



Examen d'Algorithmique et programmation C++ (DVE11)

Session Principale

Durée 1h30

UGEM

Exercice 1 [4 pts] :

On va Ecrire une fonction qui prend en paramètre un entier et retourne combien de fois il est divisible par 2, Par exemple : $24/2=12$ et $12/2 = 6$ et $6/2 = 3$ et 3 n'est pas divisible par 2

⇒ 24 est divisible par 2 trois fois.

1. Donnez son prototype.
2. Ecrire le code de la fonction.
3. Écrire également son programme de test (main).

Exercice 2 [3 pts] :

La suite de Fibonacci est définie comme suit :

- $F_0=F_1=1$
- $F_n=F_{n-2}+F_{n-1}$ pour $n>2$

1. Écrire une fonction récursive qui prend comme paramètre un entier n , et qui calcule et retourne F_n .
2. Tester la fonction avec une valeur aléatoire de l'intervalle $[0,100]$

Exercice 3 [4 pts] :

1. Écrire une fonction récursive qui prend comme paramètre une chaîne de caractères, puis elle remplace les caractères espaces par `_`. Par exemple, si la chaîne est "Bonne chance au devoir", La fonction devrait renvoyer la chaîne "Bonne_chance_au_devoir"

Voici le prototype de la fonction :

`string fonction (string phrase, int i)`

2. Tester votre programme avec une phrase saisie par l'utilisateur

Exercice 4 [3 pts] :

Une matrice symétrique est une matrice carrée qui est égale à sa propre transposée, c'est-à-dire telle que $a_{i,j} = a_{j,i}$ pour tous i et j compris entre 0 et $n-1$.

1. Ecrire une fonction `Est_Symétrique()` qui prend comme paramètres une matrice carrée M d'ordre n et affiche si M est symétrique ou non.
2. Tester la fonction avec la matrice :

	1	4	6
M =	4	5	9
	6	9	2

Exercice 5 [6 pts] :

1. Ecrire une fonction `Initialiser()` permettant de remplir un tableau T et sa taille n avec des valeurs réels saisie par l'utilisateur.
2. Ecrire une fonction `PremOccurrence ()` qui prend comme paramètre un tableau T et sa taille n et un réel x puis elle retourne la position de la première occurrence de x dans T , sinon elle retourne -1
3. Ecrire la fonction `DemOccurr ()` qui retourne la position de la dernière occurrence de x dans T .
4. En utilisant les deux fonctions précédentes écrire la fonction `UneOccurrence()` permettant de tester si le nombre d'occurrence x dans T est égal à 1.
5. Tester les fonctions.