Révisions

Programmation en C (LC4)

31 mai 2007

1 Pointeurs

▶ **Exercice 1**

programme	a	b	c	p1, *p1	p2, *p2
`int a, b, c, *p1, *p2;`	?	?	?	-,?	-,?
`a = 1, b = 2, c = 3;`	1	2	3	-,?	-,?
`p1 = &a, p2 = &c;`	1	2	3	&a, 1	&c, 3
`*p1 = (*p2)++;`	3	2	4	&a, 3	&c, 4
`p1 = p2;`	3	2	4	&c, 4	&c, 4
`p2 = &b;`	3	2	4	&c, 4	&b, 2
`*p1 -= *p2;`	3	2	2	&c, 2	&b, 2
`++*p2;`	3	3	2	&c, 2	&b, 3
`*p1 *= *p2;`	3	3	6	&c, 6	&b, 3
`a = ++*p2 * *p1;`	24	4	6	&c, 6	&b, 4
`p1 = &a;`	24	4	6	&a, 24	&b, 4
`*p2 = *p1 /= *p2;`	6	6	6	&a, 6	&b, 6

▶ **Exercice 2**

```
void echange(int *ptr_a, int *ptr_b) {
    int tmp = *pts_a;
    *ptr_a = *ptr_b;
    *ptr_b = tmp;
}

int main(void) {
    int ec1 = 4, ec2 = 2;
    printf("%i, %i \n", ec1, ec2);
    echange(&ec1, &ec2);
    printf("%i, %i \n", ec1,  ec2);
    return 0;
}
```

▶ **Exercice 3**

```
void echange_tab(int **t, int **r) {
    int *tmp = *t;
    *t = *r;
    *r = tmp;
}

void affiche_vecteur(int *vecteur, int dimension) {
    int i;
    printf("( ");
    for(i = 0; i < dimension; i++)
        printf("%d ", vecteur[i]);
```

```
    printf(")\n");
}

int main(void) {
    int *t = malloc(3 * sizeof(int));
    int *r = malloc(3 * sizeof(int));
    t[0] = 1;
    t[1] = 6;
    t[2] = 2;
    r[0] = 5;
    r[1] = 3;
    r[2] = 9;
    echange_tab(&t, &r);
    printf("t :\n");
    affiche_vecteur(t, 3);
    printf("r :\n");
    affiche_vecteur(r, 3);
    free(r);
    free(t);
    return 0;
}
```

▶ **Exercice 4**

a) $*p + 2$: 14

b) $*(p + 2)$: 34

c) $\&p + 1$: pas grand chose (&p est l'adresse à laquelle est stockée le pointeur p)

d) $\&a[4] - 3$: adresse de a[1]

e) $a + 3$: adresse de a[3]

f) $\&a[7] - p$: 7

g) $p + (*p - 10)$: adresse de a[2]

h) $*(p + *(p + 8) - a[7])$: 23 (a[1])

2 Chaînes de caractères

▶ **Exercice 5**

```
char *recherche_char(char *s, char c) {
    for (; *s; s++) {
        if (*s == c)
            return s;
    }
    return NULL;
}
```

▶ **Exercice 6**

```
int compte_char(char *s, char c) {
    int n = 0;
    while ((s = recherche_char(s, c))) {
        n++;
        s++;
    }
    return n;
}
```

▶ **Exercice 7**

```c
#include <stdio.h>
#include <string.h>

char *strdup(const char *s) {
    char *t ;
    size_t len;

    len = strlen(s);
    t = malloc(len + 1);

    if (t == NULL)
        return NULL;

    strncpy(t, s, len + 1);

    return t;
}
```

▶ **Exercice 8**

```c
#include <string.h>

int strend(const char *s, const char *t) {
    size_t len_s, len_t;

    len_s = strlen(s);
    len_t = strlen(t);

    if (len_s < len_t)
        return 0;

    return (strcmp(s + len_s - len_t, t) == 0);
}
```

▶ **Exercice 9**

```c
#include <ctype.h>

int strcasecmp(char *s1, char *s2)
{
    unsigned char c1, c2;

    do {
       c1 = tolower((unsigned char) *s1++);
       c2 = tolower((unsigned char) *s2++);
    } while (c1 && c2 && c1 == c2)

    return (c1 - c2);
}
```

3 Listes chaînées

▶ **Exercice 10**

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

```c
struct couple_s {
    int x, y;
};

typedef struct couple_s *couple_t;

couple_t alloue_couple(int x, int y)
{
    couple_t c = malloc(sizeof(struct couple_s));
    if (c == NULL)
        return NULL;
    c->x = x;
    c->y = y;
    return c;
}

void libere_couple(couple_t c)
{
    free(c);
}

void affiche_couple(couple_t c)
{
    printf("(%d, %d)\n", c->x, c->y);
}
```

► **Exercice 11**

```c
struct liste_s {
    couple_t couple;
    struct liste_s *suivant;
};

typedef struct liste_s *liste_t;

void affiche_liste(liste_t l)
{
    while (l != NULL) {
        affiche_couple(l->couple);
        l = l->suivant;
    }
}
```

► **Exercice 12**

```c
liste_t concatene(couple_t c, liste_t l)
{
    liste_t t = malloc(sizeof(struct liste_s));
    if (t == NULL)
        return NULL;
    t->couple = c;
    t->suivant = l;
    return t;
}
```

► **Exercice 13**

```c
int compte_differents(liste_t l)
```

```
{
    int n = 0;
    while (l != NULL) {
        if (l->couple->x != l->couple->y)
            n++;
        l = l->suivant;
    }
    return n;
}
```

► **Exercice 14**

```
int dedouble(liste_t l)
{
    liste_t t;
    couple_t c;
    while (l != NULL) {
        if ((t = malloc(sizeof(struct liste_s))) == NULL)
            return -1;
        if ((c = malloc(sizeof(struct couple_s))) == NULL) {
            free(t);
            return -1;
        }
        c->x = l->couple->y;
        c->y = l->couple->x;
        t->couple = c;
        t->suivant = l->suivant;
        l->suivant = t;
        l = t->suivant;
    }
    return 0;
}
```

4 Entrées-sorties

► **Exercice 15**

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

char *lire_ligne_clavier(void) {
    int c, cpt = 0;
    char *ligne = NULL, *t;
    while((c = getchar()) != '\n') {
        t = realloc(ligne, cpt + 1);
        if (t == NULL)
            return ligne;
        ligne = t;
        ligne[cpt] = c;
        cpt++;
    }
    t = realloc(ligne, cpt + 1);
    if (t == NULL)
        return ligne;
    ligne = t;
    ligne[cpt] = '\0';
    return ligne;
```

```
}

int main(void) {
    char *s = lire_ligne_clavier();
    printf("%s\n", s);
    free(s);
    return 0;
}
```

► **Exercice 16**

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

int copie_fichier_par_bloc(const char *src, const char *dest)
{
    const size_t taille_bloc = 4096;
    int taille;
    char *bloc;
    FILE *fp_src, *fp_dest;
    if ((fp_src = fopen(src, "rb")) == NULL)
        return -1;
    if ((fp_dest = fopen(dest, "wb")) == NULL) {
        fclose(fp_src);
        return -1;
    }
    while ((taille = fread(buffer, 1, taille_bloc, fp_src)) == taille_bloc) {
        if (fwrite(buffer, 1, taille_bloc, fp_dest)) != taille_bloc)
            return -1;
        if (taille < taille_bloc)
            break;
    }
    if (ferror(fp_src)) {
        fclose(fp_dest);
        fclose(fp_src);
        return -1;
    }
    if (feof(fp_src)) {
    }
    fclose(fp_dest);
    fclose(fp_src);
    return 0;
}
```

► **Exercice 17**

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

void affiche_fiche(fiche_t f){
    printf("%s %s :\n", f->prenom, f->nom);
    printf("   age = %d an(s)\n", f->data.age);
    printf("   taille = %f m\n", f->data.taille);
    printf("   ville = %d \n", f->data.code_postal);
}

int ecrire_fiche(fiche_t f, FILE *fp) {
```

```
    int len_nom = strlen(f->nom);
    int len_prenom = strlen(f->prenom);
    if (fwrite(&len_nom, sizeof(int), 1, fp) < 1) {
        printf("probleme a l'ecriture de len_nom\n");
        return -1;
    }
    if (fwrite(f->nom, 1, len_nom+1, fp) < (len_nom + 1)) {
        printf("probleme a l'ecriture de nom\n");
        return -1;
    }
    if (fwrite(&len_prenom, sizeof(int), 1, fp) < 1) {
        printf("probleme a l'ecriture de len_prenom\n");
        return -1;
    }
    if (fwrite(f->prenom, 1, len_prenom + 1, fp) < (len_prenom + 1)) {
        printf("probleme a l'ecriture de len_prenom\n");
        return -1;
    }
    if (fwrite(&f->data, sizeof(donnees_s), 1, fp) < 1) {
        printf("probleme a l'ecriture de data %d %d\n");
        return -1;
    }
    return 1;
}

int lire_fiche(fiche_t f, FILE *fp) {
    int len_nom, len_prenom;
    len_nom = -1;
    if (fread(&len_nom, sizeof(int), 1, fp) < 1) {
        printf("probleme a la lecture de len_nom\n");
        return -1;
    }
    f->nom = malloc(len_nom + 1);
    if (fread(f->nom, 1, len_nom + 1, fp) < (len_nom + 1)) {
        printf("probleme a la lecture de nom\n");
        return -1;
    }
    if (fread(&len_prenom, sizeof(int), 1, fp) < 1){
        printf("probleme a la lecture de len_prenom\n");
        return -1;
    }
    f->prenom = malloc(len_prenom + 1);
    if (fread(f->prenom, 1, len_prenom + 1, fp) < (len_prenom + 1)) {
        printf("probleme a la lecture de prenom\n");
        return -1;
    }
    if (fread(&f->data, sizeof(donnees_s), 1, fp) < 1) {
        printf("probleme a la lecture de data\n");
        return -1;
    }
    return 1;
}

int main(void) {
    FILE *fp;
    fiche_t fiche_GL = malloc(sizeof(struct fiche_s));
```

```
    fiche_t fiche_essai = malloc(sizeof(struct fiche_s));
    donnees_s donnees_GL;
    fiche_GL->nom = "Lambert";
    fiche_GL->prenom = "Gerard";
    donnees_GL.age = 42;
    donnees_GL.code_postal = 75000;
    donnees_GL.taille = 1.80;
    fiche_GL->data = donnees_GL;

    affiche_fiche(fiche_GL);

    if ((fp = fopen("essai" , "wb")) == NULL) {
        printf("erreur lors de l'ouverture du fichier essai\n") ;
        free(fiche_essai->nom);
        free(fiche_essai->prenom);
        free(fiche_essai);
        free(fiche_GL);
        return 1;
    }

    ecrire_fiche(fiche_GL, fp);

    if (fclose(fp) == EOF) {
        printf("erreur lors de la fermeture du fichier essai\n") ;
        free(fiche_essai->nom);
        free(fiche_essai->prenom);
        free(fiche_essai);
        free(fiche_GL);
        return 1;
    }

    printf("----------------------\n");
    if ((fp = fopen("essai", "rb")) == NULL){
        printf("erreur lors de l'ouverture du fichier essai\n") ;
        free(fiche_essai->nom);
        free(fiche_essai->prenom);
        free(fiche_essai);
        free(fiche_GL);
        return 1;
    }

    lire_fiche(fiche_essai, fp);
    affiche_fiche(fiche_essai);

    fclose(fp);
    free(fiche_essai->nom);
    free(fiche_essai->prenom);
    free(fiche_essai);
    free(fiche_GL);
    return 0;
}
```