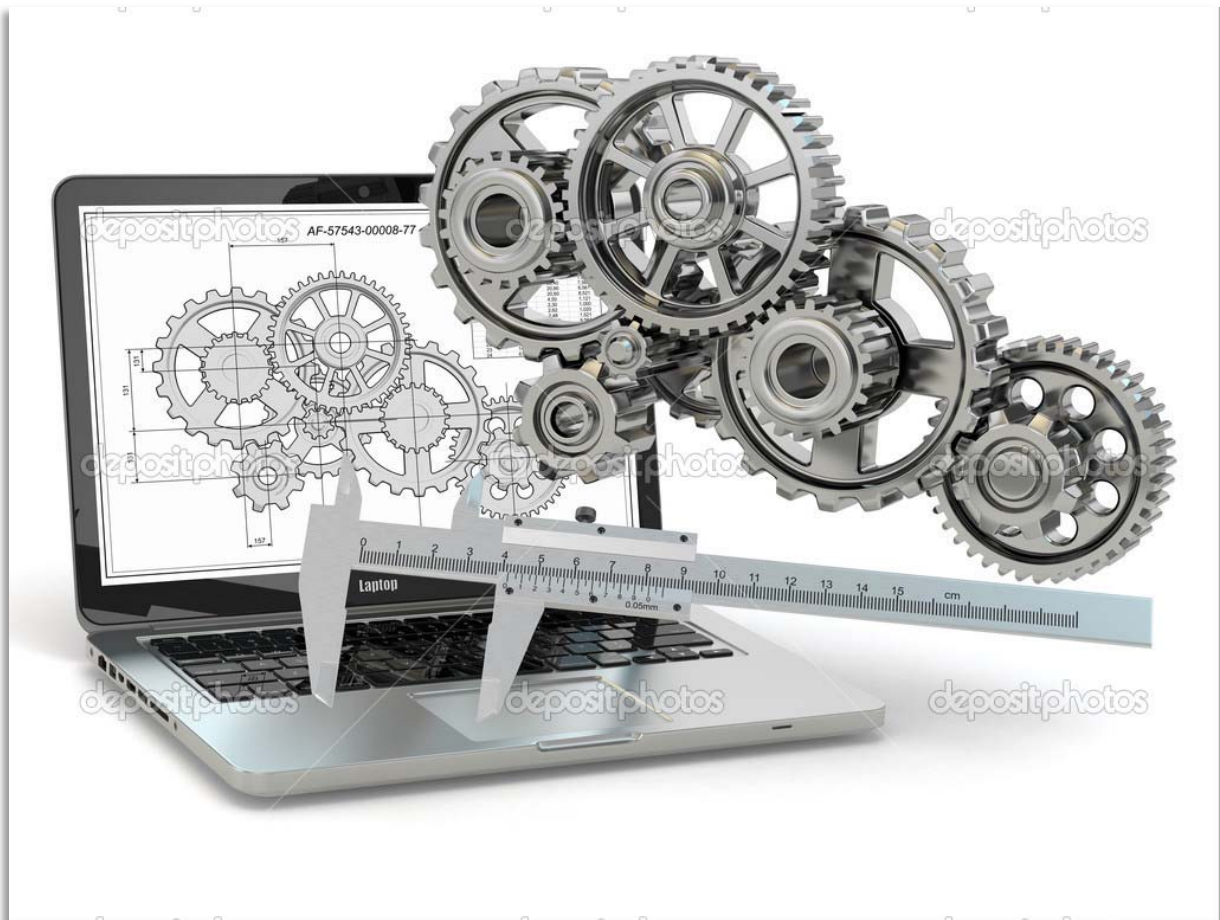


Convertisseur Numérique



(Une image originale représentant le projet)

Pittier Stéphane – Cin4B
ETML
Environ 110H
Chef de projet : Gilbert Gruaz
Expert 1 : Xavier Carrel
Expert 2 : Jean Zahn

Table des matières

1	SPÉCIFICATIONS.....	4
1.1	TITRE.....	4
1.2	DESCRIPTION	4
1.3	MATÉRIEL ET LOGICIELS À DISPOSITION	4
1.4	PRÉREQUIS.....	4
1.5	CAHIER DES CHARGES	4
2	PLANIFICATION INITIALE	5
3	ANALYSE.....	6
3.1	OPPORTUNITÉS.....	6
3.2	DOCUMENT D'ANALYSE ET CONCEPTION	6
3.2.1	<i>Analyse concurrentielle.....</i>	<i>6</i>
3.2.2	<i>Structure graphique.....</i>	<i>7</i>
3.2.3	<i>Maquette.....</i>	<i>9</i>
3.2.4	<i>Structogrammes.....</i>	<i>10</i>
3.2.5	<i>Développement.....</i>	<i>13</i>
3.3	CONCEPTION DES TESTS	14
3.4	PLANIFICATION DÉTAILLÉE.....	14
4	RÉALISATION.....	14
4.1	DOSSIER DE RÉALISATION	14
4.1.1	<i>Listes des outils.....</i>	<i>14</i>
4.1.2	<i>Programme.....</i>	<i>14</i>
4.1.3	<i>Nombre à virgule.....</i>	<i>14</i>
4.1.4	<i>Fonction Décimal -> BCD.....</i>	<i>15</i>
4.1.5	<i>Fonction Binaire -> GRAY.....</i>	<i>16</i>
4.1.6	<i>Fonction convertButton_Click.....</i>	<i>16</i>
4.1.7	<i>Fonction convertToAll.....</i>	<i>17</i>
4.1.8	<i>Méthode des Calculs.....</i>	<i>18</i>
4.1.9	<i>Initialisation des Opérations.....</i>	<i>18</i>
4.1.10	<i>Fonction addValue.....</i>	<i>18</i>
4.1.11	<i>Fonction subtractValue.....</i>	<i>19</i>
4.1.12	<i>Fonction multiplyValue.....</i>	<i>20</i>

4.1.13	Fonction d'affichage.....	20
4.2	MODIFICATIONS.....	20
4.2.1	Conversion de nombre à virgule.....	20
5	TESTS.....	21
5.1	DOSSIER DES TESTS	21
6	CONCLUSION.....	21
6.1	BILAN DES FONCTIONNALITÉS DEMANDÉES	21
6.2	BILAN DE LA PLANIFICATION	21
6.3	BILAN PERSONNEL.....	21
7	DIVERS.....	21
7.1	JOURNAL DE TRAVAIL.....	21
7.2	BIBLIOGRAPHIE.....	21
7.3	WEBOGRAPHIE	22
8	ANNEXES	22

1 SPÉCIFICATIONS

1.1 Titre

Réalisation d'une application de contrôle d'exercices en électronique numérique

1.2 Description

Implémenter une application qui va permettre aux utilisateurs (élèves ou enseignants), de contrôler les résultats des exercices de conversions de données numérique en binaire, octal, décimal, hexadécimal, BCD nombres réels à virgule flottante, ainsi que des opérations élémentaires comme des compléments à deux, addition et soustraction.

1.3 Matériel et logiciels à disposition

- 1 ordinateur standard ETML
- Visual Studio 2015
- Suite Office 2016

1.4 Prérequis

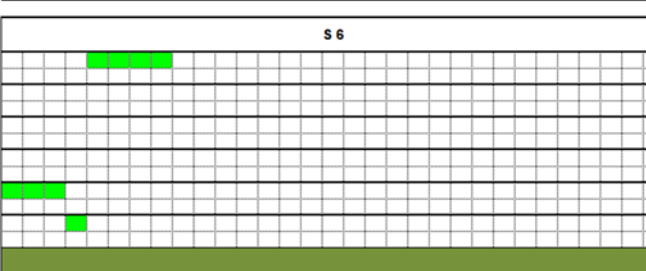
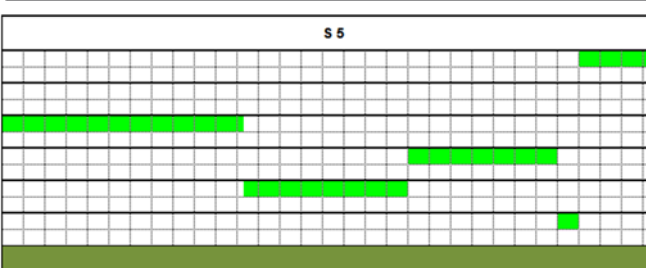
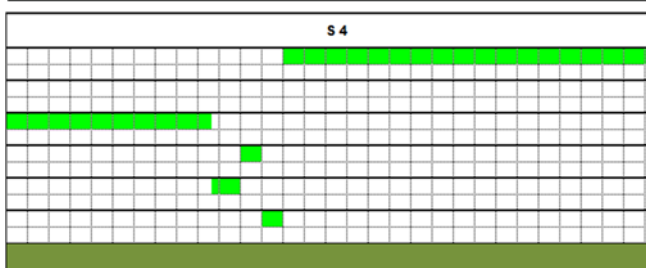
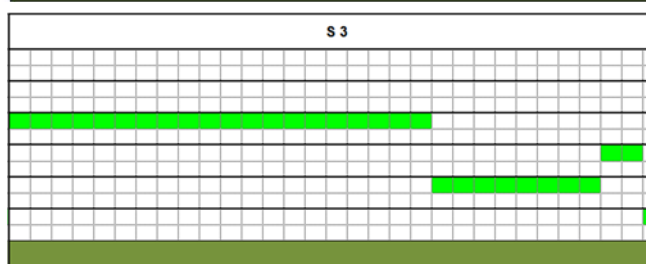
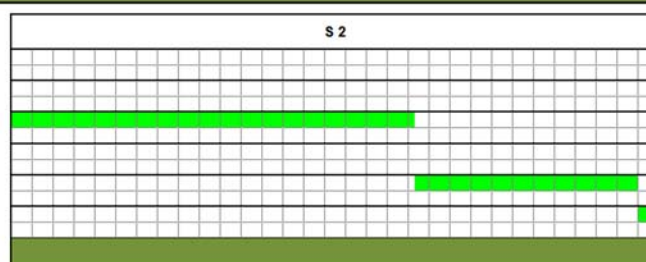
Avoir suivi les modules ICH à l'ETML, les projets et effectué des stages.

1.5 Cahier des charges

Le document fourni par le chef de projet fait foi. Il doit être mis en annexe → [Lien sur CDC](#)

2 PLANIFICATION INITIALE

Tâches - objectifs	Nb 1/4 heure	S 1
Absence - Imprévus	78	
	0	
Analyse	81	
	0	
Implémentation	181	
	0	
Tests	30	
	0	
Documentation	101	
	0	
Journal de travail	18	
	0	
Total planifié	489	
Total réalisé	0	



3 ANALYSE

3.1 Opportunités

A compléter

3.2 Document d'analyse et conception

Comme écrit dans les opportunités, le programme sera développé en C#. Il utilisera le type Windows Form Project, plus simple d'utilisation pour l'utilisateur que la console. Il sera aussi séparé en 2 fenêtre. Le convertisseur et le calculateur.

3.2.1 Analyse concurrentielle

Pour cette partie, j'ai trouvé deux sites permettant de faire presque pareil que mon application. Le premier contient plus de types que je veux, mais la conversion en ternaire et en quintal n'est pas une priorité. Pour le deuxième, il ne peut convertir que le décimal, hexadécimal et binaire.

La différence se trouve sur le fait qu'aucun des deux sites ne puissent convertir des valeurs à virgule, et que le premier ne puisse pas convertir une valeur négative.

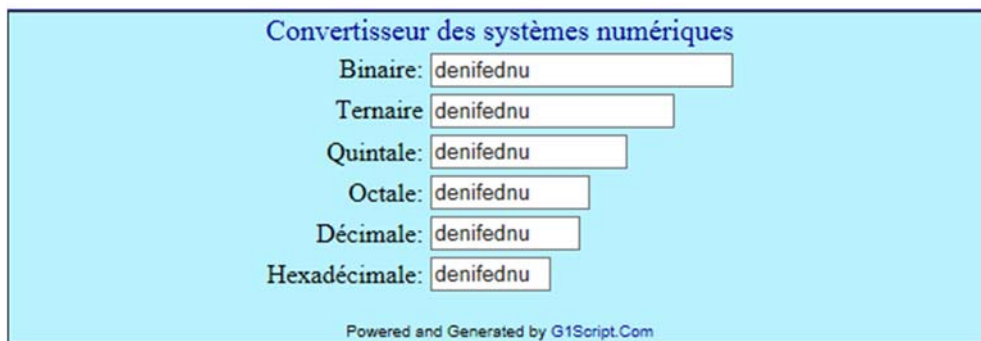


Figure 2 Exemple de conversion d'une valeur négative sur le site 1

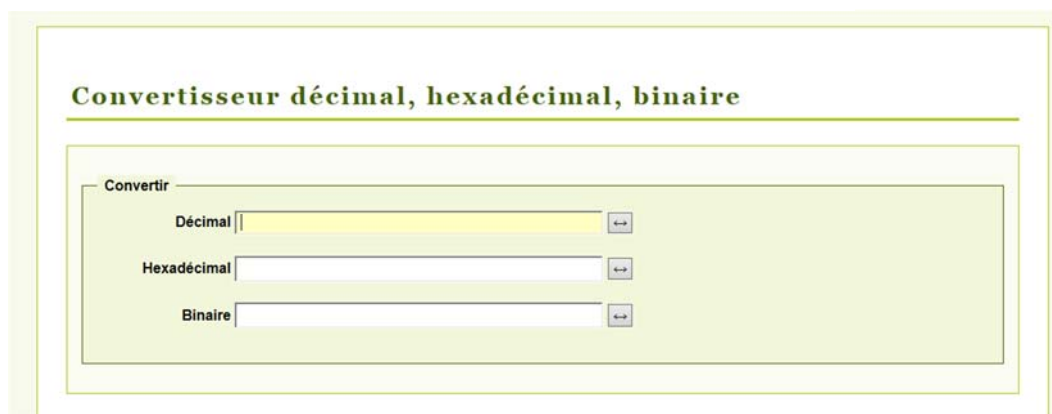
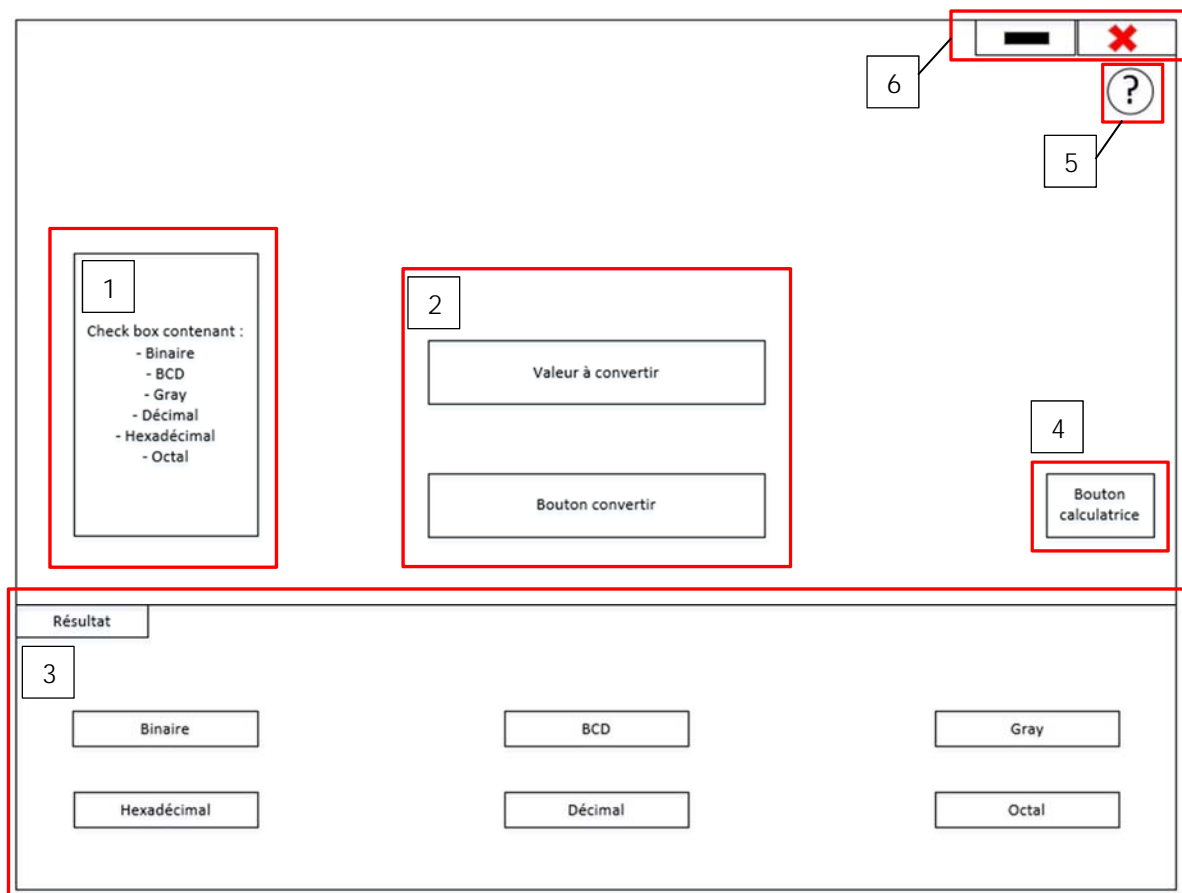


Figure 1 Page d'accueil du site 2

3.2.2 Structure graphique



Le diagramme illustre la structure graphique de l'interface utilisateur. Les éléments sont regroupés et numérotés comme suit :

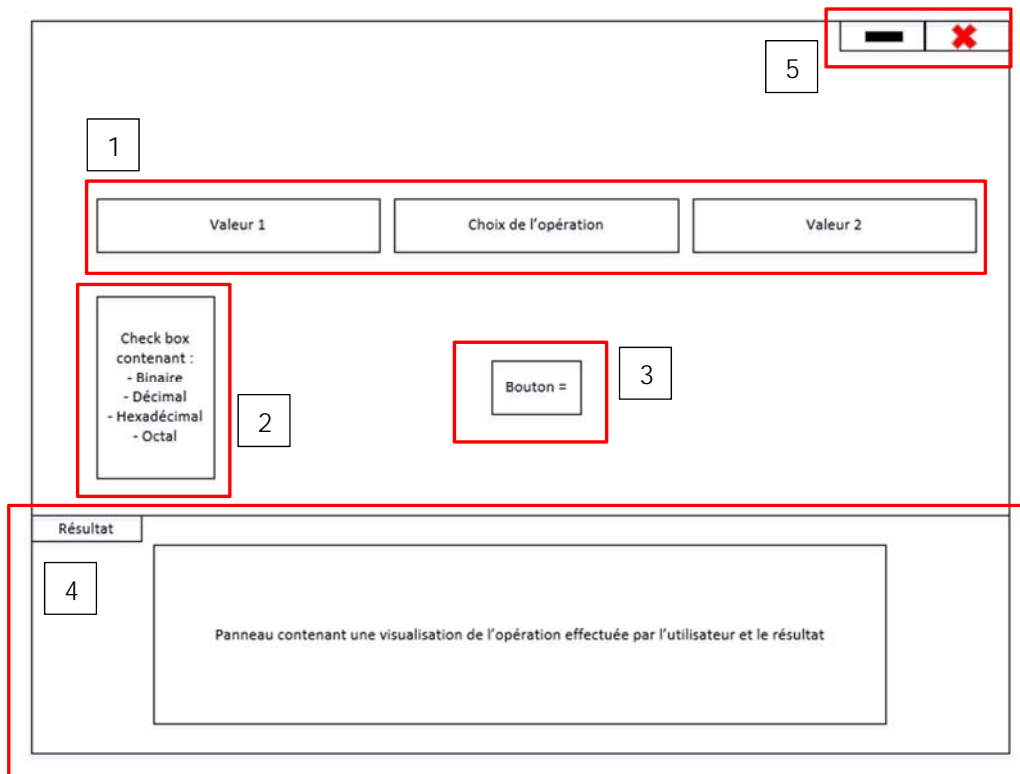
- 1** : Une zone contenant une liste de cases à cocher pour sélectionner la base de départ : Binaire, BCD, Gray, Décimal, Hexadécimal, Octal.
- 2** : Une zone contenant un champ de saisie pour la "Valeur à convertir" et un bouton "Bouton convertir".
- 3** : Une zone dédiée à l'affichage du "Résultat", comprenant six boutons pour sélectionner la base de destination : Binaire, Hexadécimal, BCD, Décimal, Gray, et Octal.
- 4** : Un bouton "Bouton calculatrice".
- 5** : Un bouton circulaire avec un point d'interrogation, servant à afficher une fenêtre d'opérations.
- 6** : Une barre d'état ou de menu en haut à droite, comportant un bouton avec une barre horizontale et un bouton avec une croix rouge.

La page principale du programme contient la partie convertisseur. Elle contient un nombre de petites informations diverses :

- 1) L'utilisateur doit choisir son type de base. Une série de check box seront à sa disposition pour informer le programme de quelle base de chiffre il doit partir pour effectuer la conversion.
- 2) Après avoir choisi son type, l'utilisateur doit entrer la valeur à convertir. Après avoir cliqué sur le bouton convertir, le programme va vérifier que toutes les informations sont correctes et lancer la méthode « Conversion ». Je décide de mettre ces options car dans le cas d'une recherche automatique du type, si l'utilisateur converti le nombre « 3654 » comment savoir s'il est question d'un nombre décimal, octal ou hexadécimal.
- 3) Une fois la conversion effectuée, le résultat va s'afficher. Pour gagner du temps, le résultat comprendra toutes les possibilités du programme (Binaire, BCD, Gray, Décimal, Octal et Hexadécimal).
- 4) Le bouton permettant d'afficher la fenêtre pour les opérations

- 5) Ce bouton permet d'afficher les informations relatives aux informations et au fonctionnement du logiciel
- 6) Bouton pour réduire et fermer le programme

La fenêtre pour les opérations ressemble beaucoup à celle de la conversion. Cette page s'ouvre à côté de l'autre et est indépendant de la conversion.

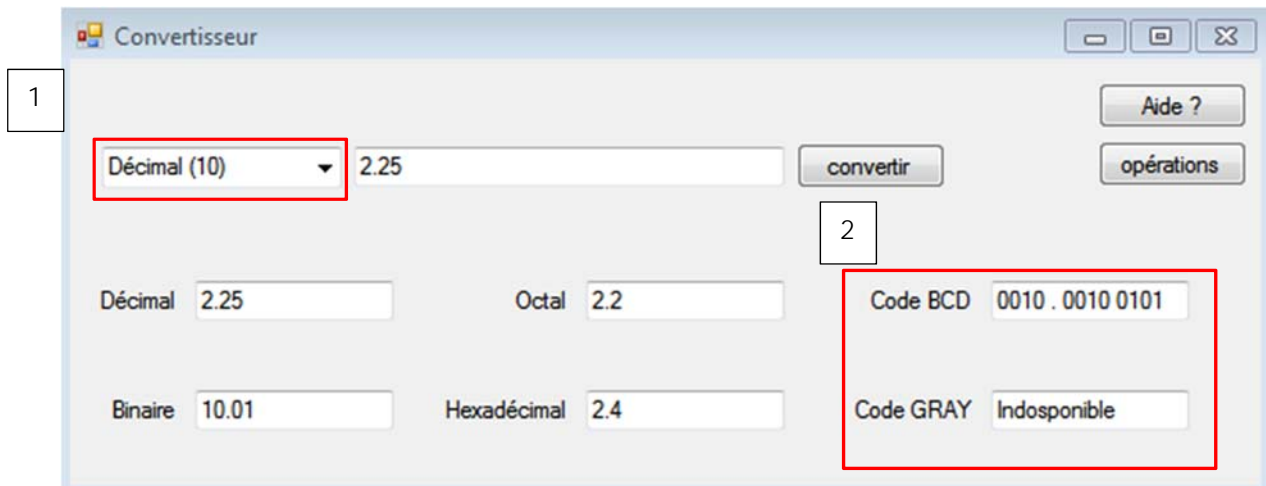


- 1) L'utilisateur entre les valeurs et choisit l'opération à effectuer (addition, soustraction et multiplication).
- 2) L'utilisateur choisit le type des valeurs qu'il a entré au point 1. Comme pour la fenêtre de conversion, le choix est indispensable pour éviter des problèmes dans le futur développement.
- 3) Une fois les valeurs entrées et le type choisi, l'utilisateur va cliquer sur le bouton « = ». Le programme va alors vérifier les valeurs et les envoyer dans la méthode « Calcul ».
- 4) Le panneau de résultat va afficher le résultat mais aussi un développement de l'opération pour que l'utilisateur puisse comprendre, de manière graphique, comment l'opération s'effectue.

- 5) Bouton pour réduire ou fermer la fenêtre de calcul. Etant donné qu'elle est indépendante par rapport à la conversion, si elle est fermée elle ne quitte pas le logiciel.

3.2.3 Maquette

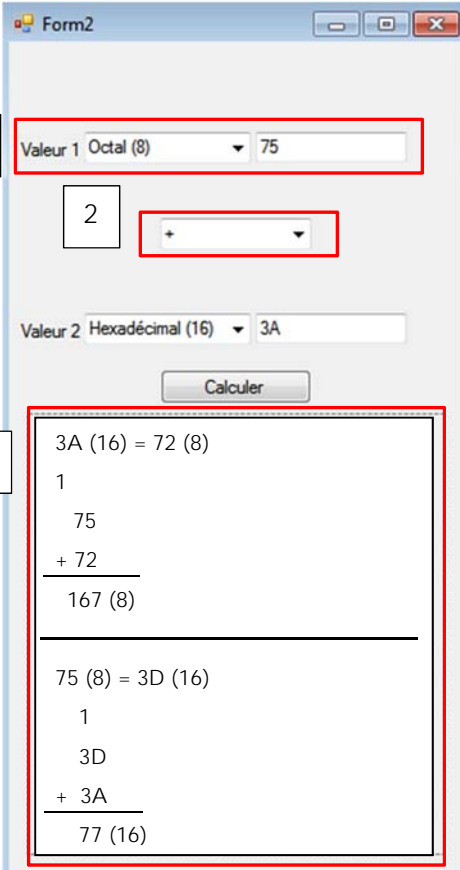
La maquette du site ressemble beaucoup à la version abordée au point 3.2.2. Cependant plusieurs modifications, discutée avec le chef de projet, viennent améliorer le programme.



Un remaniement de l'espace a été envisagé pour permettre une meilleure lisibilité.

- 1) Les checkboxes de la structure graphique, ont été remplacées par une liste déroulante plus pratique et plus petites à placer sur la fenêtre
- 2) Petite précision pour le code BCD et GRAY. Tous les formats peuvent être converti en code BCD et inversement.

Le code GRAY peut être converti uniquement en code BCD, et s'obtient uniquement à partir du BCD.



1

Valeur 1 Octal (8) 75

2

+

Valeur 2 Hexadécimal (16) 3A

Calculer

3

3A (16) = 72 (8)
 1
 75
 + 72

 167 (8)

75 (8) = 3D (16)
 1
 3D
 + 3A

 77 (16)

en colonne avec les retenues.

Pour la maquette de la calculatrice, certaines différences font aussi leurs apparitions.

1) Les 2 valeurs possèdent maintenant aussi une liste avec les types pour permettre les opérations avec des valeurs de types différentes.

Les types disponibles dans ces opérations sont le binaire, le décimal, l'octal et l'hexadécimal.

2) Les opérations sont aussi sous format de liste déroulante pour la même raison que les types des valeurs.

3) Pour les résultats étant donné qu'il faut une base commune pour effectuer les opérations, j'ai choisi de faire les 2 bases. La première version de l'opération sera toujours dans la base de la valeur 1. Et la deuxième dans la base de la valeur 2. Ensuite le programme prendra les deux valeurs pour effectuer l'opération demandée et afficher le détail sous forme d'opération

3.2.4 Structogrammes

Le programme sera décomposé en 3 parties majeurs. La conversion, le calcul et l'affichage.

On peut aussi séparer les grandes méthodes en plus petites mais faire un structogramme pour ces étapes intermédiaires ne servirait pas à grand-chose.

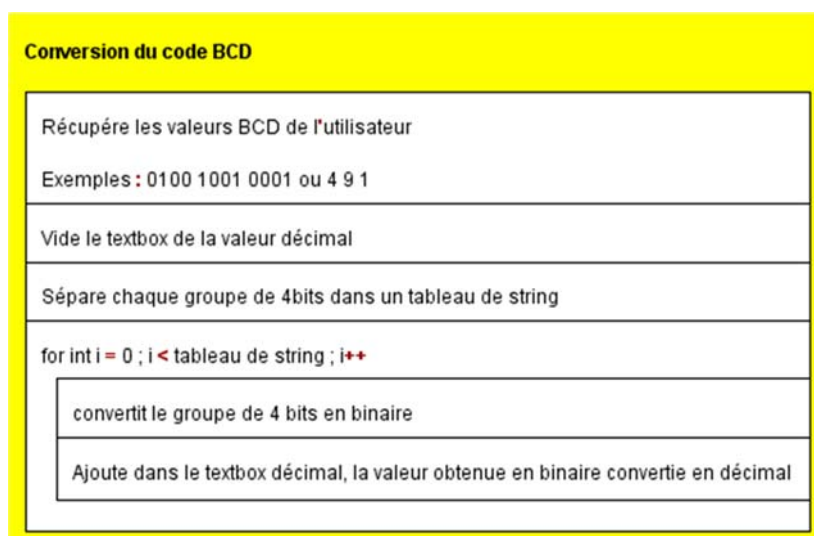
1) La méthode Conversion



Ce structogramme va me servir pour la conversion binaire, décimal, octal et hexadécimal. Les fonctions fournies par c# suffiront amplement à remplir ce travail.

2) BCD et GRAY

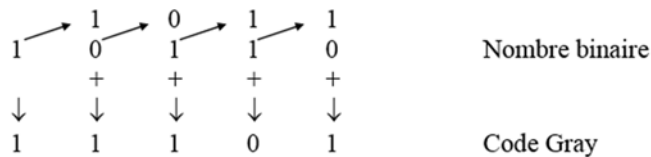
Pour le BCD et le code gray les choses se compliquent un peu. Pour le code BCD il faut récupérer les valeurs de chaque « segments » de 4 bits afin de les convertir dans les bons formats.



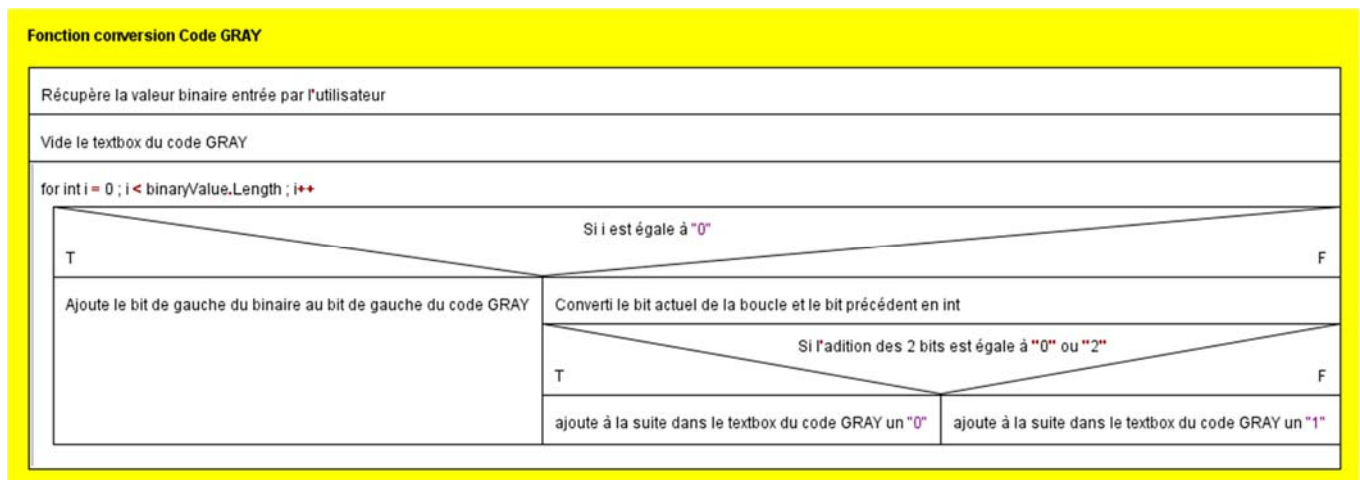
Ce qui suit la méthode présentée dans le document d'ELEOC donné aux élèves.

3	4	7 ₁₀
↓	↓	↓
0011	0100	0111 _{BCD}

Pour le code GRAY, Je garde aussi le principe donné en cours d'ELEOC. Je convertis la valeur entrée par l'utilisateur en binaire. Ensuite je fais une addition logique entre le bit actuel et le bit précédent. Ce qui suis le schéma suivant.

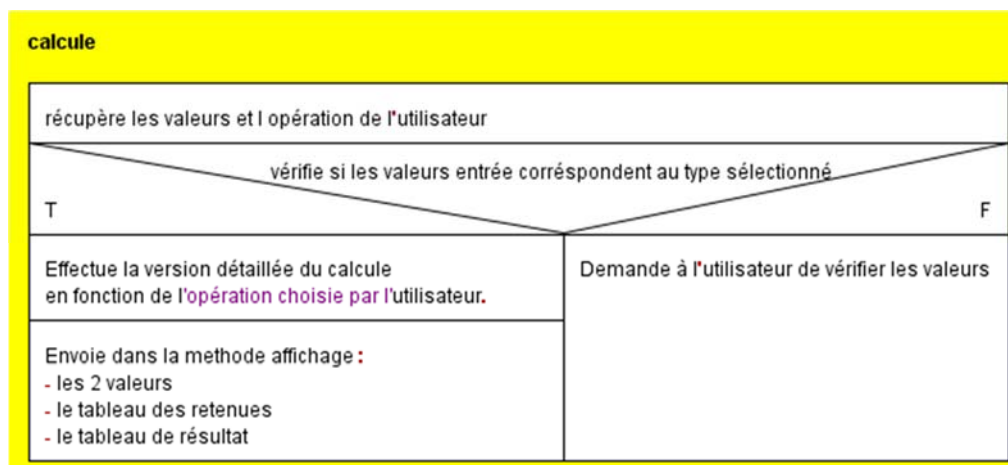


Ce qui donne le structogramme suivant pour la fonction binaireToGray

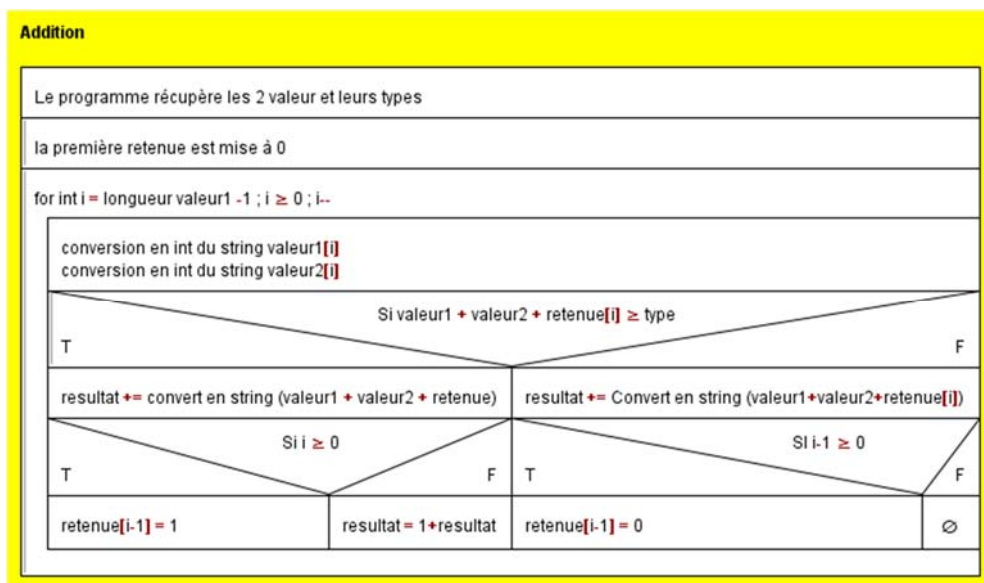


3) La méthode Calcul

Comme marqué dans le chapitre 4.2.1, j'ai complètement remanié le système de calcul et d'affichage.

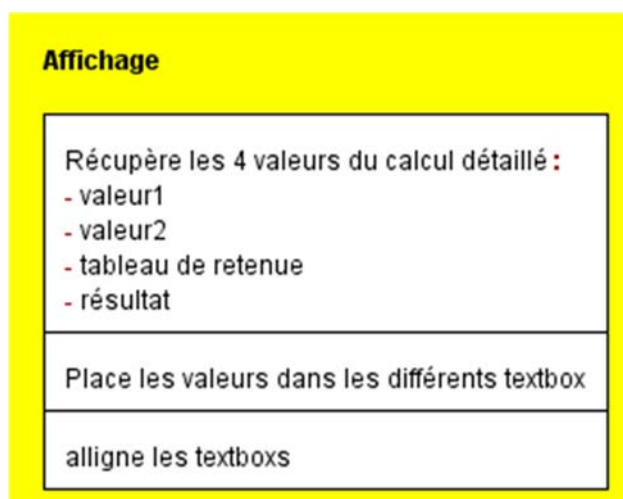


Pour montrer comment fonctionne la version détaillée du calcul, voici le structogramme de la fonction addition.



Chaque opération fonctionnera sur le même système.

4) La méthode Affichage



Pour la méthode d'affichage, je vais me baser sur le document utilisé en ELEOC. Il arrive à montrer de manière précise comment sont effectuées les opérations.

Pour le placement se sera sûrement des textbox qui seront utilisées. Mais si une meilleure alternative est trouvée, il en sera fait mention dans le chapitre **NOCHAPITRE**.

3.2.5 Développement

La plus grosse partie de ce TPI réside dans la partie développement du programme. Pour ce faire, je vais utiliser l'ide Visual Studio 2015. Il sera développé en langage c# et suivra les normes de l'ETML.

3.3 Conception des tests

Les tests se dérouleront en 2 parties :

- La grille de tests
- Les tests unitaires

Pour la grille elle comportera une série de manipulations, bonnes ou mauvaises, pour vérifier le bon fonctionnement

3.4 Planification détaillée

- A ce stade, après l'analyse complète du projet, un planning détaillé et complet (avec tâches, sous-tâches, dépendances, durée, ...) peut être finalisé.
- Le planning détaillé doit s'inscrire dans le planning initial. Il faut que l'on puisse situer cette planification détaillée par rapport à la planification initiale.

4 RÉALISATION

4.1 Dossier de Réalisation

4.1.1 Listes des outils

- Visual Studio 2015
- ConventionsDeCodageV2.3.3 ETML

4.1.2 Programme

Les fichiers contenant le code sources sont disponible sur le git ou dans les annexes. Dans cette partie je vais reprendre les méthodes les plus intéressantes et les décrire dans le détail.

4.1.3 Nombre à virgule

Pour le moment les nombres à virgule flottante ne fonctionne pas. La fonction de base de c# ne supporte pas les variables de format « Float » ou « Double ».

Cependant une alternative à l'aide d'un tableau de bits est envisageable. Cela rendrait Le développement un peu plus long. En effectuant des recherches, j'ai trouvé une méthode qui permet de passer d'un nombre à virgule décimal au tableau de bits, mais impossible de le transformer en un autre format.

Une fois l'affichage terminé, je me pencherais plus en profondeur sur ce problème.

4.1.4 Fonction Décimal -> BCD

La fonction pour convertir un nombre décimal en un nombre en code BCD me semble un bon exemple pour commencer.

```
102     /// <summary>
103     /// Permet de convertir la valeur de l'utilisateur en code BCD
104     /// </summary>
105     /// <param name="decimalValue">Valeur décimale de la conversion en cours</param>
106     private void convertToBcd(string decimalValue)
107     {
108         bcdTextBox.Text = "";
109
110         //Separe la valeur en unité pour effectuer la conversion
111         for (int i = 0; i < decimalValue.Length; i++)
112         {
113             string temp = Convert.ToString(decimalValue[i]);
114             int value = Convert.ToInt32(temp);
115             temp = Convert.ToString(value, 2);
116
117             //Tant que le chiffre n'est pas sur 4 bits rajoute un 0 devant
118             if (temp.Length != 4)
119             {
120                 int maxZero = temp.Length;
121                 for (int y = 0; y < 4 - maxZero; y++)
122                 {
123                     temp = "0" + temp;
124                 }
125             }
126
127             bcdTextBox.Text += temp + " ";
128         }
129     }
130
```

La valeur entrée par l'utilisateur, est envoyée dans la méthode sous forme de tableau « String ». La première boucle commence, elle a pour effet de prendre chaque chiffre de la valeur pour ensuite la convertir en binaire. Une fois la conversion en binaire effectuée, cette nouvelle valeur est stockée dans une variable temporaire (appelé « temp »). Le code BCD nécessite 4 bit et dépendant de la valeur il se peut que ce maximum ne soit pas atteint. C'est alors que la deuxième boucle entre en jeu. Son principe est de calculer la longueur de notre string et de rajoute le bon nombre de 0 devant pour arriver à quatre bits. Une fois ceci effectué, la variable temp est placée dans la textbox de la valeur BCD.

4.1.5 Fonction Binaire -> GRAY

```
148 /// <summary>  
149 /// Permet de convertir le binaire en code Gray  
150 /// Basé sur : http://stackoverflow.com/questions/1691074/gray-code-in-net  
151 /// <param name="binaryValue">Valeur Binaire de la conversion en cours</param>  
152 /// </summary>  
153 private void convertToGray(string binaryValue)  
154 {  
155     grayTextBox.Text = "";  
156     for (int i = 0; i < binaryValue.Length; i++)  
157     {  
158         if (i == 0)  
159             grayTextBox.Text += binaryValue[0];  
160         else  
161         {  
162             //Récupere chaque bit et effectue le complément à 2 avec le bit d'avant  
163             string val1 = Convert.ToString(binaryValue[i]);  
164             string val2 = Convert.ToString(binaryValue[i - 1]);  
165  
166             int test1 = Convert.ToInt32(val1);  
167             int test2 = Convert.ToInt32(val2);  
168  
169             if (test1 + test2 == 0 || test1 + test2 == 2)  
170                 grayTextBox.Text += "0";  
171             else  
172                 grayTextBox.Text += "1";  
173         }  
174     }  
175 }  
176
```

Pour cette fonction, le principe suit le même cheminement que pour le « Décimal -> BCD ». Une valeur en tableau « String » est envoyée dans la fonction. Elle va ensuite prendre la longueur de ce tableau. Comme le bit de gauche est le même en binaire et en code gray, il va directement le placer dans la textbox. Pour les autres bits, il va sélectionner le bit actuel de la boucle plus celui à la position « i-1 », afin d'effectuer le complément à deux. Pour être sûre que le programme ne fasse pas d'erreur il va vérifier le résultat de ce complément et attribuer la valeur 1 ou 0 à la suite du textbox.

4.1.6 Fonction convertButton_Click

Avant de convertir dans les 6 types, le programme va préparer les informations.

```
41 /// <summary>  
42 /// Lance la conversion dans les 6 types disponibles  
43 /// </summary>  
44 /// <param name="sender"></param>  
45 /// <param name="e"></param>  
46 private void convertButton_Click(object sender, EventArgs e)  
47 {  
48     checkIntegrityValue();  
49     string exception = Convert.ToString(typesComboBox.SelectedItem);  
50     int baseNumber;
```

Il va tout d'abord récupérer la valeur de l'item sélectionné dans la liste déroulante et initialiser la variable baseNumber.


```

53         if (exception == "Code BCD")
54         {
55             convertBcdToDecimal(valueTextBox.Text);
56         }
57         else if (exception == "Code GRAY")
58         {
59             convertGrayToBinary(valueTextBox.Text);
60         }
61         else
62         {
63             char[] splitters = new char[] { '|' };
64             string test = Convert.ToString(typesComboBox.SelectedItem);
65             string[] basetype = test.Split(splitters);
66
67             baseNumber = Convert.ToInt32(basetype[1]);
68             convertToAll(valueTextBox.Text, baseNumber);
69         }
70     }

```

Ensuite, grâce à la variable « exception », le programme va définir la base de la valeur entrée par l'utilisateur. Si elle est égale à BCD ou GRAY, il va d'abord lancer la fonction de conversion adéquate.

```

67         baseNumber = Convert.ToInt32(basetype[1]);
68         convertToAll(valueTextBox.Text, baseNumber);
69     }
70 }

```

Sinon il va séparer la valeur au niveau du | pour ne garder que le chiffre de la base. Pour l'exemple, si l'utilisateur choisi le type « Binaire | 2 » Le programme donnera la valeur 2 à la variable userBase.

Par la suite, il lancera la fonction « convertToAll » avec comme argument :

- La valeur entrée par l'utilisateur dans le textbox
- La base de la valeur choisie par l'utilisateur

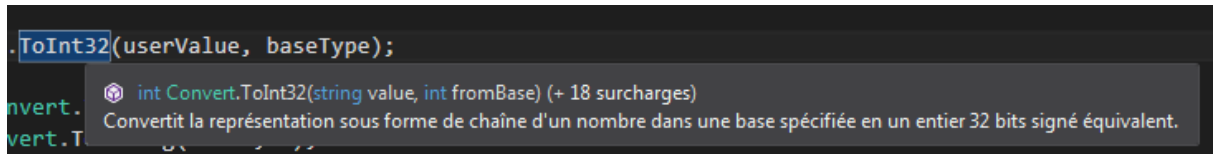
4.1.7 Fonction convertToAll

```

81     /// <summary>
82     /// Permet de convertir dans les format du Binaire | Décimal | Octal | Hexadécimal
83     /// </summary>
84     /// <param name="userValue">Valeur entrée par l'utilisateur dans le textbox</param>
85     /// <param name="baseType">base de la valeur obtenue avec le split dans la fonction convertButton_Click</param>
86     private void convertToAll(string userValue, int baseType)
87     {
88         int value = Convert.ToInt32(userValue, baseType);
89
90         binaryTextBox.Text = Convert.ToString(value, 2);
91         octalTextBox.Text = Convert.ToString(value, 8);
92         decimalTextBox.Text = Convert.ToString(value, 10);
93         hexaTextBox.Text = Convert.ToString(value, 16);
94
95         convertToBcd(decimalTextBox.Text);
96         convertToGray(binaryTextBox.Text);
97     }

```

Pour cette fonction, le principe est simple. Les valeurs envoyées par la fonction « Convert_click » sont ensuite transposées dans toutes les bases. Pour ce faire, j'utilise la fonction de conversion de type proposée par c#.



Lorsque que l'on effectue une conversion en type INT ou STRING, un argument permet de venir insérer une valeur (2,8,10 ou 16) pour choisir la base de notre base.

Une fois ces opérations effectuée, le programme lance les fonction de conversion en BCD et en GRAY

4.1.8 Méthode des Calcules

La méthode pour effectuer les opérations, se situe dans la Form Calculate. Elle se compose de quatre méthodes principales :

- Addition
- Soustraction
- Multiplication
- Division

Le fait de la séparer en quatre petites fonctions, permet de mieux contrôler les spécificités de chaque opération arithmétique. Cependant cela rajoute des lignes de codes. Pour les opérations avec des valeurs de bases différents, les fonctions retourneront le résultat avec la valeur 2 convertie avec la base de la valeur 1.

4.1.9 Initialisation des Opérations

Au moment où l'utilisateur va activer le bouton opération, le programme va initialiser tous les éléments. Cette initialisation va prendre place dans 2 fonction :

- calculateButton_Click
- calcualteValues

Un switch est mis en place pour faciliter l'aiguillage entres les diverses fonctions de calcules.

4.1.10Fonction addValue

Le principe restera pour toutes les fonctions de calculs. Voici le fonctionnement de cette fonction.

```

186 // <summary>
187 // Fonction permettant d'effectuer une addition en colonne entre 2 valeurs de meme base entrée par l'utilisateur.
188 // </summary>
189 // <param name="value1">Valeur 1 de l'utilisateur convertie dans son format</param>
190 // <param name="value2">Valeur 2 de l'utilisateur convertie dans son format</param>
191 // <param name="baseValue">Prends la base de la valeur 1 ou 2 pour afficher le résultat en fonction</param>
192 // <param name="result">Stock et permet l'affichage du résultat</param>
193 // <param name="retenue">Tableau de int permettant de stocker les retenues lors de l'addition en colonne. Est aussi utilisée lors de l'affichage</param>
194 private void addValue(string value1, string value2, int baseValue, string result, int[] retenue)
195 {
196     retenue[value1.Length-1] = 0;
197 }

```

Le programme va prendre chaque unité des valeurs, les unes après les autres.

```

128 //Permet de verifier les valeurs et ainsi de placer les retenues adéquates
129 if (intValue1Unit + intValue2Unit + retenue[i] >= baseValue)
130 {
131     result = Convert.ToString(intValue1Unit + intValue2Unit + retenue[i] - baseValue) + result;
132     if (i - 1 >= 0)
133     {
134         retenue[i - 1] = 1;
135     }
136     else
137     {
138         result = "1" + result;
139     }
140 }
141 //sinon il va juste effectuer l'opération et place une retenue à 0 pour la suite du calcul
142 else
143 {
144     result = Convert.ToString(intValue1Unit + intValue2Unit + retenue[i]) + result;
145     if (i - 1 >= 0)
146     {
147         retenue[i - 1] = 0;
148     }
149 }
150 }
151 }

```

Ensuite, il va effectuer une addition entre le chiffre 1 et le chiffre 2. Si la valeur obtenue est plus grande que la base des 2 valeurs, le programme va placer une retenue de 1 dans le tableau. Pour le résultat, si l'addition est plus grande que la base, il va retirer la base et placer cette valeur dans la variable de résultat.

Si la valeur obtenue est moins grande que la base, il va placer un 0 dans le tableau des retenues et placer la valeur dans la variable de résultat. Mais c'est la dernière opération, et que la retenue actuelle vaut 1, il va la placer en première position du string résultat.

4.1.11 Fonction *subtractValue*

Pour la fonction de soustraction, le principe reste le même que l'addition. Une fois les valeurs transformées et envoyées dans la fonction, le programme va soustraire à chaque unité. Si le résultat est plus petit que 0, le programme va retirer 1 à l'unité suivante et ajouter 10 à la retenue du calcul actuelle. La valeur calculée via : $v1 + \text{retenue} - v2$ est placée dans un string, qui sera affiché par la suite. **A COMPLETER AVEC DES SCREENS**

4.1.12 Fonction *multiplyValue*

Pour cette fonction, le principal problème a été de séparer les retenues du nombre à ajouter au résultat. Car si dans l'addition, une retenue est toujours égale à 1 ou 0, là il se peut que la retenue vaille 9. Pour contrer ce problème j'ai mis en place un système de tableau à 2 dimension.

Dans le cas d'une valeur à 2 chiffre, l'index 1 du tableau est consacré aux retenues du premier chiffre et l'index 0 aux retenues du deuxième. Les résultats sont stockés dans un tableau de string.

Pour alléger le code, j'ai mis en place le 31.05.2017, 2 variables « factor ». Elles permettent de stocker les valeurs à multiplier et permet une meilleure lisibilité du code.

Une fois les multiplications faites, une additionne tous les strings contenus dans le tableau, est cela nous donne le résultat final. **A compléter avec les sceens.**

4.1.13 Fonction d'affichage

Pour afficher les résultats, 4 textboxes sont nécessaire. Ils permettent de montrer de manière simple les retenues, les valeurs et le résultat.

Pour le code il suffit simplement de placer les bons paramètres dans les bon textboxes.

4.2 Modifications

4.2.1 Conversion de nombre à virgule

La conversion de nombre à virgule ne fonctionne pas. La méthode de base de c# ne supporte pas les valeurs de types « Float » ou « Double ».

Après quelques recherches je suis tombé sur un début de solution. Il consiste à transformer notre valeur en tableau de bits, et de convertir la valeur bits par bits. Cependant une erreur se produit lors de la conversion. Dès que j'essaie de convertir les bits, il revient sur une base 10.

Sinon une deuxième solution serait de passer par un parseInt. J'ai trouvé un convertisseur qui fonctionne avec cette méthode : <http://coderstoolbox.net/number/>

5 TESTS

5.1 Dossier des tests

- On dresse le bilan des tests effectués (qui, quand, avec quelles données...) sous forme de procédure. Lorsque cela est possible, fournir un tableau des tests effectués avec les résultats **attendus et obtenus**, ainsi que les actions à entreprendre en conséquence (et une estimation de leur durée).
- Si des tests prévus dans la stratégie n'ont pas pu être effectués :
 - raison, décisions, etc.
- Liste des bugs répertoriés avec la date de découverte et leur état:
 - Corrigé, date de correction, corrigé par, etc.

6 CONCLUSION

6.1 Bilan des fonctionnalités demandées

- Il s'agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.
- Si ce n'est pas le cas, estimer en « % » ou en « temps supplémentaire » le travail qu'il reste à accomplir pour terminer le tout.

6.2 Bilan de la planification

- Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet. Indiquer les différences entre les planifications initiales et détaillées avec le journal de travail.

6.3 Bilan personnel

- Si c'était à refaire:
 - Qu'est-ce qu'il faudrait garder ? Les plus et les moins ?
 - Qu'est-ce qu'il faudrait gérer, réaliser ou traiter différemment ?
- Qu'est que ce projet m'a appris ?
- Suite à donner, améliorations souhaitables, ...
- Remerciements, signature, etc.

7 DIVERS

7.1 Journal de travail

- Date, activité (description qui permet de reproduire le cheminement du projet), durée, liens et références sur des documents externes. Lorsqu'une activité de recherches a été entreprise, il convient d'énumérer ce qui a été trouvé, avec les références.

7.2 Bibliographie

- Références des livres, revues et publications utilisés durant le projet.

7.3 Webographie

- Analyse concurrentielle
 - www.aly-abbara.com/utilitaires/convertisseur/convertisseur_chiffres.html
 - www.sebastienguillon.com/test/javascript/convertisseur.html
- Convertisseur avec nombre à virgule
 - <http://coderstoolbox.net/number/>
- Documentations c#
 - <https://docs.microsoft.com/en-us/dotnet/csharp/>
 - <https://msdn.microsoft.com/fr-fr/library>
-

8 ANNEXES

- **Cahier des charges**
- Listing du code source (partiel ou, plus rarement complet)
- Guide(s) d'utilisation et/ou guide de l'administrateur
- Etat ou « dump » de la configuration des équipements (routeur, switch, robot, etc.).
- Extraits de catalogue, documentation de fabricant, etc.