Université Blaise Pascal

Licence 1ère année
Année universitaire 2009-2010

Algorithmique et programmation : langage `C`

Examen terminal (première session)

Durée : 2h

Documents interdits, traducteurs autorisés, calculatrices interdites.

Le sujet comporte trois pages. Tout programme (ou partie de programme) qui ne sera pas clair sera considéré comme faux. En cas de difficultés à écrire une fonction, il est conseillé de se limiter à l'écriture de son entête avec ses paramètres afin de pouvoir l'utiliser ultérieurement.
Les exercices sont indépendants. N'hésitez pas à mettre de côté un exercice ou une question qui vous paraît trop difficile pour y revenir plus tard.

Exercice 1 : Fonctions

Partie 1.1 : Drapeau danois

Le 26 mai célèbre l'anniversaire du prince Frederik de Danemark. Dans cette partie, nous allons faire un programme pour dessiner le drapeau danois.

Question 1. Écrivez un programme complet en langage C permettant de lire un entier n au clavier.

Question 2. Écrivez la partie d'un programme permettant d'afficher sur une même ligne : $n+1$ points, une étoile, puis $2n+2$ points. Pour $n=2$, la partie du programme affiche :

```
...*......
```

Question 3. Écrivez la partie d'un programme permettant d'afficher un drapeau danois dans un rectangle de $3n+4$ de large, et de $2n+1$ de haut. Un exemple est présenté sur la figure suivante, pour $n=2$.

```
...*......
...*......
**********
...*......
...*......
```

Partie 1.2 : Équations du second degré

Un polynôme du second degré à coefficients réels du type ax^2+bx+c peut avoir zéro, une ou deux racines réelles, selon le signe de son discriminant Δ.

Question 4. Considérez le prototype suivant :

```
int polyDegreDeux(float a, float b, float c, float * x1, float * x2);
```

Le retour de la fonction indique le nombre de racines. Que représentent `x1` et `x2` ? Pourquoi utilise-t'on `float *` pour `x1` et `x2` ?

Question 5. Écrivez le corps de la fonction `main` qui réalise les opérations suivantes :

1. demande à l'utilisateur de saisir `a`, `b` et `c`,
2. appelle la fonction `polyDegreDeux`,
3. affiche le nombre de racines et les éventuelles racines du polynôme $ax^2 + bx + c$.

Il n'est pas demandé d'écrire le corps de la fonction `polyDegreDeux`.

Exercice 2 : Approximation du nombre d'or

Le nombre d'or, aussi appelé la divine proportion, est égal à $\varphi = (1+\sqrt{5})/2$. Ce nombre peut être approché en utilisant la suite (φ_n) définie par : $\varphi_0 = 1$ et $\varphi_{n+1} = 1 + 1/\varphi_n$ pour tout $n \geq 0$.

Question 6. Écrivez un algorithme qui demande la saisie d'un entier p compris entre 5 et 20, quitte à demander plusieurs fois.

Question 7. Écrivez un algorithme qui donne une approximation du nombre d'or en calculant φ_p (p étant l'entier précédent).

Question 8. On peut remarquer (et on admettra) que $(\varphi_{i+1} - \varphi_i)^2$ décroît à mesure que i augmente. On considère que pour une certaine valeur $\varepsilon > 0$ donnée, on obtient une approximation satisfaisante à partir du moment où $(\varphi_{i+1} - \varphi_i)^2 < \varepsilon^2$. Écrivez un algorithme qui donne une approximation satisfaisante du nombre d'or, ε étant donné en paramètre.

Exercice 3 : Le jeu du pendu

Toutes les fonctions de cet exercice doivent être réalisées en langage C.

Dans cet exercice, nous allons réaliser un jeu de pendu. L'utilisateur proposera un mot que l'ordinateur va tenter de deviner. Pour deviner le mot, l'ordinateur propose des lettres une par une. Initialement, toutes les lettres du mot apparaissent cachées. À chaque fois que l'ordinateur propose une lettre qui se trouve dans le mot, toutes les occurrences de cette lettre sont révélées. Quand l'ordinateur propose une lettre qui ne se trouve pas dans le mot ou qui a déjà été proposée, l'ordinateur perd un essai. Au bout de huit essais perdus, l'ordinateur perd la partie. Si l'ordinateur trouve toutes les lettres (sans perdre huit essais), l'ordinateur gagne la partie.

La figure 1 présente un exemple de partie.

```
Proposez un mot : arbre
Mot : -----, 8 essais
Lettre choisie : e
Mot : ----e, 8 essais
Lettre choisie : z
Mot : ----e, 7 essais
Lettre choisie : a
Mot : a---e, 7 essais
Lettre choisie : r
Mot : ar-re, 7 essais
Lettre choisie : b
Bravo, le mot était arbre.
```

Figure 1 – Un exemple de partie de pendu.

Question 9. La fonction `lireCaractere` ci-dessous possède des erreurs. Corrigez-les.

```
char lireCaractere(char c) {
scanf("%c", &c);
}
```

Question 10. Écrivez la fonction `char lireLettre()` qui lit une lettre minuscule, quitte à redemander plusieurs fois si nécessaire. Les lettres minuscules sont les caractères compris entre `'a'` et `'z'`.

Question 11. Adaptez la fonction `lireLettre` pour qu'elle puisse lire une lettre minuscule ou le caractère retour chariot (correspondant à la touche entrée).

Question 12. Écrivez la fonction `int lireMot(char mot[30])` qui lit un mot composé uniquement de lettres minuscules, terminé par le caractère retour chariot, et le stocke dans le tableau `mot`. La fonction retourne le nombre de lettres lues. On notera qu'aucun mot en français ne dépasse 30 lettres.

Question 13. Écrivez la fonction `void afficherMot(char mot[30], int n, int devinees[30])` qui affiche le mot à deviner, n étant la longueur du mot. Les lettres non encore devinées sont représentées par le caractère `'-'`. Vous utiliserez le tableau `devinees` défini de la manière suivante : `devinees[i]` vaut 1 si la $(i+1)$-ème lettre a été devinée, et 0 sinon.

Question 14. Écrivez la fonction `gagne` qui, à partir d'un entier n et d'un tableau de n entiers, retourne 1 si tous les entiers sont égaux à 1, et retourne 0 sinon.

Question 15. Écrivez la fonction `proposerLettre1` qui fait en sorte que l'ordinateur propose une des lettres minuscules aléatoirement.

Question 16. La fonction `proposerLettre1` écrite au dessus pose problème : en l'utilisant, il est possible que l'ordinateur propose deux fois la même lettre au cours d'une partie, ce qui n'est pas souhaitable. Pour éviter cela, l'ordinateur peut utiliser un tableau des lettres qu'il a déjà proposées. Nous allons réaliser une fonction `proposerLettre2` qui fait en sorte que l'ordinateur propose une lettre aléatoirement, mais ne propose pas l'une des n lettres déjà proposées.

1. Écrivez la fonction `estDejaProp(char c, char dejaProp[34], int t);` qui retourne 1 si `c` est présente dans l'une des t lettres du tableau `dejaProp`, et 0 sinon. t représente le numéro du tour (à ne pas confondre avec n, la longueur du mot). On notera qu'il n'est pas possible de jouer plus de 34 fois (26 lettres et 8 erreurs).
2. Écrivez la fonction `char proposerLettre2(char dejaProp[34], int t);` qui fait en sorte que l'ordinateur propose aléatoirement une lettre minuscule qui n'a pas encore été proposée, puis retourne cette lettre.

Question 17. Écrivez la fonction `verifierLettre` qui : (1) ajoute la lettre proposée dans le tableau `dejaProp`, (2) modifie le tableau `devinees` pour toutes les occurrences de la lettre, si la lettre est présente dans le mot à deviner, (3) retourne le nombre d'essais restants.

Question 18. Écrivez la fonction `jouerTour` qui joue un tour. Les actions à réaliser dans le tour sont : (1) la proposition d'une lettre par l'ordinateur, (2) la vérification de la lettre, (3) l'affichage du mot et du nombre d'essais restants.

Question 19. Écrivez le programme principal qui réalise les opérations suivantes : (1) déclare et initialise (si nécessaire) les trois tableaux `mot`, `devinees` et `dejaProp`, (2) demande à l'utilisateur de taper son mot, (3) fait jouer l'ordinateur, (4) affiche le gagnant de la partie.