TD 10

Programmation en C (LC4)

Semaine du 2 avril 2007

1 Listes

▶ **Exercice 1**

```c
#include <stdlib.h>

void decoupe(struct liste *l, struct liste **l1, struct liste **l2){
    struct liste *t;
    *l1 = l;
    *l2 = (l != NULL) ? l->suivant : NULL;
    while (l != NULL) { /* équivalent à: while (l) { */
        t = l->suivant;
        if (t != NULL) { /* équivalent à: if (t != NULL) { */
            l->suivant = t->suivant;
            l = t->suivant;
            t->suivant = (t->suivant != NULL) ? t->suivant->suivant
                                              : NULL;
        } else
            l = NULL;
    }
}
```

▶ **Exercice 2**

```c
void filtre(struct liste *l) {
    struct liste *t;
    while (l != NULL) { /* équivalent à: while (l) { */
        t = l->suivant;
        if (t) {
            l->suivant = t->suivant;
            free(t);
        }
        l = l->suivant;
    }
}
```

2 Arbres

▶ **Exercice 3**

```c
int meme_squelette(struct arbre *a, struct arbre *b) {
    if (!a && !b) /* équivalent à: if (a == NULL && b == NULL) */
        return 1;
    else
        if (a && b) { /* équivalent à: if (a != NULL && b != NULL) */
            return meme_squelette(a->gauche, b->gauche)
```

```
                && meme_squelette(a->droite, b->droite);
        } else
            return 0;
}
```

► **Exercice 4**

```
int prefixe(struct arbre *a, struct arbre *b) {
    if (!a) /* équivalent à: if (a == NULL) */
        return 1;
    if (!b) /* équivalent à: if (b == NULL) */
        return 0;
    if (a->valeur != b->valeur)
        return 0;
    return prefixe(a->gauche, b->gauche)
        && prefixe(a->droite, b->droite);
}
```

► **Exercice 5**

```
struct liste_de_noeuds;

struct arbre {
    int valeur;
    struct liste_de_noeuds *liste_de_fils;
};

struct liste_de_noeuds {
    struct arbre *noeud;
    struct liste_de_noeuds *suivant;
};

int arite_max(struct arbre *a) {
    int arite_max_fils = 0;
    int nb_fils = 0;
    int t;
    struct liste_de_noeuds *l = a->fils;
    while (l) { /* équivalent à: while (l != NULL) */
        nb_fils++;
        t = arite_max(l->noeud);
        if (t > arite_max_fils)
            arite_max_fils = t;
        l = l->suivant;
    }
    if (nb_fils > arite_max_fils)
        return nb_fils;
    else
        return arite_max_fils;
}
```

3 Pointeurs vers des fonctions

► **Exercice 6**

```
int (*)(int)
```

► **Exercice 7**

```c
int pour_tout(struct liste *l, int (*predicat)(int)) {
    while (l)
        if ((*predicat)(l->valeur)) /* ou: if (predicat(l->valeur)) */
            l = l->suivant;
        else
            return 0;
    return 1;
}
```

► **Exercice 8**

```c
int est_pair(int n) {
    return (n % 2 == 0);
}

int tous_pairs(struct liste *l) {
    return pour_tout(l, &est_pair); /* ou: return pour_tout(l, est_pair); */
}
```

► **Exercice 9**

```c
struct liste *insere(struct liste *l, void * x, int (*inferieur)(void *, void *)) {
    struct liste **pere = &l;
    struct liste *resultat = malloc(sizeof(struct liste));
    while (l && (*inferieur)(l->valeur, x)) {
/* ou:
 *  while (l && inferieur(l->valeur, x)) {
 */
        pere = &l->suivant;
        l = l->suivant;
    }
    resultat->valeur = x;
    resultat->suivant = l;
    *pere = resultat;
    return l;
}
```