

Documents non autorisés !

Exercice 1 : Soit la fonction suivante

```
double calculer(int n, int m){
    if(n==0) return 0;
    int mm = (m < 0 ? -m : m);
    double res = 1.0;
    int i;
    for(i=1; i<=mm; i++){
        res = res*n;
    }
    if(mm != m) return 1.0/res;
    return res;
}
```

- Que retourne cet algorithme dans chacun des appels suivants :
calculer (10000, 0) ; calculer (2, -1), - calculer (3, 2), - calculer (2, 3) ?
- Combien de fois l'instruction **res = res * n** est exécutée ?
- Que retourne cette fonction dans le cas général ?

Exercice 2 : Qu'affiche le code suivant

```
int main() {
    struct l{
        int n;
        int m;
    };
    int * pn = new int;
    struct l * pl = new struct l;
    int n = 5;
    *pn = n*2;
    pl->n = *pn;
    cout<<n<<endl;
    cout<<+*pn<<endl;
    cout<<+pl->n<<endl;
    pn = &n;
    pl->m = *pn;
    cout<<n<<endl;
    cout<<+*pn<<endl;
    cout<<pl->m<<endl;
}
```

Exercice 3 : Ecrire un algorithme récursif qui prend comme paramètre un nombre entier positif n et vérifie si il existe un k tel que $n = 2^k$ (on ne demande pas de trouver le k).

Exercice 4 : pour manipuler des grand nombre naturel on propose de représenter chaque nombre sous forme d'une liste de caractères. Un élément de la liste contient l'un des symboles 0 à 9. Par exemple le nombre 1978 est représenté avec une liste de 4 éléments dont le premier élément est 8, le deuxième est 7, le troisième est 9 et le dernier est 1.

- Ecrire cette représentation en C++. Vous pouvez faire une représentation qui simplifie les algorithmes qui suivent.

2. Ecrire un algorithme qui prend 2 nombres $N1$ et $N2$ et retourne 0 si $N1=N2$, 1 si $N1 > N2$ et -1 si $N1 < N2$
3. Ecrire un algorithme qui prend 2 entiers $N1$ et $N2$ et retourne $N3 = N1+N2$
4. Ecrire un algorithme qui prend un nombre N et retourne $N/10$
5. Ecrire un algorithme qui prend 2 entiers $N1$ et $N2$ et retourne $N3 = |N1-N2|$
6. Ecrire un algorithme qui prend 2 entiers $N1$ et $N2$ et retourne $N3 = N1*N2$

Exercice 5 : pour exécuter des processus, les systèmes d'exploitation utilisent des algorithmes d'ordonnancement. **L'un de ces algorithmes fonctionne comme suit :**

- les processus sont rangés dans une liste selon leur ordre d'arrivée
- les processus sont exécutés dans l'ordre pendant une durée fixe (quantum)
- le processus qui vient d'être exécuté et n'est pas terminé est mis à la fin de la liste pour continuer le reste de son exécution plus tard

Par exemple supposons que nous avons les processus $P1$, $P2$, $P3$ dans cet ordre et que leur temps d'exécution sont respectivement : 10s, 5 sec, 7 sec et que le quantum $\Delta = 6$ s.

L'exécution sera donc : $P1$ (pour 6sec), $P2$ (5sec), $P3$ (6 sec), $P1$ (4sec) et $P3$ (1 sec).

1. Proposer une structure de données pour manipuler ses processus et les exécuter selon cet algorithme
2. Ecrire l'algorithme qui permet de prendre une liste de processus et de sortir leur ordre d'exécution avec le temps d'exécution (comme dans l'exemple)