Exercice 2.22 (N^e chiffre)

Le but est d'obtenir les chiffres de la séquence A033307 (ou A007376, à un décalage d'index près). L'index n fourni correspond exactement au nombre de chiffres à ignorer dans la séquence.

La solution la plus directe pour cet exercice est d'énumérer tous les entiers naturels en partant de 1, tout en sachant en permanence de combien de chiffres sont constitués les entiers en question.

Tant que le nombre de chiffres à ignorer est plus grand que le nombre de chiffres de i, on passe à i+1, en ayant ignoré autant de chiffres supplémentaires qu'il y a de chiffres dans i.

S'il est strictement plus petit, on veut un chiffre de i, donc on décompose i avec des divisions entières par 10 et un modulo pour obtenir le chiffre souhaité (unités s'il faut encore ignorer `nb_digits-1` chiffres, dizaines si `nb_digits-2`, etc.

Par exemple :

```c
int nth(int n) {
    int i = 1;              // Un compteur pour énumérer les entiers
    int nb_digits = 1;      // Le nombre de chiffres de i
    int next_increase = 10; // Le prochain i pour lequel le nombre de
                            // chiffres augmente

    while (true) {
        // Invariant : n est le nombre de chiffres restant à ignorer
        if (n<nb_digits) {
            // On veut un des chiffres de i
            for (int j=0; j<nb_digits-1-n; ++j) {
                i = i/10;
            }
            return i%10;
        } else {
            // Il faut aller chercher plus loin
            n = n-nb_digits;
            i = i+1;
            if (i == next_increase) {
                nb_digits = nb_digits+1;
                next_increase = 10*next_increase;
            }
        }
    }
}
```

On effectue de l'ordre de n opérations. Mais on peut faire mieux!

On sait que de 1 à 9, il y a 9×1 chiffres. De 10 à 99, il y a 90×2 chiffres. De 100 à 999, il y a 900×3 chiffres. Cela permet de progresser dans les i bien plus vite!

Tant que l'on peut éliminer un bloc (1 à 9, 10 à 99, etc.) d'un seul coup, on le fait. Par exemple, si $n = 3000$, on élimine les trois premiers blocs, qui totalisent 2889 chiffres. On cherche donc un chiffre d'un nombre entre 1000 et 9999.

Dans ce bloc, on cherche le chiffre d'index $3000 - 2889 = 111$. Chaque nombre de ce bloc contient quatre chiffres, donc on peut éliminer les $\lfloor 111/4 \rfloor = 27$ premiers nombres du bloc (1000 à 1026), qui totalisent 108 chiffres. On cherche donc le chiffre d'index $111 - 108 = 3$ dans l'entier 1027, soit 7.

Par exemple :

```c
int nth(int n) {
    int i = 1;              // Un compteur pour énumérer les entiers
    int nb_digits = 1;      // Le nombre de chiffres de i
    int pow = 1;            // 10^nb_digits

    // On cherche le bon groupe
    while (9*pow*nb_digits <= n) {
        n = n-9*pow*nb_digits;
        i = i+9*pow;
        pow = 10*pow;
        nb_digits = nb_digits+1;
    }

    // i est la puissance de 10 inférieure ou égale au i recherché
    // n le nombre de chiffres restant à ignorer
    i = i+n/nb_digits;
    n = n%nb_digits;

    // i est l'entier contenant le chiffre recherché
    // n le nombre de chiffres restant à ignorer (n < nb_digits)
    for (int j=0; j<nb_digits-1-n; ++j) {
        i = i/10;
    }
    return i%10;
}
```

On effectue ainsi de l'ordre de $\log_{10}(n)$ opérations, et on ne peut guère espérer mieux.