TD 4 : Pointeurs et structures

Semaine du 19 février 2007

1 Chaînes de caractères

▶ **Exercice 1**

```
char *recherche(char *s, char c) {
    while (*s != '\0') {
        if (*s == c)
            return s;
        s++;
    }
    return NULL;
}
```

▶ **Exercice 2**

```
int compte(char *s, char c) {
    int n = 0;
    s = recherche(s, c);
    while (s != NULL) {
        n++;
        s++;
        s = recherche(s, c);
    }
    return n;
}
```

2 Polynômes

▶ **Exercice 3**

```
struct polynome *somme_polynome(struct polynome *P, struct polynome *Q) {
    struct polynome *resultat = malloc(sizeof(struct polynome));
    int i;
    if (P->degre < Q->degre) {
        struct polynome *T = P; P = Q; Q = T;
    } /* Q->degre <= P->degre */
    resultat->degre = P->degre;
    resultat->coefficients = malloc((P->degre + 1) * sizeof(double));
    for (i = 0; i <= Q->degre; i++)
        resultat->coefficients[i] = P->coefficients[i] + Q->coefficients[i];
    for (; i <= P->degre; i++) /* i = Q->degre + 1 */
        resultat->coefficients[i] = P->coefficients[i];
    return resultat;
}

struct polynome *produit_polynome(struct polynome *P,struct polynome *Q) {
    struct polynome *resultat = malloc(sizeof(struct polynome));
```

```c
    int i, j;
    resultat->degre = P->degre + Q->degre;
    resultat->coefficients = malloc((resultat->degre + 1) * sizeof(double));
    for (i = 0; i <= resultat->degre; i++)
        resultat->coefficients[i] = 0.0;
    for (i = 0; i <= P->degre; i++)
        for (j = 0; j <= Q->degre; j++)
            resultat->coefficients[i + j] += P->coefficients[i]
                                           * Q->coefficients[j];
    return resultat;
}
```

3 Matrices

▶ **Exercice 4**

```c
struct matrice *alloue_matrice(int lignes, int colonnes) {
    struct matrice *A = malloc(sizeof(struct matrice));
    int i;
    A->lignes = lignes;
    A->colonnes = colonnes;
    A->coefficients = malloc(A->lignes * sizeof(double *));
    for (i = 0; i < A->lignes; i++)
        A->coefficients[i] = malloc(A->colonnes * sizeof(double));
    return A;
}
```

▶ **Exercice 5**

```c
struct matrice *produit_matrice(struct matrice *A, struct matrice *B) {
    struct matrice *resultat;
    int i, j, k;
    if (A->colonnes != B->lignes)
        return NULL;
    resultat = alloue_matrice(A->lignes, B->colonnes);
    for (i = 0; i < A->lignes; i++)
        for (j = 0; j < B->colonnes; j++) {
            resultat->coefficients[i][j] = 0.0;
            for (k = 0; k < A->colonnes; k++)
                resultat->coefficients[i][j] += A->coefficients[i][k]
                                              * B->coefficients[k][j];
        }
    return resultat;
}
```

▶ **Exercice 6**

```c
void libere_matrice(struct matrice *A) {
    int i;
    for (i = 0; i < A->lignes; i++)
        free(A->coefficients[i]);
    free(A->coefficients);
    free(A);
}
```

4 Pile