TP1 C – ZZ1

Première partie : manipulation de chaînes et des fonctions de `string.h`

Déclarer une chaîne de caractères initialisée à `"Bonjour les ZZ1".`

1. L'afficher et calculer sa taille.

2. Mettre un `'\0'` en 8ème position, l'afficher et calculer sa taille.
   Remettre un espace à la 8ème position et vérifier que l'on est bien revenu au cas de départ en affichant à nouveau la taille et la chaîne de caractères.

3. Un affichage en `%d` sur un caractère permet d'afficher le code ASCII de celui-ci. Donner les codes de `'A'` et de `'a'`. Enfin, donner les codes ASCII en décimal et en hexadécimal de tous les caractères de la chaîne.

4. Comparer des chaînes de caractères : différence sur la taille, sur la casse d'un caractère. Vérifier que les informations données par `man` sont cohérentes.

   Copier une chaine de caractères dans une autre en utilisant la fonction `strcpy`. Faites un test pour constater qu'il n'y a pas de vérification sur la taille de la chaine de caractères (copier une grande chaîne dans une autre trop petite). Les fonctions `scanf` et `gets` souffrent du même type de problème, si l'utilisateur saisit une chaîne qui est plus grande que la variable où on souhaite la stocker.
   Chercher ensuite une fonction qui permette de copier et de contrôler le nombre de caractères ?

Deuxième partie : codage de l'examen

Voici un bref rappel des exercices de C proposés dans l'examen. Nous allons profiter du compilateur pour vérifier vos exercices…

1. Remplir un tableau de `200` éléments de nombres entre `0` et `100`. Transformer en son opposé tout nombre plus petit qu'une valeur saisie par l'utilisateur.

2. Compter le nombre d'occurrences des lettres de l'alphabet d'un mot donné. Appliquer cela aux mots lus à partir d'un fichier texte. On pourra se limiter aux majuscules puis il faudra faire les minuscules et les majuscules.

3. Faire le codage César (décalage de 13 caractères ;-)) d'abord sur les minuscules puis sur les majuscules et les minuscules. La traduction sera faite tant que l'utilisateur n'aura pas tapé une ligne contenant seulement un point.

Troisième partie : codage des fonctions usuelles

Lors des semaines bloquées, nous avons réalisé en TD les fonctions `strlen, strcpy, strcmp`.
Coder ces fonctions en changeant leur nom si vous ne l'avez pas déjà fait. Nous verrons plus tard comment optimiser le code.

Vérifier que les comportements sont bien identiques. Pour la fonction `strcmp` à recoder, la fonction retourne `0` pour des chaînes égales, une valeur négative si la première chaîne est inférieure à la seconde et une valeur positive sinon.

Quatrième partie : illustration de quelques concepts

Les derniers transparents du cours des semaines bloquées montrent quelques particularités du langage C : le masquage possible des variables et le passage des paramètres par valeur.

Coder et mettre en situation ces concepts sur un petit programme.

Le programme rédigé doit être fonctionnel, bien indenté et commenté. Il doit respecter le guide de style fourni en cours. Toutes les fonctions que vous écrivez doivent être testées et validées.

Pour la compilation avec gcc, vous utiliserez les options suivantes :

```
-Wall –ansi -pedantic
```

Elles permettent de s'assurer que votre code supporte le standard ANSI et ne comporte pas d'avertissement.