

# **Rapport du projet :**

***Conversion d'un nombre à virgule fixe et virgule flottante .***

***- Codage IEEE 754***

***(Simple précision 32 bits – 64 bits)***

Réalisé par : **Bilal ZINEDDINE** et **Yasmine HAMDANI**

Encadre par: Mr. **BENALLA Hicham**

## **Introduction :**

Ce rapport aborde le sujet de la conversion de nombres réels en représentations en virgule fixe et en virgule flottante. Ces concepts sont fondamentaux en informatique et en électronique, car ils permettent de traiter et de représenter précisément des nombres réels dans les systèmes numériques. La conversion en virgule fixe et en virgule flottante joue un rôle crucial dans les applications allant du traitement de signal aux opérations financières, en passant par les calculs scientifiques (les puissances).

## **Analyse du Problème :**

Le problème central est de développer une méthode fiable pour convertir les nombres réels dans ces deux formats différents. Chaque format a ses propres avantages et inconvénients en termes de précision, de gamme de représentation et de complexité de calcul. La virgule fixe, bien que simple dans sa représentation, est limitée en termes de gamme et de précision. D'autre part, la virgule flottante, utilisée dans les normes telles que IEEE 754, offre une plus grande flexibilité et précision, mais est plus complexe à implémenter et peut introduire des erreurs d'arrondi. Le défi est de concevoir un système capable de gérer ces conversions avec précision et efficacité.

## **Solutions Proposées :**

une solution proposée pour aborder ce problème. Il offre des fonctions dédiées pour :

- convertir les nombres décimaux en représentations binaires en virgule fixe.
- Convertir ces représentations en virgule fixe en nombres décimaux.
- Convertir les nombres décimaux en représentations binaires en virgule flottante, en tenant compte des formats à 32 et 64 bits.

## Plan d'algorithme :

### Entrée et Menu Interactif :

L'utilisateur est d'abord invité à entrer un nombre réel. Ensuite, un menu interactif propose différentes options de conversion.

### Gestion de la Virgule Fixe :


- Conversion du nombre décimal en binaire avec virgule fixe, en séparant la partie entière et la partie fractionnaire.
- Conversion inverse d'une représentation binaire en virgule fixe en nombre décimal.

### Gestion de la Virgule Flottante :

- Conversion de nombres décimaux en représentation en virgule flottante selon la norme IEEE 754, avec des options pour 32 et 64 bits. Cette conversion implique la détermination de la mantisse, de l'exposant et du bit de signe.
- Des fonctions supplémentaires pour calculer le nombre de chiffres dans les représentations binaires et décimales, et pour manipuler des tableaux de bits.

## Le programme traduit en langage C :


1. Fonction pour afficher un nombre en binaire :

A diagram of a code editor window with a light gray border. Inside, there is a white area containing C code. To the left of the code, there is a vertical gray bar with a small white rectangle at the top, representing a scrollbar or a line indicator.

```
void affiche(int* tab, int taille) {  
    for(int i=0; i<taille; i++)  
        printf("%d", tab[i]);  
}
```

La fonction `affiche(int* tab, int taille)`, affiche un nombre binaire sous forme d'un tableau.

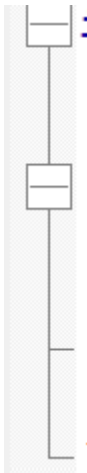
2. Fonction pour calculer le nombre de chiffre d'un nombre :



```
int nbr_digits_dec(int x) {  
    int digit=0;  
    int A=x;  
    while (A!=0) {  
        A/=2;  
        digit++;  
    }  
    return digit;  
}
```

La fonction *nbr\_digits\_dec(int x)* détermine combien de chiffres sont nécessaires pour représenter un nombre entier en format binaire. Elle répète une succession de division par 2.

3. Fonction qui calcule nombre de bits d'un nombre binaire :



```
int nbr_digits_bin(double y) {  
    int digit = 0;  
    double A = y;  
    while (A >= 1) {  
        A /= 10;  
        digit++;  
    }  
    return digit;  
}
```

La fonction *nbr\_digits\_bin(double y)* Calcule le nombre de chiffres qu'un nombre binaire aurait s'il était écrit en décimal. Elle divise le nombre binaire par 10 jusqu'à ce qu'il soit inférieur à 1.

4. Fonction pour convertir un nombre binaire en décimal :

```
int* tbBin(double bin) {
    int taille = nbr_digits_bin(bin);
    int* binaryArray = (int*)malloc(taille * sizeof(int));
    int B = (int)bin;
    int index = 0;
    while(B != 0) {
        binaryArray[taille - index - 1] = B % 10;
        B /= 10;
        index++;
    }
    return binaryArray;
}
```

La fonction *tbBin(double bin)* convertit un nombre binaire en un tableau de chiffres. La taille du tableau est définie par la fonction *nbr\_digits\_bin*, et les chiffres du nombre binaire sont insérés dans le tableau.

5. Fonction de conversion de conversion du binaire en decimale :

```
41 int binEnDec(int* presBin, int taille) {
42     int resultat = 0;
43     for (int i = 0; i < taille; i++) {
44         resultat += presBin[i] * (int)pow(2, taille - 1 - i);
45     }
46     return resultat;
47 }
```

La fonction *binEnDec(int\* presBin, int taille)* transforme un tableau représentant un nombre binaire en sa valeur décimale. Elle multiplie chaque bit par 2 à les puissances de son poids correspondant et les additionne ensemble.

6. Fonction de la conversion du deciamle en binaire :

```
int* decimalToBinary(int decimalNumber) {  
    int quotient = decimalNumber;  
    int taille=nbr_digits_dec(decimalNumber);  
    int* binaryArray=(int*)malloc(taille*sizeof(int));  
    int index = 0;  
  
    while (quotient != 0) {  
        binaryArray[taille-index-1] = quotient % 2;  
        quotient /= 2;  
        index++;  
    }  
    return binaryArray;  
}
```

La fonction *int\* decimalToBinary (int decimalNumber)* convertit un nombre décimal en un tableau représentant sa forme binaire. Elle calcule d'abord la taille nécessaire pour le tableau en utilisant *nbr\_digits\_dec*. Puis, elle remplit le tableau en calculant le reste de la division du nombre décimal par 2 (qui donne le chiffre binaire), en le plaçant dans le tableau à partir de la fin. Ce processus se poursuit jusqu'à ce que le nombre décimal soit entièrement converti en binaire.

7. Fonction conversion d'un nombre reel en binaire :

```
int* fracEnBin(float x, int ver, int* taillefr, int p) {  
    int i = 0;  
    int *F = (int)malloc((p - ver) * sizeof(int));  
  
    while (i < (p-ver)) {  
        x = x * 2;  
        if (x >= 1)  
        { F[i]=1;  
          x -= 1;  
        }  
        else{F[i]=0;}  
        i++;  
    }  
  
    *taillefr =p-ver;  
  
    return F;  
}
```

La fonction *fracEnBin(float x, int ver, int\* taillefr, int p)* convertit la partie fractionnaire d'un nombre à virgule flottante en binaire. Elle multiplie la fraction par 2 et enregistre le résultat comme 1 si le produit est supérieur ou égal à 1, sinon comme 0. Ce processus se répète pour un nombre de fois déterminé par la différence entre la précision totale et le nombre de chiffres après la virgule. La taille du tableau binaire résultant est définie par cette différence et le tableau est retourné.

8. Fonction conversion de decimal en representaion en virgule fixe :

```
void decimalEnVFixe() {
    float num, F;
    int E, p;
    int* E1;
    int* F1;
    int tailleF1;

    printf("Entrer votre nombre reel:\n");
    scanf("%f", &num);

    // Extraction de la partie entière avec floor()
    E = (int)floor(num);
    // Extraction de la partie fractionnaire
    F = num - E;

    int tailleE1 = nbr_digits_dec(E);
    E1 = (int*)malloc(tailleE1 * sizeof(int));

    printf("Entrer une precision pour la partie fractionnaire\n");
    scanf("%d", &p);

    // Conversion de la partie entière en binaire
    E1 = decimalToBinary(E);

    // Allocation de mémoire pour la partie fractionnaire en binaire
    F1 = (int*)malloc(p * sizeof(int));
    // Conversion de la partie fractionnaire en binaire
    F1 = fracEnBin(F, tailleF1, &tailleF1, p);

    printf("La representation en virgule fixe de votre nombre est : \n");
    affiche(E1, tailleE1);
    printf(".");
    affiche(F1, tailleF1);
    printf("\n");
}
```



La fonction *decimalEnVFixe()*, convertit un nombre réel en sa représentation en virgule fixe. L'utilisateur entre d'abord le nombre réel et la précision souhaitée pour la partie fractionnaire. La fonction sépare le nombre en parties entière et fractionnaire. La partie entière est convertie en binaire en utilisant *decimalToBinary*. Pour la partie fractionnaire, elle utilise *fracEnBin* qui convertit la fraction en un tableau binaire selon la précision spécifiée. Finalement, elle affiche les deux parties en format binaire avec un point séparateur pour représenter la virgule fixe.

#### 9. Fonction de conversion de représentation virgule fixe en décimale :

```
void vFixeEnDecimal() {
    double E, F;
    printf("Entrer la partie entier de ton representation:\n");
    scanf("%lf", &E);
    int tailleE1=nbr_digits_bin(E);
    //on le convertit en un tableau
    int*E1=tbBin(E);
    E=binEnDec(E1, tailleE1);
    printf("Entrer la partie fractionnaire de ton representation\n");
    scanf("%lf", &F);
    int tailleF1=nbr_digits_bin(F);
    int*F1=tbBin(F);
    float F2=0;
    float resultat;
    for(int i=0; i<tailleF1; i++){
        F2 += (F1[i] * pow(2, -(i+1)));
    }
    resultat=E+F2;
    printf("la conversion en decimale est : %f\n", resultat);
}
```

La fonction *vFixeEnDecimal()*, convertit une représentation en virgule fixe en un nombre décimal. L'utilisateur entre séparément les parties entière et fractionnaire en format binaire. Chaque partie est convertie en tableau binaire, puis en décimal. La partie fractionnaire est traitée en ajoutant chaque bit multiplié par 2 à la puissance négative de sa position. Le résultat

final est la somme des deux conversions, affichant ainsi la représentation décimale du nombre en virgule fixe.

#### 10. Fonction de conversion du decimal en virgule flottante :

```
void decimalEnVFlot(int choix) {
    int i, tailleExp, taille, taillefr, signe = 0, E, p;
    int *Ex, *bin_fr;
    float num, fr;
    int *Xbin;

    printf("Entrez votre nombre reel: ");
    scanf("%f", &num);

    // Extraction de la partie entière avec floor()
    int x = (int) floor(num);
    taille = nbr_digits_dec(abs(x));
    if (x < 0)
        signe = 1;

    // Extraction de la partie fractionnaire
    fr = num - x;
    printf("donner la precision:\n");
    scanf("%d", &p);
```

```

Xbin = decimalToBinary(abs(x));
fr /= pow(10, nbr_digits_dec(fr));
if(choix==1){
    bin_fr = fracEnBin(fr,taille-1, &taillefr, 23);
    E = (taille - 1) + 127;
    tailleExp=nbr_digits_dec(E);
    Ex = decimalToBinary(E);

    printf("%d ", signe);

    for (i = 0; i < 8; i++) {
        if (i < tailleExp)
            printf("%d", Ex[i]);
        else
            printf("%d", 0);
    }

    printf(" ");

    for (i = 1; i < taille; i++)
        printf("%d", Xbin[i]);

    for (int k = 0; k < taillefr; k++)
        printf("%d", bin_fr[k]);
}
if(choix==2){
    bin_fr = fracEnBin(fr,taille-1, &taillefr, 52);
    E = (taille - 1) + 1023;
    tailleExp=nbr_digits_dec(E);
    Ex = decimalToBinary(E);

    printf("%d ", signe);

```

```

for (i = 0; i < 11; i++) {
    if (i < tailleExp)
        printf("%d", Ex[i]);
    else
        printf("%d", 0);
}

printf(" ");

for (i = 1; i < taille; i++)
    printf("%d", Xbin[i]);

for (int k = 0; k < taillefr; k++)
    printf("%d", bin_fr[k]);
}

```

La fonction *decimalEnVFlot(int choix)*, convertit un nombre réel en sa représentation en virgule flottante selon le standard IEEE 754, en simple32bits (23 bits de mantisse) ou double précision64bits (52 bits de mantisse). L'utilisateur saisit un nombre réel et spécifie la précision. La fonction décompose le nombre en parties entière et fractionnaire, détermine le signe, et calcule la mantisse et l'exposant. Pour l'exposant, elle ajoute un biais de 127(32bits) ou 1023(64bits). Elle affiche ensuite la représentation binaire du nombre en virgule flottante, incluant le bit de signe, l'exposant et la mantisse.

#### 11. Fonction menu :

```

void menu() {
    // Affichage du menu
    int choix;
    printf("Veuillez entrer votre choix:\n");
    printf("1.Conversion du Decimal en Virgule Fixe\n");
    printf("2.Conversion du representation Virgule Fixe en Decimal\n");
    printf("3.conversion du Decimal en Virgule Flottante\n");

    // Lecture du choix de l'utilisateur
    scanf("%d", &choix);

    // Appel de la fonction appropriée en fonction du choix de l'utilisateur
    switch (choix) {
        case 1:
            decimalEnVFixe();
            break;
        case 2:
            vFixeEnDecimal();
            break;
        case 3:
            // Demander à l'utilisateur de choisir entre 32 et 64 bits
            printf("Choisissez la taille de virgule flottante (1 pour 32 bits, 2 pour 64 bits): ");
            int choixFlot;
            scanf("%d", &choixFlot);
            decimalEnVFlot(choixFlot);
            break;
        default:
            printf("Option invalide.\n");
    }
}

```

La fonction *menu()*, est un menu interactif permettant à l'utilisateur de choisir parmi trois options de conversion numérique : conversion d'un nombre décimal en virgule fixe, conversion d'une représentation en virgule fixe en décimal, et conversion d'un nombre décimal en virgule flottante. L'utilisateur fait son choix en entrant un numéro correspondant à l'option désirée. Selon le choix, la fonction appropriée est appelée pour effectuer la conversion spécifiée. Pour la conversion en virgule flottante, l'utilisateur est invité à choisir entre une représentation de 32 bits ou de 64 bits.

## 12. Fonction main :

```

238 int main() {
239     menu();
240 }
241

```

Appelle de la fonction *menu()*.

Voici un exemple d'exécution :

```
222                                     break;  
223                                case 2:  
  
C:\Users\user\Desktop\AOprojets\projet3\IEEE\main.exe  
Veuillez entrer votre choix:  
1.Conversion du Decimal en Virgule Fixe  
2.Conversion du representation Virgule Fixe en Decimal  
3.conversion du Decimal en Virgule Flottante  
1  
Entrer votre nombre reel:  
-15.35  
Entrer une precision pour la partie fractionnaire  
4  
La representation en virgule fixe de votre nombre est :  
-10000.1010  
  
Process returned 0 (0x0)   execution time : 56.326 s  
Press any key to continue.
```