

Travaux dirigés n° 3

Les fonctions

Exercice 1 (Nombres premiers)

On suppose que l'on dispose d'une fonction `estPremier(n : entier) : booléen` qui teste si l'entier n est premier.

1°) Écrire une fonction qui compte le nombre de nombres premiers compris dans un intervalle d'entiers positifs fixé à $[a, b[$. Par exemple : `comptePremiers(5, 15) == 4`.

2°) Écrire une fonction qui, pour un entier n donné, calcule le nombre premier de rang n . Par exemple : `premier(1) == 2` car 2 est le premier nombre premier et `premier(5) == 11` car 11 est le cinquième.

3°) Écrire une fonction qui détermine le rang d'un nombre premier.

Exercice 2 (Compter, accumuler des opérations)

On désire écrire un algorithme permettant de saisir une liste de notes et d'afficher leur moyenne.

1°) On suppose que le nombre de notes est connu à l'avance (et saisi en premier). Écrire un algorithme permettant ce traitement.

2°) L'utilisateur n'est pas satisfait d'avoir à entrer lui-même le nombre de notes ! Il faut donc adopter une nouvelle convention permettant de gérer le cas où le nombre de notes n'est pas connu à l'avance : la saisie de la série de notes sera interrompue quand l'utilisateur entrera une valeur particulière : -1 par exemple (qui ne peut pas être confondu avec une note !). Modifier votre algorithme pour qu'il utilise cette convention.

3°) Ajouter à votre algorithme des fonctionnalités permettant, après la saisie, de déterminer :

- le nombre de notes ;
- le nombre de notes au moins égales à 10 ;
- le minimum de la série de notes ;
- le maximum de la série de notes ;
- la moyenne des notes.

Exercice 3 (Les classiques)

1°) On se donne une suite numérique $(u_i)_{i \in \mathbb{N}}$.

Pour $n \in \mathbb{N}$ on pose :

$$S_n = \sum_{i=0}^n u_i \quad P_n = \prod_{i=0}^n u_i \quad M_n = \max\{u_i \mid 0 \leq i \leq n\} \quad m_n = \min\{u_i \mid 0 \leq i \leq n\}$$

Donner les fonctions permettant de calculer les termes des suites $(S_n)_{n \in \mathbb{N}}$, $(P_n)_{n \in \mathbb{N}}$, $(M_n)_{n \in \mathbb{N}}$ et $(m_n)_{n \in \mathbb{N}}$. On suppose disposer de la fonction `u(i : entier) : entier` qui calcule le terme u_i .

2°) Soit $(n, p) \in \mathbb{N} \times \mathbb{N}$ avec $0 \leq p \leq n$, donner la fonction pour calculer le coefficient binomial C_n^p .

Rappel : $C_n^p = \frac{n!}{p!(n-p)!} = \frac{n(n-1)\dots(n-p+1)}{p!}$

3°) Soit $(x, n) \in \mathbb{R} \times \mathbb{N}$, donner une fonction pour calculer $S = \sum_{k=0}^n \frac{x^k}{k!}$

Exercice 4 (Recherche séquentielle)

1°) Écrire une fonction qui détermine le plus petit diviseur ($\neq 1$) d'un entier au moins égal à 2.

2°) En déduire une fonction qui teste si un entier est premier.

Exercice 5 (Nombre à deviner, recherche par dichotomie)

On vous propose ici un jeu où le joueur doit deviner, en un minimum d'essais, un nombre entier choisi "par l'ordinateur" entre 0 et 1023 (pour généraliser, entre 0 et $2^n - 1$).

1°) Élaborer une stratégie et déterminer le nombre maximum d'essais nécessaires.

2°) Écrire un algorithme pour une partie sans limiter le nombre d'essais : pour chaque tentative, l'ordinateur analyse la proposition du joueur et répond par "gagné" (éventuellement "gagné en ... coups"), "trop petit" ou "trop grand". On suppose l'existence de la fonction `aleatoire(min, max)` : `entier` qui retourne un entier aléatoire dans l'intervalle $[min, max]$.

3°) Modifier l'algorithme précédent afin de limiter le nombre d'essais au maximum nécessaire. En cas de dépassement, l'ordinateur répond : "Tu aurais du gagner en ... coups maximum : trop tard!".

4°) Afin d'obliger le joueur à avoir une stratégie intelligente, on propose ici que l'ordinateur lui indique s'il joue mal en lui répondant : "trop petit, je te l'ai déjà dit !" ou "trop grand, je te l'ai déjà dit !". Adapter l'algorithme à cette nouvelle contrainte.

Exercice 6 (Date du lendemain)

On désire écrire un algorithme permettant, après saisie de la date du jour, d'afficher la date du lendemain :

- les années sont codées sur 4 chiffres ;
- un message d'erreur doit être affiché si la valeur saisie pour le mois n'est pas dans $[1; 12]$;
- un message d'erreur doit être affiché si la valeur saisie pour le jour n'est pas conforme (en tenant compte du mois et de l'année).

Rappel : Une année bissextile est une année comportant 366 jours (au lieu de 365). Une année est bissextile lorsqu'elle est soit divisible par 4, mais non divisible par 100, soit divisible par 400.

1°) Écrivez une fonction permettant de déterminer si une année est bissextile.

2°) Écrivez une fonction donnant le nombre de jours d'un mois d'une année déterminée.

3°) Écrivez une fonction permettant de déterminer si une date est valide.

4°) Écrivez une fonction permettant de comparer deux dates. Elle retournera vrai si la deuxième date est inférieure ou égale à la seconde.

5°) Écrivez, finalement, un algorithme qui, après saisie d'une date et vérification de sa conformité, permet d'obtenir la date du lendemain.