TP3 – IGI 2101 – 2h

Manipulation des tableaux, chaines de caractères

Exercice 1 : précision des calculs flottants

Tester le programme suivant vu en cours :

```
#include<stdio.h>

int main( ) {
  float a=0.1;
  float b=0.2;
  float res;
  res=a+b;
  printf("%f\n", a);
  printf("%f\n", b);
  printf("%f\n", res);

  if (res==0.3)
    printf("oui, a+b=0.3 \n");
  else
    printf("non, a+b est different de 0.3 !!! \n");
  return 0;
}
```

Comme expliqué en cours, ce n'est pas un bug mais l'impossibilité pour un processeur de stocker exactement certaines valeurs comme 0.1 et 0.2, elles sont approximées : pour visualiser les valeurs utilisées par votre processeur ajouter les lignes suivantes :

```
printf("%.17f\n", a);
printf("%.17f\n", b);
printf("%.17f\n", res);
```

Exercice 1 : tableau

Ecrire un programme qui alloue le tableau suivant **int Notes[]={8,11,14,7,17};**

1. Quelle quantité de mémoire (en octets) est nécessaire pour stocker ce tableau dans la mémoire ? Vérifiez avec l'opérateur sizeof.

2. On peut calculez **le nombre d'éléments** de ce tableau à l'aide du sizeof précédent et de la taille de chaque élément que l'on obtient aussi avec sizeof. Rappel : sizeof d'un tableau ne donnera pas la taille du tableau quand le tableau sera passé en argument d'une fonction, nous reverrons cela.
3. A l'aide d'une *boucle for* utilisant le calcul précédent, affichez chaque élément de ce tableau un à un ainsi que son adresse.
4. A l'aide d'une *boucle for* calculez et affichez la moyenne de ce tableau. Affichez le résultat en décimal (demandez à l'intervenant si vous n'arrivez pas à afficher la bonne valeur…il faut que votre accumulateur ne soit pas un entier).

Exercice 2 : printf %s et %c

Ecrire un programme qui crée et initialise une chaîne de caractères :

char chaine1[50] ="ABCDEFGHIJKLMNOPQRSTUVWXYZ";

1. Affichez cette chaine avec **printf**(‘’ %s’’, chaine).
2. Affichez
    a. le nombre d'octets occupés par le tableau chaine1 en utilisant sizeof,
    b. le nombre de caractères « utiles » (c'est-à-dire réellement utilisés sur les 50 réservés) stockés dans chaine1 avec la fonction **strlen** (voir poly de cours puisque pas encore vu en amphi).
3. Une autre façon d'afficher la chaine de caractères est d'utiliser une **boucle for** et un printf **%c** pour afficher les caractères de cette chaîne **un à un** (en utilisant donc strlen pour connaitre le nombre d'itérations à effectuer).
4. Une autre façon d'affichez un à un chaque caractère est d'utiliser une **boucle while** et le fait qu'une chaine de caractères se termine toujours par le caractère '\0'. Afficher aussi le code ASCII de chaque caractère.

Exercice 3 : concaténation de chaînes de caractères

Déclarez 3 tableaux :

- *char tab3[30];*
- *char tab1[]="ceci";*
- *char tab2[]=" est une phrase";*

1. Affichez les 3 chaines avec des printf %s. Pourquoi l'affichage de tab3 peux-il poser problème à ce stade ?

Nous souhaitons que tab3 contiennent la concaténation (mise bout à bout) de tab1 et tab2.

Pour cela la première idée est d'essayer de faire tab3 = tab1 + tab2; mais nous avons vu en cours que cela n'est pas possible en C, pourquoi ? Essayez !

Pour le faire sans utiliser les fonctions de manipulation de chaines de caractères (qui seront utilisées dans l'exercice suivant), il faut :

2. Sans utiliser strlen, écrire une boucle while qui copie un à un les caractères de tab1 dans tab3.
3. Puis une seconde boucle qui copie les caractères de tab2 dans tab3 à la suite des caractères provenant de tab1. Ne pas oublier de copier le caractère \0 à la fin.
4. Afficher tab3 avec %s (si des caractères bizarres sont affichés c'est que votre programme contient un bug…)

NB : chaque boucle s'arrête quand elle tombe sur le caractère de fin de chaîne '\0'

Exercice 4 : concaténation de chaînes de caractères

Refaire l'exercice 3 en utilisant cette fois les fonctions

- **strcpy** pour copier tab1 dans tab3 (voir poly de cours)
- puis la fonction **strcat** qui concatène 2 chaines ensemble : strcat(tab3, tab1) ==> ajoute tab1 **à la fin** de tab3.

NB : ne pas oublier #include <string.h> en plus de #include<stdio.h>.