TP 6 : STRUCTURES ET NOUVEAUX TYPES

Table des matières

But

Vous devez maitriser à la fin de cette séance la manipulation des structures et la définition de nouveaux types.

Exercice 1 : nombres complexes

(1) Proposez un type structuré pour les nombres complexes. Celui-ci comportera deux champs de type double : une partie réelle et une partie imaginaire.

(2) Proposez ensuite des fonctions pour additionner, multiplier, élever au carré un nombre complexe et calculer le carré de sa norme. On rappelle que $(a+ib)+(a'+ib') = (a+a')+i(b+b')$, $(a+ib)\times(a'+ib') = (a\times a' - b\times b') + i(a\times b' + b\times a')$ et $|a+ib|^2 = a^2+b^2$.

(3) Écrire un `main` permettant de tester ces fonctions

Exercice 2 : alphabet morse

D'après Wikipedia, l'alphabet Morse permet de transmettre un message sous forme de signaux (électriques ou lumineux) courts ou longs, appelés « ti » (point) et « ta » (trait). Un « ta » est un signal trois fois plus long qu'un « ti » . Dans la suite, on représente un « ti » par '=' et donc un « ta » par trois occurrences successives de '='.

Un code morse est composé d'un certain nombre (au plus 4 pour les lettres) de tels signaux. Une caractéristique du morse est que la taille du code dépend de la lettre à coder : les lettres les plus fréquentes sont associées aux codes les plus courts. À l'intérieur d'un code, les « ti » et « ta » sont séparés par un « silence » d'une durée d'un « ti », qu'on représente ci-dessous par ' '.

Exemple : 'c' est représentée par le code Morse "=== = === =", et 'e' par "=". La longueur du code de 'c' est donc 11 et celle de 'e' est 1.

Deux lettres successives sont séparées par **trois** « silences », deux mots successifs sont séparés par sept « silences ».

On dispose du type C suivant :

```
typedef struct s_codage{
    int lg; /*longueur  du code: 1 a 15*/
    char signaux[15]; /* signaux a emettre: de 1 a 15 signes '=' ou ' '*/
} t_codage;
```

et d'une variable *globale* qui contient le codage en Morse des lettres minuscules :

```
t_codage morse[26]; /* tableau des codes morse des lettres de 'a' a 'z'*/
```

cette variable globale est définie dans le fichier `code_morse.h` que vous inclurez dans votre code.

(1) Écrire une fonction `int encode(char x, char *res)` qui prend en paramètre un caractère `x` supposé être une lettre minuscule, copie son code morse (suivi de `'\0'`) dans le tableau `res` (supposé de dimension suffisante) et renvoie le nombre de signaux ou -1 si `x` est incorrect.

(2) Écrire une fonction `char decode(char tab[], int lg)` qui renvoie la lettre correspondant au code morse contenu dans le tableau `tab`. On fournit aussi la longueur `lg` du code (nombre de signaux). La fonction retourne `'\0'` si le codage est incorrect.

(3) Écrire une fonction `void affiche_vers_morse(char *texte_alpha).` qui imprime sur la console la séquence morse correspondant à un texte contenant uniquement des minuscules, des espaces et le caractère de fin de chaîne `'\0'`.

(4) (facultatif). Écrire une fonction `void Affiche_vers_ASCII(char *texte_morse)` qui imprime la séquence ASCII correspondant à un texte morse terminé par le caractère `'\0'`. Le texte est supposé sans espaces en tête et en queue. Si le codage morse est incorrect (format incorrect, code inconnu) on émet un message d'erreur.

(5) Écrire un `main` permettant de tester ces fonctions.