut utiliser `%s` et non `%c`
7. Ligne 7 : Il ne faut pas utiliser le `&` devant `chaine`
8. Ligne 10 : Il faut partir de `strlen(chaine)-1`
9. Ligne 10 : La fonction `strlen()` n'est pas déclarée
10. Ligne 12 : Il manque `return 0 ;`

Corrigé exercice 2 (9 pts. (3 pts. par question))

1. Solution :

```
#include <stdio.h>
#include <ctype.h>

char *pstr2cstr(char *cstr, char *pstr, unsigned int maxsz)
{
    unsigned int truesz = (unsigned int)pstr[0]; // Taille de la chaîne pstr
    unsigned int sz = (truesz <= maxsz-1) ? truesz : maxsz-1;

    for (unsigned int i = 0; i < sz; ++i) {
        if (isprint(pstr[i+1])) {
            cstr[i] = pstr[i+1];
        } else {
            cstr[i] = '?';
        }
    }
    cstr[sz] = '\0';
    return cstr;
}

/*
Programme principal pas demandé
*/
int main(void)
{
    char pstr[] = { 4, 't', 15, 't', 'o' }; // 4 est la taille de la chaîne; 15 le code
                                           // ASCII d'un caractère non imprimable
}
```

```

    char cstr[6];

    pstr2cstr(cstr,pstr,6);
    printf("%s\n",cstr);

    return 0;
}

```

2. Solution :

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double mysqrt(double a)
{
    if (a == 0.0) {
        return 0.0;
    }
    double a2=a*a;
    double x = (a*a2 + 3*a2)/(3*a2+a);

    for (unsigned int i = 0; i < 4; ++i) {
        double x2 = x*x, x3 = x2*x;
        double xp = (x3 + 3*a*x) / (3*x2+a);
        x = xp;
    }
    return x;
}

int main(void)
{
    double val = 0.0, maxerr = 0.0;
    FILE *f = fopen("mysqrt.dat","w");

    if (f != (FILE*)NULL) {
        fprintf(f,"v\te\n");
        while (val <= 100) { // On ne peut pas utiliser la boucle 'for' car l'incrément est réel
            double curerr = fabs(sqrt(val)-mysqrt(val));
            if (curerr > maxerr) {
                maxerr = curerr;
            }
            fprintf(f,"%g\t %g\n",val,curerr);
            val += 0.0125;
        }
        printf("Erreur maximum: %g\n",maxerr);
    } else {
        fprintf(stderr,"Erreur d'ouverture en écriture du fichier mysqrt.dat\n");
        exit(1);
    }

    return 0;
}

```

Corrigé exercice 3 (7 pts. (1 pt. par question))

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>

// Taille maximum d'une ligne correspondant à un vers
#define TAILLE_VERS_MAX 80

typedef char *poeme_t[6];

typedef struct {
    /*
    Un poème est composé de 6 vers
    On dispose d'un tableau de vers (chaînes de caractères) possibles pour chacun des 6 vers
    */
}

```

```

*/
char **versT[6]; // versT[x] pointe sur un tableau de 'nbvers' vers possible pour le vers 'x'
unsigned int nbvers; // Nombre de vers possible pour chacun des 6 vers d'un poème
} base_vers;

base_vers creer_base(FILE *f)
{
    base_vers bd;
    char buffer[TAILLE_VERS_MAX]; // Contient temporairement un vers lors de la lecture dans le fichier
    fscanf(f,"%d\n",&bd.nbvers);

    // Création des tableaux de vers disponibles pour chacun des 6 vers requis
    for (size_t v = 0; v < 6; ++v) {
        bd.versT[v] = (char**)calloc(bd.nbvers,sizeof(char*));
    }

    for (unsigned int p = 0; p < bd.nbvers; ++p) { // Lecture de chaque poème
        for (unsigned int v = 0; v < 6; ++v) { // Lecture de chaque vers du poème
            fgets(buffer,TAILLE_VERS_MAX-1,f); // Lecture de lignes avec des espaces
            // => on ne peut pas utiliser fscanf()
            bd.versT[v][p] = (char*)calloc(strlen(buffer)+1,sizeof(char));
            strcpy(bd.versT[v][p],buffer);
        }
        fgets(buffer,TAILLE_VERS_MAX-1,f); // Absorption de la ligne vide
    }
    return bd;
}

void detruire_base(base_vers bd)
{
    for (unsigned int v = 0; v < 6; ++v) {
        for (unsigned int p = 0; p < bd.nbvers; ++p) {
            free(bd.versT[v][p]);
        }
        free(bd.versT[v]);
    }
}

void creer_poeme(base_vers bd, poeme_t p)
{
    // Le poème créé n'est qu'une collection de pointeurs vers des vers dans la base
    for (unsigned int v = 0; v < 6; ++v) {
        p[v] = bd.versT[v][rand() % bd.nbvers]; // Pas de recopie de chaîne
    }
}

void afficher_poeme(FILE *f, poeme_t p)
{
    for (unsigned int v = 0; v < 6; ++v) {
        printf("%s",p[v]);
    }
}

int main(void)
{
    FILE *f = fopen("base.pom","r");

    if (f == (FILE*)NULL) {
        fprintf(stderr,"Erreur d'ouverture en lecture de base.pom\n");
        exit(1);
    }

    srand(getpid()); // Initialisation du générateur pseudo-aléatoire avec le n° de processus

    base_vers bd = creer_base(f);

    poeme_t poeme;

    creer_poeme(bd,poeme);
}

```

```
    afficher_poeme(stdout,poeme);  
    detruire_base(bd);  
    return 0;  
}
```