ENSEIRB-MATMECA – Filière Electronique

Examen Langage C - PG108
2020-2021 1ère session
Mercredi 20 Janvier 2021
Durée de l'épreuve : 2h / documents autorisés – sans calculatrice

1) Fonctions mystère:

Que font les fonctions suivantes ?
Pour chaque fonction, maximum : une phrase pour expliquer ce que fait la fonction, éventuellement une phrase supplémentaire par paramètre pour en expliquer l'utilité.

```
int harry(int hermionne) {
    int ron=2;
    while (ron*ron <= hermionne) {
        if (hermionne%ron == 0) {
            return 0;
        }
        ron++;
    }
    return 1;
}
```

```
int loulou(char *riri) {
    int fifi = 0;
    if (riri[0] == '-')
        fifi++;
    while (riri[fifi]) {
        if (riri[fifi]<'0')
            return 0;
        if (riri[fifi]>'9')
            return 0;
        fifi++;
    }
    return 1;
}
```

```
float f(float f1) {
    float f2 = 0.0;
    float f3 = 1000*f1;
    float f4 = f2*f3 - f1;
    while (abs(f4*f4 - f1) > 0.001) {
        f4 = (f2 + f3)/2;
        if (f4*f4 > f1)
            f3 = f4;
        else
            f2 = f4;
    }
    return f4;
}
```

```
void sid(char *manny) {
    char *diego = manny;
    while (*manny) {
        *diego = *manny;
        if (*manny != ' ')
            diego++;
        manny++;
    }
}
```

2) Exercice:

Écrivez une fonction qui convertit une chaine de caractère codant une valeur hexadécimale en sa valeur équivalente au format long. Son prototype sera le suivant :

```
long hextoi(char *string);
```

3) Le *jeu* des 3 erreurs:

La fonction suivante compte le nombre de mots dans une chaine de caractères (le nombre de groupes de caractères séparés par un ou plusieurs espaces). Trois erreurs de syntaxe se sont glissées dedans.
- Mentionnez chacune de ces trois erreurs
- Re-écrivez la fonction en les corrigeant

```
int countwords(char *string) {
    int mots = (*string != ' ');
    while (*string) {
        if (*string == ' ')
            while (*string = ' ');
                string++;
            if (*string)
                mots++;
        string++;
    }
    return mots;
}
```

Précisions :
- l'algorithme est correct, seule la syntaxe est en cause
- l'indentation est bonne également (pas de piège)
- le programme compile sans problème mais fournit un mauvais résultat. Si on compile avec l'option `-Wall`, deux *Warning*s et une *Note* sont générés.

4) Problème:

Les questions du problème peuvent être traitées de manière indépendante, mais il vous sera plus facile de le mener à terme en suivant les questions dans l'ordre. Prenez le temps de lire le sujet jusqu'au bout avant de commencer.

Nous allons nous intéresser à la programmation d'une calculatrice. Le principe : le programme attend que l'utilisateur entre un calcul, une fois le calcul reçu sous forme de chaîne de caractère, le résultat est calculé puis affiché dans le terminal. Si l'utilisateur appuie sur 'entrée' sans demander de calcul, le programme s'arrête. Le programme devra accepter les opérations d'addition, de soustraction, de multiplication et de division. Il devra également respecter les priorités et accepter les parenthèses.

Nous aurons d'abord besoin d'écrire quelques sous-fonctions avant de nous intéresser au programme principal. Lors de la saisie, le programme récupère une chaine de caractère représentant tout ce qu'a entrée l'utilisateur, y compris le 'entrée' final (représenté comme un saut de ligne). Nous aurons donc besoin d'une fonction qui renvoie le nombre de caractères AVANT le saut de ligne. Cette fonction s'appellera *`longueur_ligne`* et renverra un entier.

a) Donnez le prototype de la fonction *`longueur_ligne`*.

b) Ecrivez la fonction *`longueur_ligne`* complète.

Etant donné que le programme doit accepter les parenthèses, il faut vérifier qu'il n'y a pas d'erreur dans leur organisation. Ce sera le rôle de la fonction *err_parenth* qui acceptera une chaine de caractères en argument, et qui renverra une valeur entière nulle si tout va bien, non nulle s'il y a un problème de parenthèses.

La fonction *err_parenth* utilise une variable locale. En parcourant la chaine de caractère, cette variable est incrémentée pour chaque caractère correspondant à une parenthèse ouvrante, et décrémentée pour chaque caractère correspondant à une parenthèse fermante. La fonction se termine au premier caractère de saut de ligne (ou à la fin de la chaine s'il n'y a pas de saut de ligne). Si tout va bien, la variable locale est à 0 une fois la chaine parcourue (autant de parenthèses ouvrantes que fermantes). La variable locale doit toujours être positive (on ouvre les parenthèses avant de les fermer), donc si elle est négativre, il faut retourner immédiatement qu'il y a eu un problème.

c) Donnez le code C de la fonction *err_parenth*. (Un point bonus si vous le faites avec une seule variable).

Pour calculer la formule, nous utiliserons la fonction `calcule_rec`, qui est récursive. Cette fonction reçoit une chaine de caractères et une longueur en entrée, et renvoie un flottant. La longueur indiquée sera potentiellement plus petite que le nombre de caractères dans la chaine. Cela signifie simplement qu'on ne s'intéresse pas aux caractères au-delà du nombre indiqué.

d) Donnez le prototype de `calcule_rec`.

e) Si on part d'une chaine de caractères définie par

```
char exemple[20] = "39+5*7-42";
```

- comment appeler la fonction `calcule_rec` pour n'obtenir que le résultat correspondant aux caractères 0 à 5 ?
- comment appeler la fonction `calcule_rec` pour n'obtenir que le résultat correspondant aux caractères 3 à 5 ?

f) En supposant que la fonction `calcule_rec` soit écrite, écrivez la fonction `main` qui utilise les trois fonctions précédentes pour fournir le comportement de calculatrice attendu. L'algorithme suivant est donné à titre indicatif :

```
déclaration des variables (un buffer de taille 100 et un entier)
boucle infinie
      demander de rentrer une formule
      récupérer l'entrée de l'utilisateur dans un buffer
      stocker la longueur de la formule dans l'entier
      si la longueur est nulle, on quitte tout
      s'il y a une erreur de parenthèses
            on affiche une erreur
      sinon
            on affiche le résultat calculé
```

Il ne reste maintenant qu'à écrire la fonction `calcule_rec`. Dont l'algorithme grossier est donné ci-après :

- 1 : traiter la première opération + ou – qui ne soit pas dans des parenthèses

- si une opération est rencontrée, on utilise `calcule_rec` sur les parties à droite et à gauche, on additionne/soustrait les valeurs ainsi obtenues et on renvoie le résultat.
- 2 : Traiter la première opération * ou / qui ne soit pas dans des parenthèses (cf + et -)
- 3 : Si le premier caractère est une parenthèse,
    - calculer le contenu avec calcul_rec et renvoyer le résultat
- 4 : Convertir la chaine de caractère en flottant.

Nous traiterons les différentes parties en commençant par la fin.

g) En respectant le prototype que vous avez donné à la question d), écrivez le code C correspondant à la partie (4).
ATTENTION : pour rappel, tout le programme travaille directement sur la copie originale de la chaine rentrée par l'utilisateur. Il est donc fort probable que la chaine reçue ne se termine pas immédiatement après la valeur à convertir.

h) Proposez le code C correspondant à la partie (3) de l'algorithme. Le plus simple est d'utiliser `calcule_rec` sur les caractères situés entre les parenthèses.

i) Pour coder la partie (2), inspirez-vous de la fonction *`err_parenth`*. En effet, il faut rechercher le premier symbole '*' ou '/' qui ne soit pas entre des parenthèses (compteur de parenthèses à 0). Si un tel caractère est trouvé, il faut utiliser `calcule_rec` deux fois : sur les caractère avant le symbole et sur les caractères après le symbole. Il ne vous reste qu'a effectuer l'opération idoine.

j) pour les calculs d'addition et de soustraction, le code est très proche de la question précédente, au détail qu'une valeur numérique peut commencer par + ou – sans que cela ne pose de problème au sens mathématique. Proposez une adaptation pour traiter cette exception.

Le programme est maintenant complet, mais il reste assez basique. En effet, il y a assez peu de vérification de syntaxe, et la fonction `calcule_rec` ne dispose d'aucun moyen de faire remonter une erreur (syntaxe ou division par zero).

k) proposez une solution technique pour pouvoir indiquer si le résultat renvoyé par la fonction `calcule_rec` est digne de confiance.