Université Hassan Premier
Faculté des Sciences et Techniques
Settat

Projets du module Architecture des ordinateurs

LST : Génie Informatique

Réalisé par :

- MOUADILI Abdelmounim
- MEZOUAHI Kaoutar

Encadré par :

- Mr BENALLA HICHAM

Année universitaire : 2023-2024

Sommaire

Introduction générale

Dans ce projet on s'intéresse à créer des programmes qui permet de faire des convertissions des nombres en changeant leur base et des représentations des entiers signés en binaire par des méthodes différentes.

L'environnement :

Langage de programmation :

Langage C :

C'est un langage de programmation impératif, généraliste et de bas niveau. Inventé au début des années 1970 pour réécrire Unix, C'est devenu un des langages les plus utilisés, encore de nos jours.

Logiciel utilisé :

DevC++ :

C'est un environnement de développement intégré (IDE) permettant de programmer en C et en C++ pour les systèmes d'exploitation Windows.

Les livrables attendus :

Création des programmes en langage C qui prend des entrées (nombre entier, nombre en base quelconque …) et donne le résultat souhaiter par l'utilisateur (le nombre dans une autre base, une représentation en virgule flottante « 32 bits », ...) sinon l'informe que le programme ne peut pas faire le traitement à cause du mal utilisation d'utilisateur (enter un nombre qui n'existe pas sur la base, …)

Sujet 1 : Conversion des nombres

1. *ANALYSE :*

Objectifs :

Convertir un nombre d'une base A vers une base B (A et B compris entre 2 et 16).

Analyse du problème :

Pour passer d'une base p à une autre q la méthode générale et la plus utile c'est d'utilise la base intermédiaire 10.

C'est-à-dire supposons qu'on a un nombre N en base p et on souhaite de le convertir vers une autre quelconque disons q on doit d'abord le représenter en base 10 puis en le calcul en base souhaité q.

La première étape est la représentation en base 10 est se fait en suivant la méthode suivante :

1. Chaque digit du nombre N correspond à une puissance de p.
2. Le premier digit est le digit 0, il se trouve à droite et contient le bit appelé le bit de poids faible.
   Et le dernier digit se trouve à gauche et contient le bit de poids fort.
3. On fait la multiplication de chaque digit par le p à la puissance du poids correspond avec le poids le plus faible est 0 pas de 1.
4. Et la dernière étape est de sommer tous ces opérations de multiplication.

L'expression de N de la base p en base 10 sera donc :

N = $a_n \times p^n + a_{n-1} \times p^{n-1} + \cdots + a_0 \times p^0$ avec $0 < a_i < B$ le digit et n le poids le plus fort

Un exemple pour comprendre :

$(56)_8 = 5 \times 8^1 + 6 \times 8^0 = (46)_{10}$

La deuxième étape est la représentation de la base intermédiaire 10 en base q est se fait en suivant la méthode suivante :

1. Division de N par q donne une valeur v_0 avec un reste r_0
2. On divise v_0 par q : donne v_1 et le reste r_1
3. On recommence pour v_1 et ainsi de suite jusqu'à ce qu'on trouve une valeur v < q on arrête la division
4. Le résultat du$(N)_{10}$ en base q sera $(N)_q = r_i r_{i-1} \dots r_1 r_0$ avec r_i est le dernier reste trouvé

Un exemple pour comprendre :

$(46)_{10}$ En base 2

46 /2=23 avec un reste égal à 0

23 /2=11 avec un reste égal à 1

11 /2=5 avec un reste égal à 1

5 /2=2 avec un reste égal à 1

2/2=1 avec un reste égal à 0

1 /2=0 avec un reste égal à 1

Et donc :

$(46)_{10}$ =$(101110)_2$

2. ALGORITHME :

// Fonction main

//Déclaration des variables

nbr : tableau [] de caractères

A,B : entier

Valeurdicimale : entier

q : entier

//lecture des données

q←1

E :

ECRIRE ("Entrez le nombre : ")

LIRE (nbr)

ECRIRE ("Entrez la base du nombre (entre 2 et 16) : ")

LIRE (A)

ECRIRE ("Entrez la base de conversion (entre 2 et 16) : ")

LIRE (B)

//traitement

SI ((A < 2 **OU** A > 16) **OU** (B < 2 **OU** B > 16)) **ALORS**

ECRIRE ("Les bases doivent être comprises entre 2 et 16.") ;

SINON

Valeurdicimale ← convert10 (num, A)

ECRIRE ("Le nombre converti en base : "+B)

conversionB (decimalValue, baseB)

FINMAIN

// Fonction pour convertir un chiffre en caractère hexadécimal

FONCTION int_to_hexa(nu:entier) : char

DEBUT

SI (nu>= 0 ET nu <= 9)

  ALORS RETOURNER ('0' + nu)

SINON

  RETOURNER ('A' + nu - 10)

FINSI

FINFONCTION

// Fonction pour convertir un caractère hexadécimal en chiffre

FONCTION hexa_to_int (c: char) :entier

DEBUT

SI (c >= '0' **ET** c <= '9')

  ALORS RETOURNER (c - '0')

SINON

  RETOURNER (c - 'A' + 10)

FINSI

FINFONCTION

// Fonction pour convertir un nombre de base A en base 10

FONCTION convert10 (nu:^char, A: entier) : entier

DEBUT

DECLARATION

  resultat : entier

  len : entier

TRAITEMENT

  resultat ←0

  len ← **Longueur**(nu)

  POUR i **DE** 0 **A** len

    FAIRE resultat ←resultat * A + hexa_to_int(nu[i])

RETOURNER resultat

FINPOUR

FINFONCTION

// Fonction pour convertir un nombre de base 10 en base B

FONTION conversionB (nu : entier, B : entier)

DEBUT

DECLARATION

resultat←tableau[50] de caractere

i : entier

j: entier

reste : entier

TRAITEMENT

i← 0

TANTQUE(nu > 0)

FAIRE reste = nu % B

resultat[i++] ← int_to_hexa (reste)

num ←num / B

FINTANTQUE

// Afficher le résultat à l'envers

POUR j DE i – 1 A j >= 0 PAR PAS DE -1

ECRIRE (resultat[j])

ECRIRE ("\n")

FINPOUR

FINFONCTION

// Fonction pour verifier si le nombre entre est exacte

FONTION check(num : char, base : entier) : entier

check(^num : char ,base : entier) : entier

DEBUT

DECLARATION

temp : entier

temp<-(entier)num

v←1,c : entier

TRAITEMENT

TANTQUE(temp>0)

C←temp%10

SI(c>=base)

V←0

PAUSE

FINSI

Temp←temp/10

FINTANTQUE

RETOURNER v

FINFONCTION

3. PROGRM C

```c
#include <stdio.h>
#include <stdio.h>
#include <string.h>
// Fonction pour verifier si le nombre entre est exacte
int check(char *num,int base){
        int temp=atoi(num);
        int v=1;
        while (temp>0){
                int c=temp%10;
                if (c>=base){
                        v=0;
                        break;
                }
                temp=temp/10;
        }
        return v;
}
// Fonction pour convertir un chiffre en caractère hexadécimal
char int_to_charbit(int nu) {
   if (nu >= 0 && nu <= 9) {
      return (char)('0' + nu); // ici on a met '0' pour ajouter a num le code ASCII
                                                        //et le transformer en caractère

   } else {
      return (char)('A' + nu - 10); // meme pour qu'ici ;avec en hexadecimal
                                    //les chiffres devient caractères donc on ajoute le num
au code ASCII en commencent de A
   }
}
// Fonction pour convertir un caractère hexadécimal en chiffre
int charbit_to_int(char c) {
   if (c >= '0' && c <= '9') {
      return (int)(c - '0');
   } else {
      return (int)(c - 'A' + 10);
   }
}
// Fonction pour convertir un nombre de base A en base 10
int convert10(char *num, int A) {
   int resultat = 0;
   int i;
   int len = strlen(num);

   for ( i = 0; i < len; i++) {
      resultat = resultat * A + charbit_to_int(num[i]);
```

```c
    }

    return resultat;
}
// Fonction pour convertir un nombre de base 10 en base B
void conversionB(int num, int B) {
    char resultat[50]; // Tableau pour stocker le résultat
    int i = 0;
            int j;
    while (num > 0) {
        int reste = num % B;
        resultat[i++] = int_to_charbit(reste);
        num = num / B;
    }

    // Afficher le résultat à l'envers
    for ( j = i - 1; j >= 0; j--) {
        printf("%c", resultat[j]);
    }
    printf("\n");
}
int main() {
    char nbr[100];
    int A, B;
    char q;
    int v;

    E: printf("Entrez un nombre : ");
    scanf("%s", nbr);

    printf("Entrez la base du nombre (entre 2 et 16) : ");
    scanf("%d", &A);

    printf("Entrez la base de conversion (entre 2 et 16) : ");
    scanf("%d", &B);


    if ((A < 2 || A > 16) || (B < 2 || B > 16)) {
        printf("Les bases doivent être comprises entre 2 et 16.\n");
    } else {
            //verification de saisie
            v=check(nbr,A);
            if(!v){
            printf("saisie incorrecte!!! le nobmre doit etre dans la base %d\n",A);
                    goto E;
            }
        int valeurdecimal = convert10(nbr, A);
```

```
93      printf("Le nombre converti en base %d : ", B);
94      conversionB(valeurdecimal, B);
95    }
96
97
98          printf("\n\t pour QUITTER tappez Q ! \n");
99          printf(" \t pour CONTINUER tappez une touche differente de Q ! \n");
100         scanf("%s",&q);
101         if (q=='Q'||q=='q') {printf("\t FIN PROGRAMME \n");
102         return 0;
103                 }else
104         goto E;
105
106   return 0;
107 }
108
109
110
```

4. Captures d'écran de l'exécution (les tests)

1.

```
Entrez un nombre : 01011001111000100110101011111001101111
Entrez la base du nombre (entre 2 et 16) : 2
Entrez la base de conversion (entre 2 et 16) : 16
Le nombre converti en base 16 : 3C4D5E6F

 tappez un chiffre non nul pour continuer
        tappez 0 pour arretez
1
Entrez un nombre : 12
Entrez la base du nombre (entre 2 et 16) : 3
Entrez la base de conversion (entre 2 et 16) : 8
Le nombre converti en base 8 : 5

 tappez un chiffre non nul pour continuer
        tappez 0 pour arretez
```

2.

```
Entrez un nombre : 86
Entrez la base du nombre (entre 2 et 16) : 9
Entrez la base de conversion (entre 2 et 16) : 2
Le nombre converti en base 2 : 1001110

 tappez un chiffre non nul pour continuer
        tappez 0 pour arretez
```

3.

```
1001111
Entrez un nombre : 1001111
Entrez la base du nombre (entre 2 et 16) : 2
Entrez la base de conversion (entre 2 et 16) : 13
Le nombre converti en base 13 : 61

 tappez un chiffre non nul pour continuer
         tappez 0 pour arretez
```

4.

```
Entrez un nombre : 99
Entrez la base du nombre (entre 2 et 16) : 6
Entrez la base de conversion (entre 2 et 16) : 11
Le nombre converti en base 11 : 58

 tappez un chiffre non nul pour continuer
         tappez 0 pour arretez
```

5.

```
Entrez un nombre : 182
Entrez la base du nombre (entre 2 et 16) : 10
Entrez la base de conversion (entre 2 et 16) : 4
Le nombre converti en base 4 : 2312

 tappez un chiffre non nul pour continuer
         tappez 0 pour arretez
```

Sujet 2 : Représentation des entiers signés

1. *ANALYSE :*

Objectifs :

Un programme permettant de représenter les entiers signés en utilisant les 3 méthodes : VS, Cà1, Cà2.

Analyse de problème :

La première étape pour la représentation des entiers signés est de vérifier si le nombre entré est représentable sur 8 bits

La représentation sur 8 bits est pour les nombres dans l'intervalle de $[-2^7; 2^8]$ et savoir le type de la représentation du nombre souhaité

Au cas où le nombre entrer est positive la représentation VS = la représentation CA1 = la représentation CA2

Si le nombre est négative est on veut une représentation VS on convertit d'abord le nombre en base 2 on réserve le bit du pois fort pour le signe = 1 car le nombre est négative

Si non si on veut une représentation CA1 la méthode est comme de celle de VS seulement c'est que les bits 1 et 0 sont inversé mais bien sûr que le bit du poids fort reste égale 1 puisque le nombre enter est négative

Et pour une représentation en CA2 c'est la même que CA1 on ajoute un 1

2. ALGORITHM :

// Fonction main

DEBUT

DECLARATION

nbr, q ← 1, p, t, v : entier

binaire: tableau [] de caractères

s : char

D: **ECRIRE** ("Entrez le type de représentation :(1) pour VS (2) pour CA1 (3) pour CA2 (4) pour Décimal")

LIRE (v)

TRAITEMENT

```
SI(v == 4)
ALORS ALLER A  B
SINONSI(v == 1 || v == 2 || v == 3)
ALORS
   Z: ECRIRE ("Entrez un nombre en binaire (compose de 1 et 0) :\n")
   LIRE (nbr)
   t ← checkB(nbr)
   SI(t == 2)
        ALORS
        ALLER A Z
    SINON
        ALORS
        ECRIRE (binary) // Convertir en binaire sous forme de chaîne
      SWITCH (v)
         cas 1:
            nbr ← VS_D(binary, 8)
            PAUSE
         cas 2:
            nbr ← CA1_D(binary, 8)
            PAUSE
         cas 3:
            nbr ← CA2_D(binary, 8)
            PAUSE
         DEFAUT
            ALLER A D
            PAUSE
      FIN SWITCH

   ALLER A E ;
FINSI
          SINON ALORS ALLER A D
```

FINSI

B: **ECRIRE** ("Entrer un nombre entre -128 et 127 :\n")

LIRE (nbr)

SI(nbr < -pow(2, 7) || nbr > (pow(2, 7) - 1))

ALORS

ECRIRE ("Le nombre n'est pas valide\n")

ALLER A B

FINSI

E: **ECRIRE** ("Entrez le type de représentation souhaite : (1) pour VS (2) pour CA1\n (3) pour CA2 (4) pour Décimal (5) pour tous ")

LIRE (p);

SWITCH (p)

cas 1:

ECRIRE ("La représentation de "**nbr"** en VS est : ")

VS(nbr, 8)

PAUSE

cas 2:

ECRIRE ("La représentation de "**nbr"** en CA1 est : ")

CA1(nbr, 8)

PAUSE

cas 3:

ECRIRE ("La représentation de "**nbr"** en CA2 est : ")

CA2(nbr, 8)

PAUSE

cas 4:

ECRIRE ("La représentation de "**nbr"** en Décimal est : ", nbr)

PAUSE

cas 5:

ECRIRE ("La représentation de "**nbr"** en VS est : ")

VS(nbr, 8)

ECRIRE ("La représentation de "**nbr"** en CA1 est : ")

```
        CA1(nbr, 8)
        ECRIRE ("La représentation de "nbr" en CA2 est : ")
        CA2(nbr, 8)
        PAUSE
      DEFAUT:
        ALLER A  E;
        PAUSE
   FINSWITCH
FINMAIN
// Fonction prend un entier signé et donne sa représentation en VS
FONTION VS(nb : entier, n : entier)
DEBUT
DECLATION
nbrchar : tableau []  de caractères // tableau de caractères ou on va stocker le nombre en binaire
   i = 0, j = 0 ,signe=0 : entier // signe=indicateur de signe
   y, u, k : entier
   verschar: tableau [0 ,1]  de caractères
TRAITEMENT
   SI (nb < 0)
   ALORS
      nb ← nb * (-1) // rendre le nombre négatif vers positif
      signe++ //incrémentation de l'indicateur pour rappeler que le nombre était négatif
   FINSI
   //conversion du b10 au b2 et stocker le resultat dans un tableau de caractères
   POUR i ALLANT DE 0 A 8 :
      nbrchar[i] ← nb % 2 + '0' // Convertir en caractères '0' ou '1'
      nb ← nb / 2
   FINPOUR
   //conversion de int vers char

   POUR y ALLANT DE  0 A i :
```

```
    nbrchar[y] ← verschar[nbrchar[y] - '0']     //la chaine est inversée
  FINPOUR

  POUR j ALLANT DE 0 A n :
    SI (j >= i)
ALORS //remplie le reste à gauche par 00000 car le nombre est positif
        nbrchar[j] ← '0'
    FINSI
  FINPOUR
  SI (signe == 1)
ALORS
          //le nombre est négatif donc le premier digit sera 1
    nbrchar[n - 1] ← '1' //(n-1) car la chaine est inversée
  FINSI
//on inverse la chaine vers la forme normale
  POUR u ALLANT DE n – 1 A 0 PAR PAS DE -1 :
    ECRIRE ("%c", nbrchar[u])
  FINPOUR
FINFONCTION
//fonction qui permet de faire une représentation par complément à 1
FONCTION CA1(nb : entier long, n : entier)
DEBUT
DECLARATION
  nbrchar : tableau [0 …15]  de caractères //tableau de caractères ou on va stocker le nombre en b2
  verschar : tableau [0 ,1]  de caractères
  i = 0, j = 0 : entier // indicateur de signe
  signe = 0 : entier
        y ,u ,k : entier
TRAITEMENT
        //vérification de signe
  SI (nb < 0) ALORS
```

nb ← (pow(2, n) - 1) + nb

signe ← signe +1 //incrémentation de l'indicateur pour rappeler que le nombre était négatif

FINSI

//conversion du b10 au b2 et stocker le résultat dans un tableau de caractères

POUR i **ALLANT DE** 0 **A** 7

FAIRE

nbrchar[i] ← nb % 2 + '0' // Convertir en caractères '0' ou '1'

nb ← nb / 2

FINPOUR

//i=7 ou i=15// i stocke le nombre d'itération

POUR y **ALLANT DE** 0 **A** i

FAIRE

nbrchar[y] ← verschar[nbrchar[y] - '0']

FINPOUR

//la chaine est inversée

//remplie le reste à gauche par 00000 car le nombre est positif

POUR j **ALLANT DE** 0 **A** n

FAIRE

SI (j >= i && signe == 0)

ALORS

nbrchar[j] = '0'

FINSI

//on inverse la chaine vers la forme normale

POUR u **ALLANT DE j** – 1 **A** 0 **PAR PAS DE** -1

ECRIRE ("%c", nbrchar[u])

FINPOUR

FINFONCTION

//fonction qui permet de faire une représentation par complément à 2

FONCTION CA2(nb : entier long, n : entier)

DEBUT

DECLARATION

```
   nbrchar : tableau [0 …15]  de caractères //tableau de caractères ou on va stocker le nombre en b2
   verschar : tableau [0 ,1]  de caractères
   i = 0, j = 0 : entier
   signe = 0 : entier // indicateur de signe
   y,u,k : entier
```

TRAITEMENT

```
   //verification de signe
   SI (nb < 0)
   ALORS
      nb ← pow(2, n) + nb
      signe← signe +1 //incrémentation de l'indicateur pour rappeler que le nombre était négatif
   FINSI
//conversion du b10 au b2 et stocker le résultat dans un tableau de caractères
   POUR  i ALLANT DE  0 A  0
   FAIRE
      nbrchar[i] ← nb % 2 + '0'  // Convertir en caractères '0' ou '1'
      nb ← nb / 2
   FINPOUR
   POUR y ALLANT DE  0 A i
   FAIRE
      nbrchar[y] ← verschar[nbrchar[y] - '0']
   FINPOUR
//la chaine est inversée
//remplie le reste à gauche par 00000 car le nombre est positif
   POUR j ALLANT DE  0 A n :
      SI (j >= i && signe == 0)
        ALORS
         nbrchar[j] ← '0'
      FINSI
   FINPOUR
//on inverse la chaine vers la forme normale
```

```
   POUR  u ALLANT DE  j – 1 A 0 PAR PAS DE -1 :
     ECRIRE (nbrchar[u])
   FINPOUR
FINFONTION
// Fonction pour vérifier que la longueur du nombre entrée est de 8 bits et que
FONCTION checkB(num : entier) : ENTIER
DEBUT
DECLARATION
   j, l : entier
   nbrchar: tableau [0 …150]  de caractères
TRAITEMENT
   ECRIRE (nbrchar, num)
   l ← Longueur(nbrchar)
   TANTQUE (num > 0)
      FAIRE
      j ← num % 10
      SI ((j != 0 && j != 1) || l != 8)
         ALORS
         RETOURNER 2  // Erreur détectée
      FINSI
      num ← num / 10
      SI (num == 0 || l == 8)
         ALORS
         RETOURNER 3 // Succès
      FINSI
   FINTANQUE
   RETOURNER 3  // Cas par défaut
FINFONCTION
// Fonction qui fait le traitement de représentation d'un nombre SIGNEE en CA2 a un décimal
FONCTION CA2_D(binary : tableau []  de caractères, SB : entier) : ENTIER
DEBUT
```

DECLARATION

power ← pow(2, SB - 1) : entier

sum ←0 : entier

i : entier

TRAITEMENT

POUR i **ALLANT DE** 0 **A** SB :

SI (i == 0 && binary[i] != '0')

ALORS

sum ← power * -1

FINSI

SINON

sum ← sum + (binary[i] - '0') * power

FINSINON

power /= 2

FINPOUR

RETOURNER sum

FINFONCTION

// Fonction qui fait le traitement de representation d'un nombre SIGNEE en CA1 a un decimal

FONCTION CA1_D(binary : tableau [] de caractères, SB : entier) : ENTIER

DEBUT

DECLARATION

power ← pow(2, SB - 1) , sum = 0 , i : entier

POUR I **ALLANT DE** 0 **A** SB :

SI (i == 0 && binary[i] != '0')

ALORS

sum ← power * -1 + 1

FINSI

SINON

sum ← sum + (binary[i] - '0') * power

FINSINON

power ← power/ 2

FINPOUR

FINFONCTION

// Fonction qui fait le traitement de représentation d'un nombre SIGNEE en VS a un décimal

FONCTION VS_D(binary : tableau [] de caractères, SB : entier) : ENTIER

DEBUT

DECLARATION

power ← pow(2, SB - 1) , sum = 0, s = 1, i : entier

POUR i **ALLANT DE** 0 **A** SB :

SI (i == 0 && binary[i] != '0')

ALORS

sum ← 0

s ← -1

FINSI

SINON

sum ← sum + (binary[i] - '0') * power

FINSINON

power ← power/ 2

FINPOUR

RETOURNER sum * s

FINFONCTION

FIN

3. PROGRAMME EN C

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
// fonction prend un entier signee et donne sa representation en VS
void VS(int nb, int n) {
   char nbrchar[16]; // tableau de caractères ou on va stocker le nombre en binaire
   int i = 0, j = 0;
   int signe = 0;// indicateur de signe
            int y,u,k;
   if (nb < 0) {
      nb = nb * (-1);// rendre le nombre négatif vers positif
```

```
13      signe++;//incrémentation de l'indicateur pour rappeler que le nombre était
14  négatif
15     }
16     //conversion du b10 au b2 et stocker le resultat dans un tableau de caractères
17     for(i=0; i<8; i++) {
18        nbrchar[i] = nb % 2 + '0'; // Convertir en caractères '0' ou '1'
19        nb = nb / 2;
20     }
21     //conversion de int vers char
22     char verschar[2] = {'0', '1'};
23
24     for ( y = 0; y < i; y++) {
25        nbrchar[y] = verschar[nbrchar[y] - '0'];//la chaine est inversée
26     }
27
28     for (j = 0; j < n; j++) {
29        if (j >= i) { //remplie le reste à gauche par 00000 car le nombre est positif
30           nbrchar[j] = '0';
31        }
32     }
33
34     if (signe == 1) {
35              //le  nombre est négatif donc le premier digit sera 1
36        nbrchar[n - 1] = '1';//(n-1) car la chaine est inversée
37     }
38     //on inverse la chaine vers la forme normale
39     for ( u = n - 1; u >= 0; u--) {
40        printf("%c", nbrchar[u]);
41     }
42     printf("\n");
43  }
44  //fonction qui permet de faire une représentation par complément à 1
45  void CA1(long int nb, int n) {
46     char nbrchar[16];//tableau de caractères ou on va stocker le nombre en binaire
47     int i = 0, j = 0;// indicateur de signe
48     int signe = 0;
49              int y,u,k;
50              //verification de signe
51     if (nb < 0) {
52        nb = (pow(2, n) - 1) + nb;
53        signe++;//incrémentation de l'indicateur pour rappeler que le nombre était
54  négatif
55     }
56     //conversion du b10 au b2 et stocker le resultat dans un tableau de caractères
57     for (i = 0; nb > 0; i++) {
58        nbrchar[i] = nb % 2 + '0'; // Convertir en caractères '0' ou '1'
59        nb = nb / 2;
```

```
60    }
61   //i=7 ou i=15//i stocke le nombre d iteration
62
63              //conversion de int vers char
64    char verschar[2] = {'0', '1'};
65
66    for ( y = 0; y < i; y++) {
67       nbrchar[y] = verschar[nbrchar[y] - '0'];
68    }
69    //la chaine est inversée
70              //
71              //remplie le reste à gauche par 00000 car le nombre est positif
72    for (j = 0; j < n; j++) {
73       if (j >= i && signe == 0) {
74          nbrchar[j] = '0';
75       }
76    }
77    //on inverse la chaine vers la forme normale
78    for ( u = j - 1; u >= 0; u--) {
79       printf("%c", nbrchar[u]);
80    }
81    printf("\n");
82 }
83 //fonction qui permet de faire une représentation par complément à 2
84 void CA2(long int nb, int n) {
85    char nbrchar[16];//tableau de caractères ou on va stocker le nombre en binaire
86    int i = 0, j = 0;
87    int signe = 0;// indicateur de signe
88              int y,u,k;
89              //verification de signe
90    if (nb < 0) {
91       nb = pow(2, n) + nb;
92       signe++;//incrémentation de l'indicateur pour rappeler que le nombre était
93 négatif
94    }
95    //conversion du b10 au b2 et stocker le resultat dans un tableau de caractères
96    for (i = 0; nb > 0; i++) {
97       nbrchar[i] = nb % 2 + '0'; // Convertir en caractères '0' ou '1'
98       nb = nb / 2;
99    }
100   //conversion de int vers char
101   char verschar[2] = {'0', '1'};
102
103   for ( y = 0; y < i; y++) {
104      nbrchar[y] = verschar[nbrchar[y] - '0'];
105   }
106   //la chaine est inversée
```

```
107             //remplie le reste à gauche par 00000 car le nombre est positif
108     for (j = 0; j < n; j++) {
109         if (j >= i && signe == 0) {
110             nbrchar[j] = '0';
111         }
112     }
113     //on inverse la chaine vers la forme normale
114     for ( u = j - 1; u >= 0; u--) {
115         printf("%c", nbrchar[u]);
116     }
117     printf("\n");
118 }
119 // fonction pour verifier que la longeur du nombre entrée est de 8 bits et que
120 int checkB(int num) {
121     int j, l;
122     char nbrchar[160];
123     sprintf(nbrchar, "%d", num);
124     l = strlen(nbrchar);
125 
126     while (num > 0) {
127         j = num % 10;
128 
129         if ((j != 0 && j != 1) || l != 8) {
130             return 2; // Erreur détectée
131         }
132         num = num / 10;
133         if (num == 0 || l == 8) {
134             return 3; // Succès
135         }
136     }
137     return 3; // Cas par défaut
138 }
139 // Fonction qui fait le traitement de representation d'un nombre SIGNEE en CA2 a un
140 decimal
141 int CA2_D(char binary[], int SB) {
142     int power = pow(2, SB - 1);
143     int sum = 0;int i;
144     for ( i = 0; i < SB; ++i) {
145         if (i == 0 && binary[i] != '0') {
146             sum = power * -1;
147         } else {
148             sum += (binary[i] - '0') * power;
149         }
150         power /= 2;
151     }
152 
153     return sum;
```

```
154 }
155 // Fonction qui fait le traitement de representation d'un nombre SIGNEE en CA1 a un
156 decimal
157 int CA1_D(char binary[], int SB) {
158     int power = pow(2, SB - 1);
159     int sum = 0;int i;
160     for ( i = 0; i < SB; i++) {
161         if (i == 0 && binary[i] != '0') {
162             sum = power * -1 + 1;
163         } else {
164             sum += (binary[i] - '0') * power;
165         }
166         power /= 2;
167     }
168 
169     return sum;
170 }
171 // Fonction qui fait le traitement de representation d'un nombre SIGNEE en VS a un
172 decimal
173 int VS_D(char binary[], int SB) {
174     int power = pow(2, SB - 1);
175     int sum = 0;
176     int s = 1;int i;
177     for ( i = 0; i < SB; i++) {
178         if (i == 0 && binary[i] != '0') {
179             sum = 0;
180             s = -1;
181         } else {
182             sum += (binary[i] - '0') * power;
183         }
184         power /= 2;
185     }
186     return sum * s;
187 }
188 int main() {
189     int nbr, q = 1, p, t, v;
190     char binaire[8];
191     char s;
192     int num;
193 
194     D: printf("Entrez le type de representation :\n ( 1 ) pour VS\n ( 2 ) pour CA1\n ( 3 )
195 pour CA2\n ( 4 ) pour Decimal\n");
196     scanf("%d", &v);
197 
198     if (v == 4) {
199         goto B;
200     } else if (v == 1 || v == 2 || v == 3){
```

```
201       Z: printf("Entrez un nombre en binaire (compose de 1 et 0) :\n");
202       scanf("%d", &nbr);
203       t = checkB(nbr);
204       if (t == 2) {
205          goto Z;
206       } else {
207             num=nbr;
208          snprintf(binaire, sizeof(binaire), "%d", nbr); // Convertir en binaire sous forme
209 de chaîne
210          switch (v) {
211             case 1:
212                nbr = VS_D(binaire, 8);
213                break;
214             case 2:
215                nbr = CA1_D(binaire, 8);
216                break;
217             case 3:
218                nbr = CA2_D(binaire, 8);
219                break;
220             default:
221                goto D;
222                break;
223          }
224       }
225       goto E;
226    } else goto D;
227
228    B: printf("Entrer un nombre entre -128 et 127 :\n");
229    scanf("%d", &nbr);
230    num=nbr;
231    if (nbr < -pow(2, 7) || nbr > (pow(2, 7) - 1)) {
232       printf("Le nombre n'est pas valide\n");
233       goto B;
234    }
235    E: printf("Entrez le type de representation souhaite :\n (1) pour VS\n (2) pour
236 CA1\n (3) pour CA2\n (4) pour Decimal\n (5) pour Tous\n");
237    scanf("%d", &p);
238    switch (p) {
239       case 1:
240          printf("La representation de %d en VS est : ", num);
241          VS(nbr, 8);
242          break;
243       case 2:
244          printf("La representation de %d en CA1 est : ", num);
245          CA1(nbr, 8);
246          break;
247       case 3:
```

```
248         printf("La representation de %d en CA2 est : ", num);
249         CA2(nbr, 8);
250         break;
251       case 4:
252         printf("La representation de %d en Decimal est : %d\n", num, nbr);
253         break;
254       case 5:
255             printf("La representation de %d en Decimal est : %d \n", num,nbr);
256         printf("La representation de %d en VS est : ", num);
257         VS(nbr, 8);
258         printf("La representation de %d en CA1 est : ", num);
259         CA1(nbr, 8);
260         printf("La representation de %d en CA2 est : ", num);
261         CA2(nbr, 8);
262         break;
263       default:
264         goto E;
265         break;
266     }
267
268     printf("\n");
269     while (q != 0) {
270       S: printf("Pour Une Autre Representation tapez C !, Pour RESSAYER tapez R !,
271 Pour QUITTER tapez Q !\n");
272       scanf(" %c", &q);
273       if (q == 'Q' || q == 'q') {
274         printf("FIN DU PROGRAMME\n");
275         return 0;
276       } else if (q == 'C' || q == 'c') {
277         goto E;
278       } else if (q == 'R' || q == 'r') {
          goto D;
        } else {
          goto S;
        }
      }
    }
```

4. Captures d'écran de l'exécution (les tests)

1. Exemple pour -61 du décimal vers VS, CA1, CA2 et inversement :

```
Entrez le type de representation :
 ( 1 ) pour VS
 ( 2 ) pour CA1
 ( 3 ) pour CA2
 ( 4 ) pour Decimal
4
Entrer un nombre entre -128 et 127 :
-61
Entrez le type de representation souhaite :
 (1) pour VS
 (2) pour CA1
 (3) pour CA2
 (4) pour Decimal
 (5) pour Tous
5
La representation de -61 en Decimal est : -61
La representation de -61 en VS est : 10111101
La representation de -61 en CA1 est : 11000010
La representation de -61 en CA2 est : 11000011

Pour Une Autre Representation tapez C !, Pour RESSAYER tapez R !, Pour QUITTER tapez Q !
```

```
Entrez le type de representation :
 ( 1 ) pour VS
 ( 2 ) pour CA1
 ( 3 ) pour CA2
 ( 4 ) pour Decimal
1
Entrez un nombre en binaire (compose de 8 BITS) :
10111101
Entrez le type de representation souhaite :
 (1) pour VS
 (2) pour CA1
 (3) pour CA2
 (4) pour Decimal
 (5) pour Tous
4
La representation de 10111101 en Decimal est : -61
```

```
Entrez le type de representation :
 ( 1 ) pour VS
 ( 2 ) pour CA1
 ( 3 ) pour CA2
 ( 4 ) pour Decimal
2
Entrez un nombre en binaire (compose de 8 BITS) :
11000010
Entrez le type de representation souhaite :
 (1) pour VS
 (2) pour CA1
 (3) pour CA2
 (4) pour Decimal
 (5) pour Tous
4
La representation de 11000010 en Decimal est : -61
```

```
Entrez le type de representation :
 ( 1 ) pour VS
 ( 2 ) pour CA1
 ( 3 ) pour CA2
 ( 4 ) pour Decimal
3
Entrez un nombre en binaire (compose de 8 BITS) :
11000011
Entrez le type de representation souhaite :
 (1) pour VS
 (2) pour CA1
 (3) pour CA2
 (4) pour Decimal
 (5) pour Tous
4
La representation de 11000011 en Decimal est : -61
```

2. Exemple pour 51

```
Entrez le type de representation :
 ( 1 ) pour VS
 ( 2 ) pour CA1
 ( 3 ) pour CA2
 ( 4 ) pour Decimal
4
Entrer un nombre entre -128 et 127 :
51
Entrez le type de representation souhaite :
 (1) pour VS
 (2) pour CA1
 (3) pour CA2
 (4) pour Decimal
 (5) pour Tous
5
La representation de 51 en Decimal est : 51
La representation de 51 en VS est : 00110011
La representation de 51 en CA1 est : 00110011
La representation de 51 en CA2 est : 00110011

Pour Une Autre Representation tapez C !, Pour RESSAYER tapez R !, Pour QUITTER tapez Q !
```

3. Exemple pour -127

```
Entrez le type de representation :
 ( 1 ) pour VS
 ( 2 ) pour CA1
 ( 3 ) pour CA2
 ( 4 ) pour Decimal
4
Entrer un nombre entre -128 et 127 :
-127
Entrez le type de representation souhaite :
 (1) pour VS
 (2) pour CA1
 (3) pour CA2
 (4) pour Decimal
 (5) pour Tous
5
La representation de -127 en Decimal est : -127
La representation de -127 en VS est : 11111111
La representation de -127 en CA1 est : 10000000
La representation de -127 en CA2 est : 10000001
```

4. Exemple pour -128

```
Entrez le type de representation :
 ( 1 ) pour VS
 ( 2 ) pour CA1
 ( 3 ) pour CA2
 ( 4 ) pour Decimal
4
Entrer un nombre entre -128 et 127 :
-128
Entrez le type de representation souhaite :
 (1) pour VS
 (2) pour CA1
 (3) pour CA2
 (4) pour Decimal
 (5) pour Tous
5
La representation de -128 en Decimal est : -128
La representation de -128 en VS est : 10000000
La representation de -128 en CA1 est :  1111111
La representation de -128 en CA2 est : 10000000
```

Sujet 3 : Représentation des nombres à virgule fixe et virgule flottante

1. *ANALYSE :*

Objectifs :

Un programme permettant de représenter les nombres à virgule fixe et virgule flottante.

- Codage IEEE 754 (Simple précision 32 bits)
- Codage IEEE 754 (Double précision 64 bits)

Analyse de problème :

La méthode pour représenter un nombre réel en virgule fixe :

Supposons qu'on a un réel n négative

Le premier bit à gauche sera pour la position dans ce cas pos=1(car nombre négative), on prend la partie entière positive on la convertit en binaire en faisant la division entière par 2

Après pour la partie fractionnaire supposons f=n-e (avec e : la partie entière, f : la partie fractionnaire et n le nombre réel) on multiplie f par 2 si le reste r< 1 on prend 0 et on faire la multiplication une autre fois par 2 sinon on prend 1 et on continue la multiplication avec r-1

Et ainsi de suite on continue la multiplication jusqu'on veut

Le résultat donc sera n=pos e. Partie fractionnaire

Exemple :

12. 375 En virgule fixe ?

12.375 est positive donc pos = 0

La partie entière de 12.375 est 12 en binaire c'est 1100

La partie fractionnaire est 0.375 en binaire : 0.375 *2=0.75=0+0.75

0.75 *2=1.5=1+0.5

0.5 *2=1=1+0

D'où 0.375= 0.011

Donc le résultat final sera 12.375=0 1100.011

La méthode pour représenter une virgule fixe en nombre réel :

Pour l'inverse on commence par un nombre binaire on voit le signe par le premier bit à gauche si 0 doc positive sinon c'est négatif

Puis on convertit la partie entière en décimal comme normal et la partie fractionnaire on multipliant par les 2^{-i} avec i est la position du bit respectivement de la gauche à droite

Exemple :

11101.011 en réel ?

Le premier bit est 1 donc négative

La partie entière est 1101 c'est 13 en décimal

La partie fractionnaire est 0.011=$2^{-2}+2^{-3}$= 0.25 +0.125=0.375

Donc le résultat final sera 1 1101.011=-13.375

La méthode pour représenter un nombre réel en virgule flottante: (Simple précision 32 bits)

La représentation en simple précision (32 bits) est sous la forme : S|E_b|M

S=1 bit pour le signe

E=8 bits pour l'exposant

M=23 bits pour la mantisse

Pour représenter un nombre réel N en virgule flottante la première chose on va voire est la partie entière de N, elle doit-être appartient à [-$2^{16}+1$, $2^{16}-1$] et le signe si positif S=0 si négatif S=1

Après on calcul l'exposant on fait convertir N en virgule fixe sans le bit de signe et on écrasons la virgule à gauche jusqu'à ce qu'on arrive au bit du poids fort (premier 1 à gauche), disons que nous avons écraser de p bits (si on avait écrasé à droite don -p) donc E=p et en déduit que l'exposant biaisé est $E_b = 127 + p$ (on le code en binaire),la partie après le 1 est la mantisse on compètent par des 0 si le nombre de bit est <23 jusqu'à ce qu'on arrive à 23 bits

Exemple :

-13.375 en virgule flottante (Simple précision 32 bits) :

- Le signe est négatif alors S=1
  13 ∈[-$2^{16}+1$, $2^{16}-1$] donc on peut le codé sur 32bits
- 13.375 en virgule fixe est 1101.011 on écrase jusqu'à le premier 1 à gauche on aura 1.101011× 2^3 ,E= 3 et E_b=127+3=130 en binaire sera E_b=10000010
- La mantisse M=101011 et puisque le nombre de bits est <23 on doit compléter par des 0 M=1010 1100 0000 0000 0000 000

Et donc le résultat final sera :

-13.375 = 1 | 1000 0010 | 1010 1100 0000 0000 0000 000

La méthode pour représenter un nombre réel en virgule flottante : (Simple précision 64 bits)

La représentation en simple précision (64 bits) est sous la forme : S|E_b|M

S=1 bit pour le signe

E=11 bits pour l'exposant

M=52 bits pour la mantisse

Pour représenter un nombre réel N en virgule flottante la première chose on va voire est la partie entière de N, elle doit-être appartient à [-$2^{32}+1$, $2^{32}-1$] et le signe si positif S=0 si négatif S=1

Après on calcul l'exposant on fait convertir N en virgule fixe sans le bit de signe et on écrasons la virgule à gauche jusqu'à ce qu'on arrive au bit du poids fort (premier 1 à gauche), disons que nous avons écraser de p bits(si on avait écrasé à droite don -p) donc E=p et en déduit que l'exposant biaisé est $E_b = 1023 + p$ (on le code en binaire), la partie après le 1 est la mantisse on compètent par des 0 si le nombre de bit est <52 jusqu'à ce qu'on arrive à 52 bits

Exemple :

117.375 en virgule flottante (Simple précision 32 bits) :

- Le signe est négatif alors S=0
  117 ∈[-$2^{32} + 1$, $2^{32} - 1$] donc on peut le codé sur 64bits
- 117.375 en virgule fixe est 111 0101.011 on écrasons jusqu'à le premier 1 à gauche on aura 1.110101011×2^6 ,E= 6 et E_b=1023+6=1029 en binaire sera E_b=100 0000 0101
- La mantisse M=110101011 et puisque le nombre de bits est <52 on doit compléter par des 0
  M=1101 0101 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Et donc le résultat final sera :

117.375 =0 | 100 0000 0101| 1101 0101 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

2. ALGORITHME

```
//Fonction qui permet de convertir un nombre réel vers un nombre binaire
FONCTION  Vfixe (N : double)
DEBUT
DECLARATION
  B, BA : DOUBLE
  A, Z, T, C : ENTIER
  I, j, k : ENTIER
TRAITEMENT
  i= 0
  k=0
  //vérification de signe
  SI(N<0)
    ALORS
    N← (-1)*N
    ECRIRE  ("1 ")
  FINSI
  SINON
    ECRIRE  ("0 ")
  FINSINON
  A←E(N)  //stocker la partie entière
  B←N-A  //stocker la partie fractionnaire
  BA←B
  TANTQUE (A ! = 0)
    FAIRE
    //conversion vers binaire
    T[i] ← (A%2)
```

```
    i++
    A ← A/2
  FINTANTQUE
  TANTQUE (B!=0)
    FAIRE
    //0.375=?
    // 0.375 x 2 = 0.75
    // 0.75x 2 = 1.5
    // 0.5x 2 = 1.0
    C[k]  ← E (B)
    Z←E (B)
    k++
    SI (k== (16)+1)
      ALORS
      PAUSE
      FINSI
    B← (B-Z)*2
  FINTANTQUE
  //Affichage
  POUR j ALLANT DE i-1 A 0 AVEC PAS DE -1 :
    ECRIRE (T[j])
  FINPOUR
  ECRIRE  (".")
  SI (BA ==0)
    ECRIRE (0)
  FINSI
  POUR j ALLANT DE  1 A k :
    ECRIRE (C[j])
  FINPOUR
FINFONCTION
//Fonction qui permet de convertir un nombre binaire vers un nombre réel
FONCTION bin_to_fixe(^num: chaine de char) :
DEBUT
DECLARATION
   i, k, s: ENTIER
   resultat1, resultat2: ENTIER LONG
TRAITEMENT
   i← 0
   k← 0
   s← 0
   resultat1← 0
   resultat2← 0
   //l'emplacement de la virgule
   TANTQUE (num[s]!='.')
    FAIRE
    s++
   FINTANTQUE
```

```
   k← s-1
   //conversion de binaire vers décimal
   POUR i ALLANT DE 0 A s :
     resultat1← resultat1+(num[i]-'0')*pow(2,k)
     k--
   FINPOUR
   k← -1 //puissance partie fractionnaire
   //conversion de binaire vers décimal
   POUR i ALLANT DE s+1 A Longueur(num) :
      resultat2← resultat2+((num[i]-'0')*pow(2,k)*pow(10,Longueur(num)-s-1))
      k--
  FINPOUR
  //Affichage
  ECRIRE ("La Conversion de ")
  POUR I ALLANT DE 0 A Longueur(num) :
    ECRIRE (num[i])
    ECRIRE ("    est égale à ,resultat1)
    ECRIRE (".")
    ECRIRE (resultat2)
  FINPOUR
FINFONCTION
//Fonction pour convertir un nombre décimal en virgule flottante 32
V_flottant32 (N : DOUBLE) :
DEBUT
DECLARATION
  A, N1, Z, i, j, k, a, v, u : ENTIER
  ^T, ^C, ^P, ^S: ENTIER
  B, D : DOUBLE
TRAITEMENT
  v←0
  //vérification de signe
  SI (N<0)
    ALORS
    N← N*(-1)
    ECRIRE  ("1 ")
  FINSI
  SINON
    ECRIRE  ("0 ")
  FINSINON
  A←E (N)
  N1←E (N)  //stocker la partie entière
  B← N-A
  D ←  N-A  //stocker la partie fractionnaire
   i ←  0
   k ←  0
   TANTQUE (A != 0)
     FAIRE
```

```
   //conversion vers binaire
   T[i]  ← (A%2)
   i++
   A ←  A/2
FINTANTQUE
TANTQUE (B != 0)
   FAIRE
   E (B) //Partie entiere
   C[k]  ←E (B)
   Z←E (B)
   k++
   SI (k←52)
     ALORS
     PAUSE
   FINSI
   B ← (B-Z)*2
   FINTANTQUE
   SI (N1!=0)
     ALORS
     a ← i+126 //Exposant
     P← malloc (8* sizeof (int))
     S← malloc (23* sizeof (int))  //Exposant vers binaire
    TANTQUE (a != 0)
      FAIRE
      P[v]  ← (a % 2)
      v++
      a ←  a/2
    FINTANTQUE
    //Affichage
    POUR j  ALLANT DE v-1 A 0 PAR PAS DE -1 :
      ECRIRE  (P[j])
      ECRIRE  ("  ")
    FINPOUR
    POUR j ALLANT DE 0 A 23 :
      S[j]  ← 0
    FINPOUR
    POUR j ALLANT DE 0 A i-1 :
      S[j]  ← T[i-j-2]
    FINPOUR
    POUR j ALLANT DE 1 A k :
      S[j+i-2]  ← C[j]
    FINPOUR
    POUR j ALLANT DE 0 A 23 :
      ECRIRE (S[j])
    FINPOUR
 FINSI
```

```
    SINON
     SI (D == 0)
      ALORS
      POUR j ALLANT DE 0 A 7 :
        ECRIRE ("0")
        ECRIRE (" ")
      FINPOUR
      POUR j ALLANT DE 0 A 23 :
      ECRIRE ("0")
      FINPOUR
     FINSI
     SINON
      POUR j ALLANT DE 1 A k :
       SI(C[j]← 1)
         ALORS u← j
         PAUSE
       FINSI
      FINPOUR
      a← -u+127
      v← 0
      P=malloc (8 *sizeof (int))
      S=malloc (23* sizeof (int))
      POUR j ALLANT DE 0 A 8 :
        P[j]  ← 0
      FINPOUR
      TANTQUE (a!= 0)
        FAIRE
        P[v]  ← (a % 2)
        v++
        a ←  a/2
      FINTANTQUE
      POUR j ALLANT DE 7 A 0 PAR PAS DE -1 :
        ECRIRE (P[j])
      FINPOUR
      ECRIRE (" ")
      POUR j ALLANT DE 0 A 23 :
        S[j]  ← 0
      FINPOUR
      POUR j ALLANT DE u+1 A k :
        S[j- (u+1)]  ←  C[j]
      FINPOUR
    FINSINON
  FINSI
FINFONCTION
//Fonction pour convertir un nombre décimal en virgule flottante 64
FONCTION V_flottant64 (N : DOUBLE)
DEBUT
```

```
DECLARATION
  A, N1, Z, a , v : ENTIER
  i, j, k,u  : ENTIER
  B, D, : DOUBLE
  ^T, ^C, ^P, ^S : ENTIER
TRAITEMENT
  //vérification de signe
 SI (N<0)
    ALORS
    N← N*(-1)
    ECRIRE  ("1 ")
 FINSI
 SINON
    ECRIRE  ("0 ")
 FINSINON
 //stocker la partie entière
 A←E (N)
 N1←E (N)
 //stocker la partie fractionnaire
 B← N-A
 D ←N-A
  i ←  0
  k ←  0
  TANTQUE (A != 0)
    FAIRE
    T[i]  ←  (A%2)
    i++
    A ←  A/2
  FINTANTQUE
  TANTQUE (B ! = 0)
    FAIRE
   //0.375=?
   // 0.375 x 2 = 0.75
   // 0.75 x 2 = 1.5
   // 0.5   x 2 = 1.0
   E  (B)
   C[k]  ←E (B)
   Z←E (B)
   k++
   SI (k←52)
     ALORS
     PAUSE
   B ← (B-Z)*2
   FINSI
   SI (N1!=0)
    ALORS
```

```
  a ← i+1022
  v← 0
  ^P←malloc (11 *sizeof (int))
  ^S←malloc (52 *sizeof (int))
  //conversion d'Exposant vers binaire
 TANTQUE (a != 0)
   FAIRE
   P[v]  ← (a % 2)
   v++
   a ←  a/2
 FINTANTQUE
 //Affichage
 POUR j ALLANT DE v-1 A 0 PAR PAS DE -1 :
   ECRIRE  (P[j])
 FINPOUR
 ECRIRE  ("  ")
 POUR j ALLANT DE 0 A 52 :
   S[j]  ← 0
 FINPOUR
 POUR j ALLANT DE 0 A i-1 :
   S[j]  ← T[i-j-2]
 FINPOUR
 POUR j ALLANT DE 1 A k :
   S[j+i-2]  ←  C[j]
 FINPOUR
 POUR j ALLANT DE 0 A 52 :
   ECRIRE (S[j])
 FINPOUR
FINSI
SINON
 SI (D ←  0)
  ALORS
  POUR j ALLANT DE 0 A 10 :
    ECRIRE ("0")
  FINPOUR
    ECRIRE (" ")
  POUR j ALLANT DE 0 A 52 :
    ECRIRE ("0")
  FINPOUR
 FINSI
 SINON
  POUR  j ALLANT DE 1 A k :
    SI (C[j]←1)
      ALORS
      u← j
      PAUSE
```

```
        a← -u+1022
        v← 0
        ^P← malloc (11 *sizeof (int))
        ^S← malloc (52 *sizeof (int))
        POUR j ALLANT DE 0 A 11 :
          P[j]← 0
        FINPOUR
        TANTQUE (a != 0)
          FAIRE
          P[v]← (a % 2)
          v++;
          a← a/2
        FINTANTQUE
        POUR j ALLANT DE 11 A 0 PAR PAS DE -1 :
          ECRIRE  (P[j])
        FINPOUR
          ECRIRE (" ")
        POUR j ALLANT DE 0 A 52 :
          S[j]←0
        FINPOUR
        POUR j ALLANT DE u+1 A k :
          S[j- (u+1)]  ←C[j]
        FINPOUR
    FINSINON
  FINSI
FINFONCTION
// Fonction main
FONCTION main() : ENTIER
DEBUT
DECLARATION
  i, v, q, L: ENTIER
  num : FLOAT
  n[10000] bit[10], nbr[64], choix[1] : chaine de char
TRAITEMENT
  q←1
  A:
   ECRIRE ("Sélectionnez l'opération : ")
   ECRIRE ("1. decimal to virgule fixe")
   ECRIRE ("2. Virgule fixe to décimal")
   ECRIRE ("3. Conversion réel vers virgule flottante")
   LIRE (choix)
   L=Longueur(choix)
    SI (choix[0]=='1'&&L==1)  //CHOIX 1
      ALORS
      ECRIRE ("Entrer un nombre réel :")
      LIRE (n)
      num  ← n
```

```
 SI (num < -pow(2, 16)|| num >(pow(2, 15)) )
   ALORS
   ECRIRE ("Le nombre n'est pas valide")
   ALLER A  A
  FINSI
  ECRIRE (" La représentation en virgule fixe est : " )
  Vfixe(num)
 SINONSI (choix[0] == '2'&&L==1) //CHOIX 2
  ALORS
  AA:
  ECRIRE ("Donner un nombre en virgule fixe représenter en BINAIRE")
  LIRE (nbr)
  v←0
  POUR i ALLANT DE 0 A 65 :
    SI (nbr[i]←'.')
     ALORS
     v++
    FINSI
  FINPOUR
  SI(v!=1)
    ALORS
    ECRIRE ("le nombre doit être contient une Virgule")
    ALLER A  AA
  FINSI
  bin_to_fixe(nbr)
FINSINONSI
SINONSI (choix[0] ← '3' && L ←1)  //CHOIX 3
  ALORS
  B:
  ECRIRE  ("Entrer un nombre REEL :")
  LIRE  (n)
   num ←n
  bit1:
  ECRIRE ("Saisissez le nombre de bits (32 ou 64): ")
  LIRE (bit)
  SI(bit != 32 && bit != 64)
   ALORS
   ECRIRE ("Nombre de bits invalide (doit être 32 ou 64) ")
   ALLER A  bit1
  FINSI
  // CHOIX ENTRE 32 ET 64
  SI(bit ==32)
    ALORS
    // Virgule Flottante sur 32 bits
    SI(num < -pow(2, 16)|| num >(pow(2, 16))) )
     ALORS
     ECRIRE ("Le nombre ne peut pas être présenter  en simple précision ")
```

```
      ALLER A  B
    FINSI
    ECRIRE (" La représentation en virgule flottante (simple précision) est : ", n )
    V_flottant32(num)
  FINSI
  SINON
    ALORS
    // Virgule Flottante sur 64 bits
    SI(num < -pow(2, 32)|| num >(pow(2, 32))) )
      ALORS
      ECRIRE ("Le nombre ne peut pas être présenter en double précision ")
      ALLER A  B
    FINSI
    ECRIRE (" La représentation en virgule flottante (double précision) est : " )
    V_flottant64(num)
  FINSINON
  SINON
    ALORS
    ECRIRE  ("le choix n'est pas valide")
    ALLER A A;
  FINSINON

FINFONCTION
```

3. PROGRAMME EN C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
//Fonction qui permet de convertir un nombre reel vers un nombre binaire
void Vfixe (double N){
            //verification de signe
             if(N<0){
            N=(-1)*N; printf("1 ");}
            else printf("0 ");
            int A=floor(N);//stocker la partie entiere
            double B=N-A;      //stocker la partie fractinnaire
            double BA=B;
int Z;int *T=malloc(sizeof(int));
int *C=malloc(sizeof(int));
int i=0, j, k = 0;
            while(A != 0) {

T=(int*)realloc(T, (i+1)* sizeof(int)); //reallocation de tableau
                        //conversion vers binaire
```

```
21                                  T[i] =(A%2);
22                                  i++;
23                                  A = A/2;
24              }
25              while(B!=0) {
26              //0.375=?
27              // 0.375 x 2 = 0.75
28              // 0.75  x 2 = 1.5
29              // 0.5   x 2 = 1.0
30                      C = (int*)realloc(C, (k+1)* sizeof(int));
31                      C[k]=floor(B);
32                      Z=floor(B);
33                      k++;
34                      if(k==(16)+1) break;
35                      B=(B-Z)*2;
36              }
37              //Affichage
38                      for(j=i-1;j>=0; j--) {
39                              printf("%d",T[j]);
40                      }
41  printf(".");
42  if(BA==0) printf("0");
43                      for(j=1;j<k; j++) {
44                      printf("%d",C[j]);
45                      }
46  printf("\n");
47  }
48  //Fonction qui permet de convertir un nombre binaire vers un nombre reel
49  void bin_to_fixe(char* num)
50  {
51              int i,k,s=0;
52              long long  int resultat1=0,resultat2=0;
53              //l'emplacement de la virgule
54              while(num[s]!='.') {
55                      s++;
56              }
57              k=s-1;
58              //conversion de binaire vers decimal
59              for (i=0; i<s; i++) {
60                      resultat1=resultat1+(num[i]-'0')*pow(2,k);
61                      k--;
62              }
63
64              k=-1;//puissance partie fractionnaire
65              //conversion de binaire vers decimal
66              for (i=s+1; i<strlen(num); i++) {
67              resultat2=resultat2+((num[i]-'0')*pow(2,k)*pow(10,strlen(num)-s-1));
```

```
68              k--;
69              }
70              //Affichage
71              printf("La Conversion  de ");
72              for(i=0; i<strlen(num); i++)
73                              printf("%c",num[i]);
74              printf("  est egale a   %lld",resultat1);
75              printf(".");
76              printf("%lld",resultat2);
77     printf("\n");
78  }
79  //Fonction pour convertir un nombre decimal en virgule fixe
80  void V_flottant32 (double N){
81              //verification de signe
82               if (N<0) {
83              N=N*(-1);
84              printf("1 ");
85
86              } else printf("0 ");
87
88              int A=floor(N),N1=floor(N);//stocker la partie entiere
89              double B=N-A, D = N-A;//stocker la partie fractinnaire
90              int Z;
91              int *T= malloc(sizeof(int));//declaration de tableau
92              int *C= malloc(sizeof(int));
93              int i = 0, j, k = 0;
94                              while (A != 0) {
95                              T = (int*) realloc (T, (i+1) * sizeof (int)); //reallocation de tableau
96                              //conversion vers binaire
97                              T[i]=(A%2);
98                              i++;
99                              A = A/2;
100                             }
101             while (B != 0) {
102             C = (int*) realloc (C, (k+1) * sizeof(int)); floor (B);
103             C[k]=floor(B);
104             Z=floor(B);
105             k++;
106             if (k==25) break;
107             B = (B-Z)*2;
108             }if (N1!=0) {int a =i+126,v=0;//Exposant
109             int *P=malloc (8* sizeof (int));
110             int *S=malloc (23* sizeof (int));//Exposant to binaire
111             while (a != 0) {
112             P[v]=(a % 2);
113             v++;
114             a = a/2;
```

```
115     }
116             //Affichage
117             for(j=v-1;j>=0;j--) printf("%d", P[j]);
118             printf("  ");
119 for(j=0;j<23;j++) S[j]=0;
120 for(j=0;j<i-1;j++) S[j]=T[i-j-2];
121 for(j=1;j<k;j++) S[j+i-2]=C[j];
122             for(j=0;j<23;j++) printf("%d",S[j]);
123 }else {
124             if (D ==0){for(j=0;j<=7; j++) printf("0");
125             printf(" "); for(j=0;j<23; j++) printf("0");
126 }else{
127              int u;
128             for(j=1;j<k; j++) {
129                         if (C[j]==1) {u=j; break; }
130             }
131             int a=-u+127,v=0;
132             int *P=malloc (8 *sizeof (int));
133             int *S=malloc (23* sizeof (int));
134             for(j=0;j<8;j++) P[j]=0;
135             while (a != 0) {
136             P[v]=(a % 2);
137             v++;
138             a = a/2;
139             }
140 for(j=7;j>=0;j--) printf("%d", P[j]);
141 printf(" ");
142 for(j=0;j<23;j++) S[j]=0;
143 for(j=u+1;j<k;j++) S[j- (u+1)]=C[j];
144 }
145 }
146 }
147 void V_flottant64 (double N){
148                         //verification de signe
149                          if (N<0) {
150                         N=N*(-1);
151                         printf("1 ");
152 
153                         } else printf("0 ");
154             //stocker la partie entiere
155             int A=floor(N),N1=floor(N);
156             //stocker la partie fractinnaire
157             double B=N-A, D = N-A;int Z;
158             int *T= malloc(sizeof(int));
159             int *C= malloc(sizeof(int));
160             int i = 0, j, k = 0;while (A != 0) {
161             T = (int*) realloc (T, (i+1) * sizeof (int)); //reallocation de tableau
```

```
T[i]=(A%2);
i++;
A = A/2;
}while (B != 0) {
            //0.375=?
            // 0.375 x 2 = 0.75
            // 0.75  x 2 = 1.5
            // 0.5   x 2 = 1.0
C = (int*) realloc (C, (k+1) * sizeof(int)); floor (B);
C[k]=floor(B);
Z=floor(B);
k++;
if (k==52) break;
B = (B-Z)*2;
}
if (N1!=0) {int a =i+1022,v=0;
            int *P=malloc (11 *sizeof (int));
            int *S=malloc (52 *sizeof (int));
                        //conversion d'Exposant vers binaire
                        while (a != 0) {
                        P[v]=(a % 2);
                        v++;
                        a = a/2;
                        }
            //Affichage
for(j=v-1;j>=0;j--) printf("%d", P[j]);
printf("  ");
for(j=0;j<52;j++) S[j]=0;
for(j=0;j<i-1;j++) S[j]=T[i-j-2];
for(j=1;j<k;j++) S[j+i-2]=C[j];
for(j=0;j<52;j++) printf("%d",S[j]);
            }
            else {
                        if (D ==0){
                        for(j=0;j<=10; j++) printf("0");
                        printf(" ");
                        for(j=0;j<52; j++) printf("0");
                        }           else {
                                     int u;
                                    for(j=1;j<k; j++) { if (C[j]==1) {u=j; break; }}
                                    int a=-u+1022,v=0;
                                    int *P=malloc (11 *sizeof (int));
                                    int *S=malloc (52 *sizeof (int));
                         for(j=0;j<11;j++) P[j]=0;
                                                while (a != 0) {
                                                P[v]=(a % 2);
                                                v++;
```

```
209                                         a = a/2;
210                                         }
211
212                         for(j=11;j>=0;j--) printf("%d", P[j]);
213                         printf(" ");
214                         for(j=0;j<52;j++) S[j]=0;
215                         for(j=u+1;j<k;j++) S[j- (u+1)]=C[j];
216                         }
217             }
218 }
219
220 int main(){
221             int i,v,q = 1;
222             char n[10000];
223             char bit[10];
224             char nbr[64];
225             char choix[1];
226 A:
227    printf("Selectionnez l'operation:\n");
228    printf("1. decimal To virgule fixe\n");
229    printf("2. vergule fixe to decimal\n");
230    printf("3. Conversion reel vers virgule flottante\n");
231    scanf("%s", &choix);
232    int ll=strlen(choix);
233    if (choix[0]=='1'&&ll==1){//CHOIX 1
234             printf("Entrer un nombre reel:\n");
235    scanf("%s", &n);
236    double num;
237    sscanf(n, "%lf", &num);// sscanf convertit char en double
238
239
240    if (num < -pow(2, 16)|| num >(pow(2, 15)) ){
241       printf("Le nombre n'est pas valide\n");
242       goto A;
243    }
244             printf("La representation de %s en virgule fixe est : ",n);
245             Vfixe(num);
246             }
247             else if (choix[0] == '2'&&ll==1){//CHOIX 2
248               AA:  printf("Donner un nombre en virgule fixe representer en BINAIRE\n");
249    scanf("%s",&nbr);
250    v=0;
251    for(i=0;i<65;i++){
252             if(nbr[i]=='.') v++;
253             }
254             if(v!=1){
255                         printf("le nombre doit etre contient une Virgule\n");
```

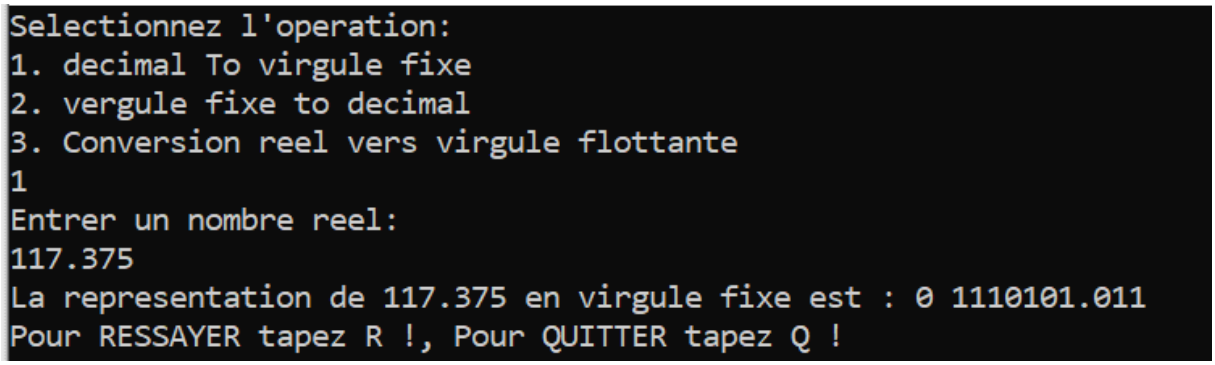
```
256                         goto AA;
257             }
258
259     bin_to_fixe(nbr);
260             }
261     else if (choix[0] == '3'&&ll==1){//CHOIX 3
262     B: printf("Entrer un nombre REEL :\n");
263     scanf("%s", &n);
264     float num=atof(n);
265     bit1:       printf("Saisissez le nombre de bits (32 ou 64): ");
266       scanf("%s", &bit);
267                         if (atoi(bit) != 32 && atoi(bit) != 64) {
268                             printf("Nombre de bits invalide (doit etre 32 ou 64).\n");
269                             goto bit1 ;
270                           }
271
272
273     // CHOIXXX
274     if(atoi(bit)==32){
275             // Virgule Flottante sur 32 bits
276             if (num < -pow(2, 16)|| num >(pow(2, 16) ){
277                printf("Le nombre ne peut pas etre presenter en simple precision\n");
278               goto B;
279             }
280
281             printf("La representation de %s en virgule flottante (simple precision)
282 est :\n");
283
284                         V_flottant32(num);
285                                 printf("\n");
286             } else {
287                         // Virgule Flottante sur 64 bits
288             if (num < -pow(2, 32)|| num >(pow(2, 32) ){
289               printf("Le nombre ne peut pas etre presenter en double precision\n");
290               goto B;
291             }
292
293             printf("La reprsentation de %s en virgule flottante (double precision)
294 est :\n");
295                         V_flottant64(num);
296                                 printf("\n");
297             }
298
299     } else {printf("le choix n'est pas valide\n");
300     goto A;
301             }
302             //Quitter ou Ressayer
```

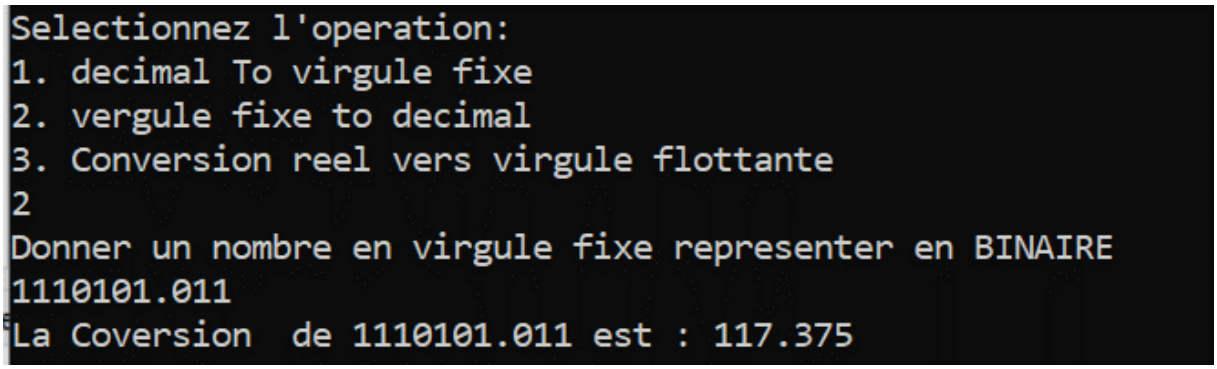
```
303  while (q != 0) {
         S: printf("Pour RESSAYER tapez R !, Pour QUITTER tapez Q !\n");
         scanf(" %c", &q);
         if (q == 'Q' || q == 'q') {
            printf("FIN DU PROGRAMME\n");
            return 0;
         } else if (q == 'R' || q == 'r') {
            goto A;
         } else {
            goto S;
         }
      }

      return 0;
   }
```

4. Captures d'écran de l'exécution (les tests)

1. Exemple pour 117.375





```
Selectionnez l'operation:
1. decimal To virgule fixe
2. vergule fixe to decimal
3. Conversion reel vers virgule flottante
3
Entrer un nombre REEL :
117.375
Donnez le nombre de bits (32 ou 64): 32
La representation de 117.375 en virgule flotante (simple precision) est :
0 10000101  11010101100000000000000
Pour RESSAYER tapez R !, Pour QUITTER tapez Q !
r
Selectionnez l'operation:
1. decimal To virgule fixe
2. vergule fixe to decimal
3. Conversion reel vers virgule flottante
3
Entrer un nombre REEL :
117.375
Donnez le nombre de bits (32 ou 64): 64
La representation de 117.375 en virgule flotante (double precision) est :
0 10000000101  1101010110000000000000000000000000000000000000000000000000000
Pour RESSAYER tapez R !, Pour QUITTER tapez Q !
```

2. Exemple pour -310.1175

```
Selectionnez l'operation:
1. decimal To virgule fixe
2. vergule fixe to decimal
3. Conversion reel vers virgule flottante
1
Entrer un nombre reel:
-310.1175
La representation de -310.1175 en virgule fixe est : 1 100110110.0001111000010100
Pour RESSAYER tapez R !, Pour QUITTER tapez Q !
```

```
Selectionnez l'operation:
1. decimal To virgule fixe
2. vergule fixe to decimal
3. Conversion reel vers virgule flottante
2
Donner un nombre en virgule fixe representer en BINAIRE
100110110.0001111000010100
La Coversion  de 100110110.0001111000010100 est : 310.1174926757812500
Pour RESSAYER tapez R !, Pour QUITTER tapez Q !
```

```
Selectionnez l'operation:
1. decimal To virgule fixe
2. vergule fixe to decimal
3. Conversion reel vers virgule flottante
3
Entrer un nombre REEL :
-310.1175
Donnez le nombre de bits (32 ou 64): 32
La representation de -310.1175 en virgule flottante (simple precision) est :
1 10000111  00110110000111100001010
```

3. Exemple pour 1212551287.25227

Ce nombre ne peut pas être représenté en simple précision mais peut-être représenter en double précision.

```
Selectionnez l'operation:
1. decimal To virgule fixe
2. vergule fixe to decimal
3. Conversion reel vers virgule flottante
3
Entrer un nombre REEL :
1212551287.25227
Donnez le nombre de bits (32 ou 64): 32
Le nombre ne peut pas etre presenter en simple precision
Entrer un nombre REEL :
1212551287.25227
Donnez le nombre de bits (32 ou 64): 64
La representation de 1212551287.25227 en virgule flottante (double precision) est :
0 10000011101  0010000100011000010000100000000000000000000000000000
Pour RESSAYER tapez R !, Pour QUITTER tapez Q !
```