TD 5 : Chaînes de caractères

Programmation en C (LC4)

Semaine du 26 février 2007

1 Chaînes de caractères

▶ **Exercice 1**

```
#include <string.h>

char *recherche(struct traduction *dico, int n, char *s) {
    int i = 0, j = n - 1, k, cmp;
    while (i <= j) {
        k = (i + j) / 2;
        cmp = strcmp(dico[k].a, s);
        if (cmp < 0)
            i = k + 1;
        if (cmp > 0)
            j = k - 1;
        if (cmp == 0)
            return dico[k].b;
    }
    return NULL;
}
```

▶ **Exercice 2**

```
#include <string.h>

int nombre_espaces(char *s) {
    int cpt = 0;
    s = strchr(s, ' ');
    while (s != NULL) {
        s++;
        cpt++;
        s = strchr(s, ' ');
    }
    return cpt;
}
```

▶ **Exercice 3**

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

char **decoupe(char *s){
    int cpt = nombre_espaces(s);
    char *t = s;
    char **res;
    res = malloc((cpt + 2) * sizeof(char *));
    cpt = 0;
```

```
    t = s;
    s = strchr(t, '␣');
    while(s != NULL) {
        res[cpt] = malloc(s − t + 1);
        strncpy(res[cpt], t, s − t);
        res[cpt][s − t] = '\0';
        s++;
        t = s;
        s = strchr(t, '␣');
        cpt++;
    }
    res[cpt] = malloc(strlen(t) + 1);
    strcpy(res[cpt], t);
    res[cpt + 1] = NULL;
    return res;
}
```

► **Exercice 4**

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

char *reconstruit(char **tab) {
    char *res;
    int len = 0;
    int i = 0;
    while (tab[i] != NULL) {
        len += strlen(tab[i]) + 1;
        i++;
    }
    res = malloc(len);
    *res = '\0';
    i = 0;
    while (tab[i] != NULL) {
        if (i)
            strcat(res, "␣");
        strcat(res, tab[i]);
        i++;
    }
    return res;
}
```

► **Exercice 5**

```
#include <stdlib.h>

char *traducteur(struct traduction *dico, int n, char *s) {
    char *mot, *res;
    char **dec = decoupe(s);
    int i = 0;
    while (dec[i] != NULL) {
        mot = recherche(dico, n, dec[i]);
        free(dec[i]);
        dec[i] = mot;
        i++;
    }
    res = reconstruit(dec);
    free(dec);
```

```
    return res;
}
```

2 Mémoire

► **Exercice 6**

```
#include <string.h>

void init_tab(int *tab, int n, int val) {
    int len = 1;
    *tab = val;
    while (2 * len <= n) {
        memcpy(tab + len, tab, len * sizeof(int));
        len *= 2;
    }
    memcpy(tab + len, tab, (n - len) * sizeof(int));
}
```

► **Exercice 7**

- Non, on ne peut pas réécrire la fonction init_tab() en utilisant la fonction memset car la variable val qui est de type **int** s'écrit sur plusieurs octets qui ne seront pas nécéssairement à la même valeur.
- Oui, on peut utiliser memset() pour faire l'équivalent de ce que fait calloc() pour allouer un tableau d'entiers :

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

void *calloc_int(int n) {
    int *tab = malloc(n * sizeof(int));
    return memset(tab, 0, n * sizeof(int));
}
```