```
double filtreUnEchantillon(double ek){
    double vktemp; // calcul intermediaire pour vk
    int i;  //indice de lecture pour les valeurs des coefficients du filtre
    int j; //indice de lecture pour les valeurs des coefficients du filtre
    int indice_lec; //indice de lecture pour les échantillons d'entrée
    double skout; //valeur pour la sortie calculée
    //suppression offset numérique
    vktemp=ek-OFFSET_ENTREE;
    //pour un RIF, NB_COEFF_A=1 donc vktemp sera inchangé
    indice_lec = indice_ecr;
    //  le premier element utile du tableau denCoeffs est a1
    for (j=1;j<NB_COEFF_A;j++) {  //M itérations
        //ici le premier vk pris en compte est vk-1, donc on décremente indice_lec avant utilisation
        indice_lec=indice_lec-1;
        if (indice_lec < 0)
            indice_lec = MEMORYSIZE - 1;
        vktemp=vktemp-a[j] * memoireVk[indice_lec];
    }
    //rangement de la valeur calculée dans tableau vk à l'indice indice_ecr
    memoireVk[indice_ecr]=vktemp;
    //calcul de sk
    skout = 0; //valeur par défaut pour le résultat
    indice_lec = indice_ecr;
    for (i = 0; i <NB_COEFF_B; i++){ //N+1 itérations
        skout = skout + b[i] * memoireVk[indice_lec];
        //ici le premier vk pris en compte est vk, donc on décremente indice_lec avant après utilisation
        indice_lec=indice_lec-1;
        if (indice_lec < 0)
            indice_lec = MEMORYSIZE - 1;
    }
    //incrémentation du pointeur d'écriture pour la prochaine itération
    indice_ecr = (indice_ecr + 1);
    if (indice_ecr>=MEMORYSIZE)
        indice_ecr=0;
    if (skout >  VALEUR_MAX )    // gestion des saturations du résultat
        skout = VALEUR_MAX ;
    else if (skout < VALEUR_MIN )
        skout = VALEUR_MIN ;
    //ajout offset numérique
    return  skout + OFFSET_SORTIE;
```