Programmation et Structures de Données 1995-96
TD 10
Les pointeurs en C

Frédéric Vivien, bureau 306, e-mail: Frederic.Vivien

18 décembre 1995

1 Liste chaînée

On définit, une fois n'est pas coutume, un type liste chaînée et un certain nombre de fonctions classiques sur ce type:

```
#define N 100

#define UNDEF_VALUE -1
#define UNDEF_LIST NULL


typedef int     item;

struct cell {
        item            value;
        struct cell    *next;
        enum {
                FREE, BUSY
        }               status;
};

typedef struct cell *list;

static struct cell tab[N];

/************************************************************/

void error(char *s)
{
        fprintf(stderr, "Error detected: %s\n", s);
        exit(-1);
}

void init()
{
        struct cell    *p;
        for (p = tab; p < tab + N; p++) {
                p->value = UNDEF_VALUE;
                p->next = UNDEF_LIST;
```



```
        }
}

void check(list l)
{
        if (l == UNDEF_LIST)
                error("undefined list");
        if ((l != tab) &&
            (l->next == UNDEF_LIST))
                error("ill-linked list");
}

list empty()
{
        return (tab);
}

int is_empty(list l)
{
        check(l);
        return (l == tab);
}

item car(list l)
{
        check(l);
        if (is_empty(l))
                error("car");
        return (l->value);
}

list cdr(list l)
{
        check(l);
        if (is_empty(l))
                error("cdr");
        return (l->next);
}

/*********************************************************/

static struct cell * allocate()
{
        struct cell    *p;
        for (p = tab + 1; p < tab + N; p++)
                if (p->status == FREE) {
                        p->status = BUSY;
                        return (p);
                }
        error("cannot allocate any new cell");
}

list
```

```
cons(item x, list l)
{
        struct cell    *p;
        check(l);
        p = allocate();
        p->value = x;
        p->next = l;
        fprintf(stderr, ".");
        return (p);
}

/********************************************************/

void print_list(list l)
{
        check(l);
        if (is_empty(l))
                return;
        fprintf(stderr, "%d ", car(l));
        print_list(cdr(l));
}

void print_newline()
{
        fprintf(stderr, "\n");
}

/*********************************************************/

void main()
{
        list            l;
        int             i;
        l = empty();

        init();

        for (i = 0; i < N - 2; i++)
                l = cons(i, l);

        print_list(l);
        print_newline();

        l = cons(N - 2, l);

        print_list(l);
        print_newline();

        l = cons(N - 1, l);

        print_list(l);
        print_newline();
}
```


Comprendre le fonctionnement de ce programme et en particulier du programme principal.

Pour les exercices suivant il est fortement recommandé de récupérer ce code, soit dans le fichier `fvivien/Enseignement/Sujets/p` soit à l'URL `http://www.ens-lyon.fr/~fvivien/Enseignement/prog_TD_10.c`.

2 Fonctions classiques...

En plus des fonctions fournies, implémenter les fonctions `rplacd` (remplace le "cdr"), `rplaca` (remplace le "car"), `reverse` et deux versions de la fonction `append`: l'une avec remplacement physique et l'autre sans.

3 Un tri

Implémenter le tri par insertion et l'exécuter dans le pire cas...

4 Liste doublement chaînée

Se servir de la version proposée des listes simplement chaînées pour implémenter des listes doublement chaînées (on placera une cellule vide à chaque extrémité de la liste). Écrire les fonctions d'insertion à droite et à gauche.