Rapport du projet :

Conversion d'un nombre d'une base à une autre base (entre 2 et 16)

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

Encadre par: Mr. **BENALLA Hicham**

Introduction :

La conversion entre différentes bases numériques est une opération courante dans les domaines de l'informatique et des mathématiques. Elle permet de représenter les nombres dans différents formats. Ce rapport présente un projet qui aborde le défi de la conversion des nombres d'une base vers une autre base quelconque (les bases se situe entre 2 et 16).

Analyse du Problème

Le projet vise à résoudre le problème de la conversion de nombres d'une base A ,vers une autre base, notée B. La conversion entre ces bases peut être complexe, surtout lorsqu'elle implique des chiffres hexadécimaux allant de 0 à 9 et de A à F .

Solutions Proposées :

On dispose à rendre la base 10 comme la base intermédiaire , donc on va convertit le nombre de la base de depart (A) vers la base 10 et de la base 10 vers la base voulue(B) , voici l'algorithme a suivre pour elaborer ce genre de solution :

Plan d'algorithme :

Entrée :

L'utilisateur est invité à entrer un nombre, la base de départ (baseA), et la base de destination (baseB).

Validation :

L'algorithme vérifie si les bases se situent entre 2 et 16 (inclus). Si ce n'est pas le cas, un message d'erreur est affiché et le programme se termine.

Conversion de la base A vers la base 10 :

On parcourt le nombre de droite à gauche. Pour chaque chiffre, on le convertit en décimal. Si un chiffre est invalide pour la base A, un message d'erreur est affiché et le programme se termine. Le chiffre converti est ajouté au résultat après avoir été multiplié par la puissance appropriée de la base A. Cette étape permet de convertir le nombre de la base A en base 10.

Conversion de la Base 10 vers la Base B :

Le résultat de la conversion précédente est maintenant converti de la base 10 vers la base B. On extrait les chiffres du résultat en utilisant une division et en prenant les restes successifs. Chaque chiffre obtenu est ensuite converti en caractère hexadécimal.

Les caractères hexadécimaux sont stockés dans une séquence dans l'ordre inversé (du chiffre le plus significatif au moins significatif). Cette séquence représente le nombre dans sa forme en base B.

Affichage du Résultat :

Si la séquence est vide (cas spécial pour le nombre 0), on affiche simplement “0".Sinon, les chiffres hexadécimaux sont affichés dans le bon ordre. Cet algorithme permet de convertir un nombre d'une base A à une base B en passant par une conversion intermédiaire en base 10. Il vérifie également les entrées pour s'assurer que les bases se situent dans la plage appropriée.

Le programme traduit en langage C :

1. **Fonction pour convertir un caractère hexadécimal en décimal :**

```c
int hexaCaractereEnDecimal(char c) {
    switch (c) {
        case '0': return 0;
        case '1': return 1;
        case '2': return 2;
        case '3': return 3;
        case '4': return 4;
        case '5': return 5;
        case '6': return 6;
        case '7': return 7;
        case '8': return 8;
        case '9': return 9;
        case 'A': case 'a': return 10;
        case 'B': case 'b': return 11;
        case 'C': case 'c': return 12;
        case 'D': case 'd': return 13;
        case 'E': case 'e': return 14;
        case 'F': case 'f': return 15;
        default:
            printf("Caractère invalide : %c\n", c);
            exit(1);
    }
}
```

La fonction *hexaCaractereEnDecimal(char c)* prend un caractère (c) en tant qu'argument et renvoie la valeur décimale correspondante pour le caractère hexadécimal donné. Si le caractère n'est pas un chiffre hexadécimal valide, la fonction générera une erreur.

2. **Fonction pour convertir un caractère décimal en hexadécimal :**

```
char decimalEnHexaCaractere(int n) {
    switch (n) {
        case 0: return '0';
        case 1: return '1';
        case 2: return '2';
        case 3: return '3';
        case 4: return '4';
        case 5: return '5';
        case 6: return '6';
        case 7: return '7';
        case 8: return '8';
        case 9: return '9';
        case 10: return 'A';
        case 11: return 'B';
        case 12: return 'C';
        case 13: return 'D';
        case 14: return 'E';
        case 15: return 'F';
        default:
            return '\0';
    }
}
```

La fonction *DecimalEnhexaCaractere (int n)* prend un entier (n) en tant qu'argument et renvoie la valeur hexadécimale correspondante pour le nombre donné. Si la valeur décimale est en dehors de l'intervalle valide (0 à 15), elle retourne un caractère nul.

3. Fonction pour convertir un nombre d'une base A vers la base 10 :

```c
int convertirEnDecimal(const char* nombre, int baseA) {
    int longueur = strlen(nombre);
    int i;
    int resultat = 0;
    int multiplicateur = 1;

    for ( i = longueur - 1; i >= 0; i--) {
        int chiffre = hexaCaractereEnDecimal(nombre[i]);
        if (chiffre < 0 || chiffre >= baseA) {
            printf("Caractere invalide dans le nombre %s pour la base %d.\n", nombre, baseA);
            exit(1);
        }
        resultat += chiffre * multiplicateur;
        multiplicateur *= baseA;
    }

    return resultat;
}
```

La fonction *convertirEnDecimal(const char* nombre, int baseA)* prend une chaîne de caractères (nombre) et une base de départ (baseA) comme argumets. Elle convertit le nombre de la base A en décimal. Si le nombre contient des caractères invalides pour la base de départ, elle génère une erreur.

4. **Fonction pour convertir un nombre de la base 10 vers la base B :**

```
void convertirDeDecimal(int nombreDecimal, int baseB) {
    char resultat[60];
    int k = 0;

    while (nombreDecimal > 0) {
        int reste = nombreDecimal % baseB;
        resultat[k] = decimalEnHexaCaractere(reste);
        k++;
        nombreDecimal /= baseB;
    }

    if (k == 0) {
        printf("0"); // Cas spécial pour le nombre 0
    } else {
        for (int i = k - 1; i >= 0; i--) {
            printf("%c", resultat[i]);
        }
    }

    printf("\n");
}
```

La fonction *convertirDeDecimal(long long nombreDecimal, int baseB)* prend un nombre décimal (nombreDecimal) et une base de destination (baseB) comme arguments. Elle convertit le nombre décimal en base B. Elle stocke les caractères résultants dans un tableau pour l'affichage .

5. **Fonction principale pour convertir de la base A à la base B :**

```
void convertirBase(char* nombre, int baseA, int baseB) {
    int nombreDecimal = convertirEnDecimal(nombre, baseA);

    printf("Le nombre %s en base %d est equivalent a ", nombre, baseA);
    convertirDeDecimal(nombreDecimal, baseB);
}
```

La fonction *convertirBase(const char* nombre, int baseA, int baseB)* prend une chaîne de caractères (nombre), une base de départ (baseA), et une base de destination (baseB) comme arguments. Elle combine les fonctions précédentes pour la conversion complète du nombre de la base A à la base B. Puis elle affiche le résultat final.

6.Fonction main :

```c
int main() {
    char nombre[65];
    int baseA, baseB;

    printf("Entrez le nombre : ");
    scanf("%s", nombre);

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

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

    if (baseA < 2 || baseA > 16 || baseB < 2 || baseB > 16) {
        printf("Les bases doivent etre comprises entre 2 et 16.\n");
        return 1;
    }

    convertirBase(nombre, baseA, baseB);

    return 0;
}
```

Cette fonction est la fonction principale du programme. Elle prend les entrées de l'utilisateur pour le nombre, la base de départ et la base de destination. Elle vérifie la validité des bases entrées. Elle faire appeler la fonction *convertirBase* pour effectuer la conversion et afficher le résultat.

Voici un exemple d'execution :

```
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129 }
```

```
C:\Users\user\Desktop\AOprojets\projet1\mainC\main.exe
Entrez le nombre : BC12
Entrez la base de depart (entre 2 et 16) : 16
Entrez la base de destination (entre 2 et 16) : 2
Le nombre BC12 en base 16 est equivalent a 1011110000010010

Process returned 0 (0x0)   execution time : 12.191 s
Press any key to continue.
```