TD6 : BPC

Exercice 17. [Comparaison de structures] Considérons la structure suivante:

```
struct date_s {
  int jour, mois, annee;
};
```

Donnez le code d'une fonction de prototype

```
int cmp_date(struct date_s, struct date_s);
```

qui compare si une date est plus récente qu'une autre. Cette fonction renverra 1 si la première date est plus récente, -1 si elle est plus ancienne, et 0 si les deux dates sont identiques. Pour éviter une répétition de code, il est suggéré d'écrire une macro adaptée.

```
#define cmp(val1, val2)\
    if (val1 > val2) return 1;
    if (val1 < val2) return -1;

int cmp_date(struct date_s a, struct date_s b){
    cmp(a.annee, b.annee);
    cmp(a.mois, b.mois);
    cmp(a.jour, b.jour);
    return 0;
}
```

1000 IQ

3

Exercice 18. [Manipulations de formules] On désire manipuler des formules caractérisées par :

- un nom de variable est un identificateur auquel est associée une valeur flottante ;
- un terme est soit une valeur flottante immédiate, soit un nom de variable ;

Une formule peut être un terme, la somme de deux termes, le produit (ou le quotient) d'un terme et d'un coefficient entier ;

- l'évaluation d'un terme est la valeur flottante *associé* au terme.
- l'évaluation d'une formule est une valeur flottante *produite* par la formule.

Ainsi, les formules suivantes sont valides :

$$a, \qquad a + 4.0, \qquad a/4, \qquad a + b, \qquad b \times 3, \quad 5.0.$$

On propose les définitions et déclarations suivantes pour les variables et les termes:

```
/* Les variables */
#define NAME_LG 16 /* longueur d'un id de variable */
#define NVARS 32 /* nombre maximal de variables */

struct var_s {
  char v_name[NAME_LG]; /* nom de la variable */
  float v_val; /* valeur de cette variable */
};

static struct var_s vars[NVARS]; /* "table des symboles" */
static unsigned int nvars=0;     /* nombre de variables d\'efinies */

/* Les termes */
enum term_type_e {VAL_T, ID_T}; /* 2 types de termes~: flottant et identificateur */

struct term_s {
  enum term_type_e t_type;
  union { /* union anonyme */
    float t_val;
    char t_var[NAME_LG];
  } t_term;
};
```

1. Proposez une fonction d'identificateur `value_of_term` qui évalue un terme (i.e. qui retourne sa valeur). Vous aurez probablement besoin de deux fonctions intermédiaires: une pour évaluer une variable, et au sein de celle-ci, une fonction pour comparer deux chaînes de caractères (pour rappel, une chaîne de caractères est un tableau de caractères qui contient le caractère `\'0'` marquant la fin de la chaîne).

2. Proposez une définition de type pour représenter une formule.

3. Proposez une fonction d'identificateur `value_of_formula` qui évalue une formule (i.e. qui retourne la valeur d'une formule passée en paramètre).

```
int cmp_str (char a[3], char b[3]){
    int i=0;
    while (a[i] != '\0' || b[i] != '\0'){
        if (a[i] != b[i]){
            return 0;
        }
        i++;
    }
    return 1;
}
```

40 -> 3,5

go