Université de Provence
Université Paul Cézanne
LICENCE INFORMATIQUE
Stage C
2008-2009

version du *17 septembre 2008*

Boucles et Conditions

A Exercices

A.1 Ecrire un programme `bissextile` qui détermine si une année (passée en ligne de commande) est bissextile dans le calendrier grégorien.

A.2 Ecrire un programme `racines2` qui calcule les racines (réelles ou imaginaires) d'un polynôme réel de degré deux (les coefficients sont passés en ligne de commande). (indication : utiliser la fonction `double sqrt (double x)` de `math.h`)

A.3 Ecrire un programme `codeascii` qui affiche tous les entiers de 0 à 255 suivis du caractère correspondant dans le code ASCII.

A.4 Ecrire un programme `plusoumoins` qui choisit un entier entre 1 et 1024 au hasard. Le programme demande ensuite à l'utilisateur de lui donner un entier jusqu'à ce que l'utilisateur découvre l'entier choisit. A chaque fois le programme indique si la solution proposée est trop grande ou trop petite.

Modifier le précédent programme pour n'autoriser que 9 tentatives.

A.5 (facultatif) Ecrire un programme `eratosthene` qui prend en paramètre un entier n inférieur à 50 et renvoie la liste des nombres premiers compris entre 2 et n en employant la méthode du crible d'Eratosthène.

Cette méthode consiste à considérer un tableau formé des n premiers entiers supérieurs à 1. On commence par afficher le premier nombre de ce tableau et ensuite on "raye" de ce tableau tous les multiples de ce nombre. On recherche ensuite l'entier suivant qui n'est pas rayé, et on recommence.

B Anagrammes

B.6 Ecrire un programme `testverif` qui prend en paramètre un mot et affiche s'il appartient au dictionnaire ou non. Une bibliothèque est fournie pour vérifier qu'un mot est valide. Voir `verif.h` et `verif.o`.

B.7 Ecrire une fonction `int longueur(char * chaine)` qui renvoie la longueur de la chaîne de caractères `chaine`. On supposera que cette chaîne respecte la convention usuelle de terminaison par '\0'.

Cette fonction est à mettre dans la bibliothèque `libchaine`.

B.8 Ecrire une fonction `void transposition(char * mot)` qui échange les deux premières lettres de `mot`.

Ecrire une fonction `void circulaire(char * mot)` qui effectue une permutation circulaire des lettres de `mot`.

Ecrire une fonction `int fact(int n)` qui renvoie la factorielle de `n`.

Toutes ces fonctions sont à rassembler dans la bibliothèque `libperm`.

B.9 Ecrire un programme `anagramme` qui prend en paramètre un mot et effectue une alternance de `transposition` et permutation `circulaire` sur celui-ci, en testant à chaque fois si le mot obtenu est un mot du dictionnaire ou non. Chaque fois qu'un mot valide est trouvé, celui-ci est affiché, il s'agit d'une anagramme du mot donné.

Au bout de combien de fois est-on certain de retrouver des mots déjà testés ? A-t-on réellement testé *toutes* les permutations possibles ? Proposer une "démonstration expérimentale".