
Convertisseur Numérique c#



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>11</i> |
| 3.2.5 | <i>Développement.....</i> | <i>14</i> |
| 3.3 | CONCEPTION DES TESTS | 14 |
| 3.4 | PLANIFICATION DÉTAILLÉE..... | 14 |
| 4 | RÉALISATION..... | 15 |
| 4.1 | DOSSIER DE RÉALISATION | 15 |
| 4.1.1 | <i>Listes des outils</i> | <i>15</i> |
| 4.1.2 | <i>Programme</i> | <i>15</i> |
| 4.1.3 | <i>Nombre à virgule.....</i> | <i>15</i> |
| 4.1.4 | <i>Fonction Décimal -> BCD.....</i> | <i>16</i> |
| 4.1.5 | <i>Fonction Binaire -> GRAY.....</i> | <i>17</i> |
| 4.1.6 | <i>Fonction convertButton_Click.....</i> | <i>17</i> |
| 4.1.7 | <i>Fonction convertToAll.....</i> | <i>19</i> |
| 4.1.8 | <i>Méthode des Calculs</i> | <i>19</i> |
| 4.1.9 | <i>Initialisation des Opérations.....</i> | <i>20</i> |
| 4.1.10 | <i>Fonction addValue.....</i> | <i>20</i> |
| 4.1.11 | <i>Fonction subtractValue.....</i> | <i>21</i> |

| | | |
|--------|---|----|
| 4.1.12 | Fonction <i>multiplyValue</i> | 22 |
| 4.1.13 | Fonction d'affichage..... | 23 |
| 4.2 | MODIFICATIONS..... | 23 |
| 4.2.1 | Conversion de nombre à virgule..... | 23 |
| 4.2.2 | Les retenues des opérations..... | 24 |
| 4.2.3 | Division détaillée..... | 24 |
| 4.2.4 | Améliorations possible | 24 |
| 5 | TESTS..... | 25 |
| 5.1 | GRILLE DE TESTS | 25 |
| 5.2 | DOSSIER DES TESTS | 26 |
| 5.2.1 | Bilan des tests..... | 26 |
| 5.2.2 | Découverte des bugs | 26 |
| 6 | CONCLUSION..... | 27 |
| 6.1 | BILAN DES FONCTIONNALITÉS DEMANDÉES | 27 |
| 6.2 | BILAN DE LA PLANIFICATION | 28 |
| 6.3 | BILAN PERSONNEL..... | 29 |
| 7 | DIVERS..... | 31 |
| 7.1 | JOURNAL DE TRAVAIL..... | 31 |
| 7.2 | WEBOGRAPHIE | 31 |
| 8 | ANNEXES | 32 |
| 8.1 | CAHIER DES CHARGES..... | 32 |
| 8.2 | MANUEL UTILISATEUR | 34 |
| 8.3 | CODE SOURCE | 38 |
| 8.4 | PLANIFICATION DÉTAILLÉE..... | 39 |
| 8.5 | JOURNAL DE TRAVAIL..... | 40 |

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

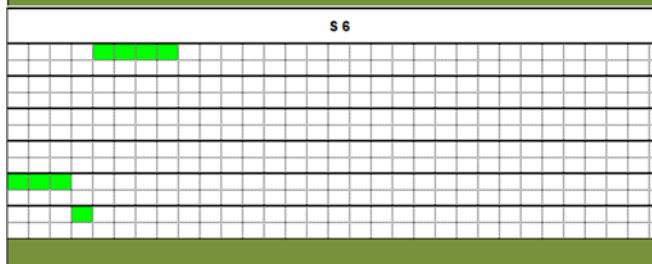
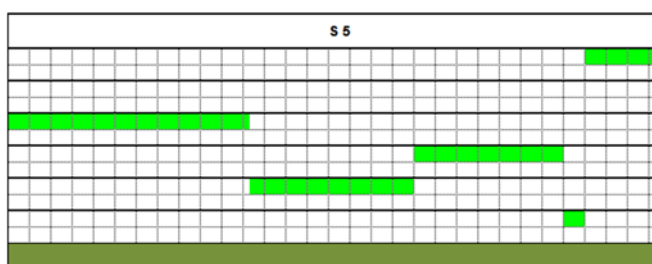
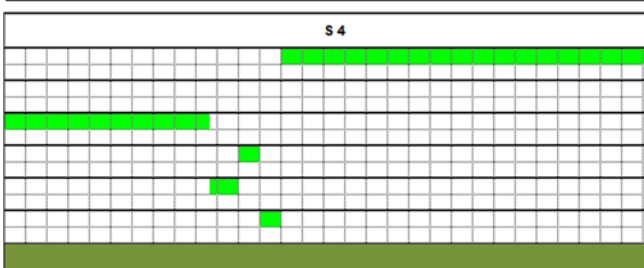
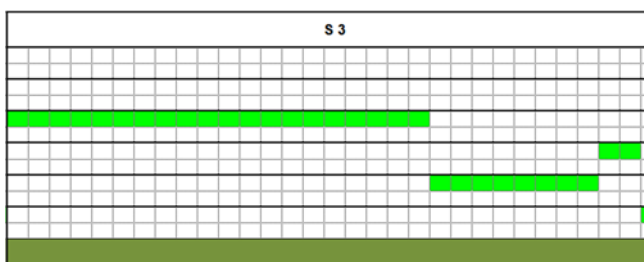
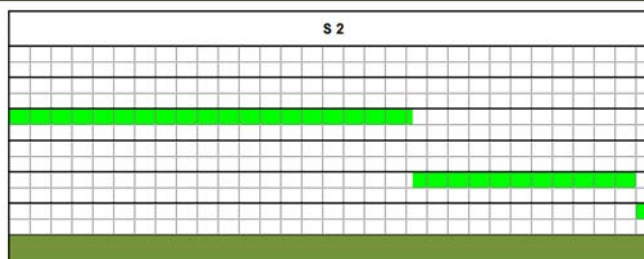
Avoir suivi les modules ICH à l'ETML, les projets et effectué des stages.

1.5 Cahier des charges

[Lien vers le cahier des charges \(Annexe 8.1\)](#)

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

Ce projet va me permettre de m'améliorer dans les points suivant :

- Code c#
- Gestion de projet
- Optimisation de code

C'est aussi un bon défi pour voir si j'arrive à gérer le stress lié à un projet sur six semaines.

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.

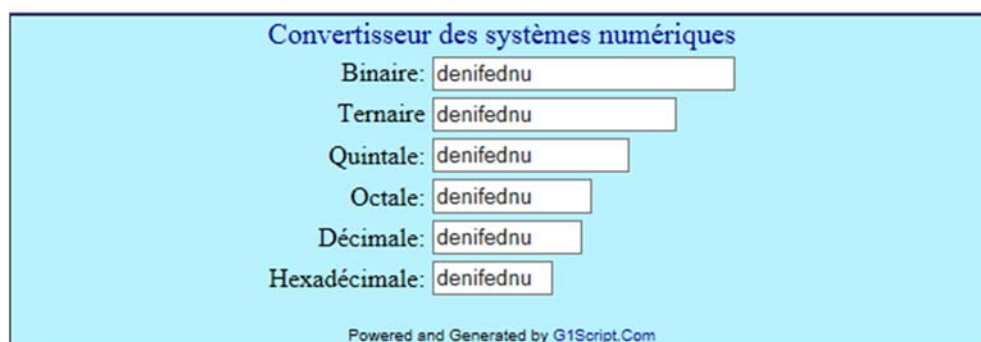


Figure 0.1 Exemple de conversion d'une valeur négative sur le site 1

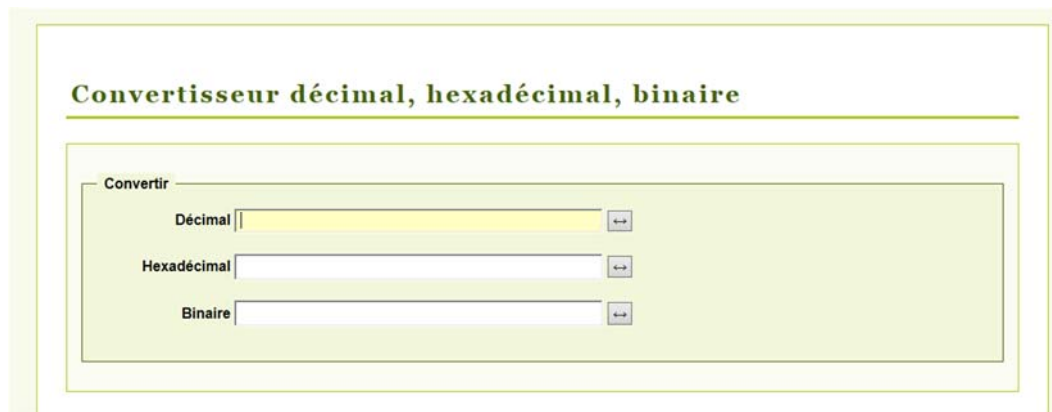


Figure 0.2 Page d'accueil du site 2

3.2.2 Structure graphique

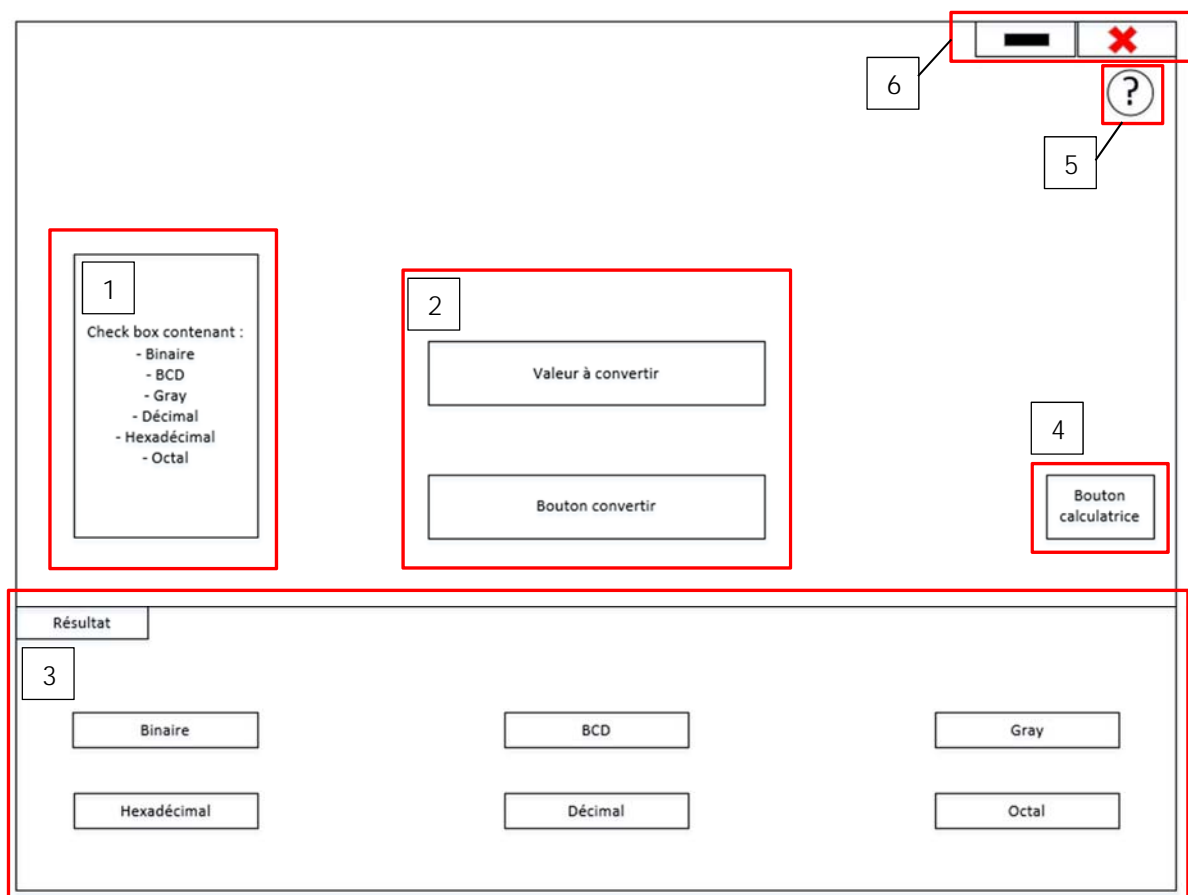


Figure 1 Design graphique de la FormConvert

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.

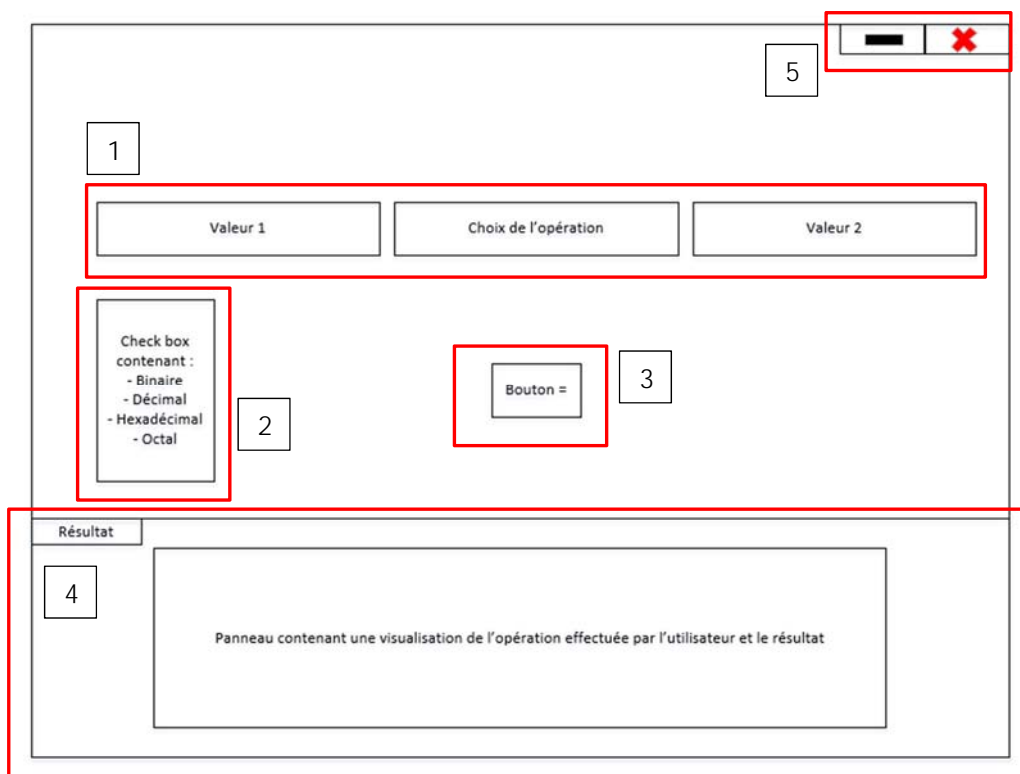


Diagramme de la fenêtre de calculatrice (FormCalculator) avec des numéros 1 à 5 indiquant des éléments spécifiques :

- 1 : Zone d'entrée pour la valeur à convertir.
- 2 : Zone de sélection de la base de destination (Binaire, Décimal, Hexadécimal, Octal).
- 3 : Bouton d'égalité (=) pour effectuer la conversion.
- 4 : Zone d'affichage du résultat.
- 5 : Bouton de réduction/fermeture (icône de barre et croix).

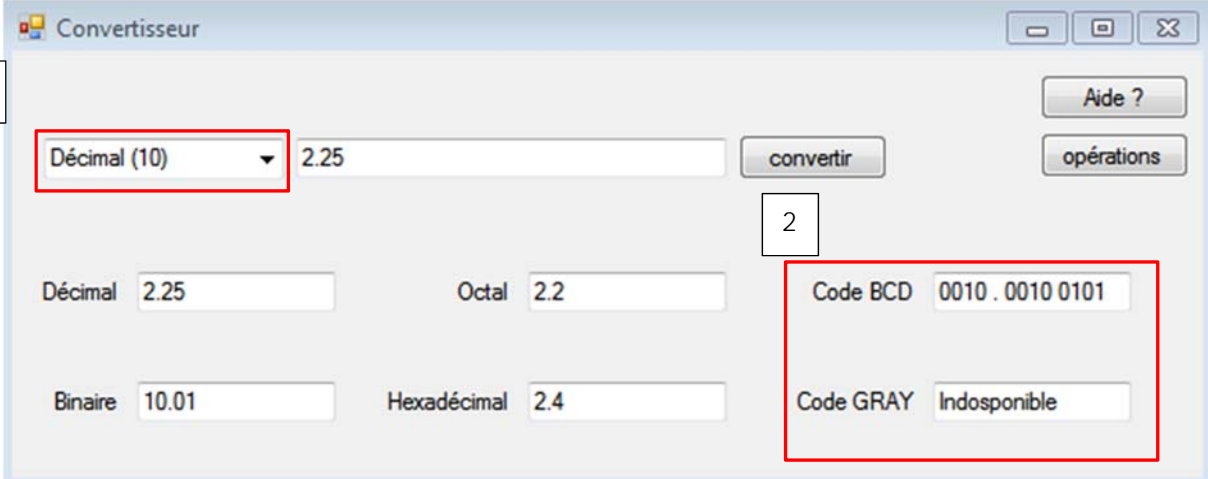
Le diagramme illustre la structure de la fenêtre avec des zones d'entrée, de sélection, de calcul et d'affichage, ainsi qu'un bouton de gestion de la fenêtre.

Figure 2 Design graphique de la FormCalculator

- 1) L'utilisateur entre les valeurs et choisi l'opération à effectuer (addition, soustraction et multiplication).
- 2) L'utilisateur choisi 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. Entant 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.



La figure présente une maquette d'une fenêtre intitulée "Convertisseur". Elle contient un menu déroulant "Décimal (10)" (encadré rouge 1) et un champ de texte "2.25". À droite se trouvent les boutons "convertir" et "opérations", ainsi qu'un bouton "Aide ?". En dessous, il y a quatre champs de texte : "Décimal 2.25", "Octal 2.2", "Binaire 10.01" et "Hexadécimal 2.4". À droite de ces champs, il y a deux autres champs : "Code BCD 0010 . 0010 0101" et "Code GRAY Indisponible" (encadré rouge 2). Une légende "2" est placée à côté du bouton "convertir".

Figure 3 Maquette de la FormConvert

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.

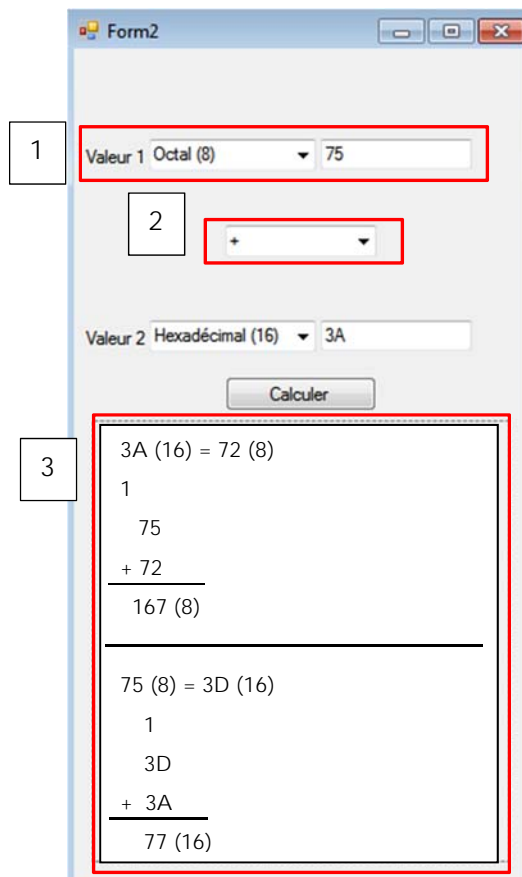


Figure 4 shows a screenshot of a software interface titled 'Form2'. It contains two input fields for numbers. The first field, labeled 'Valeur 1 Octal (8)', has the value '75'. The second field, labeled 'Valeur 2 Hexadécimal (16)', has the value '3A'. Between these fields is a dropdown menu showing a plus sign '+'. Below the input fields is a 'Calculer' button. At the bottom of the interface, there is a large text area displaying the results of the calculation. The results are shown in two columns. The first column shows the conversion of '3A (16)' to '72 (8)' and then the addition of '75' to '72' to get '167 (8)'. The second column shows the conversion of '75 (8)' to '3D (16)' and then the addition of '3A' to '3D' to get '77 (16)'. Red boxes highlight the input fields, the operation dropdown, and the results area.

Figure 4 Maquette de la FormCalculator

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 en colonne avec les retenues.

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

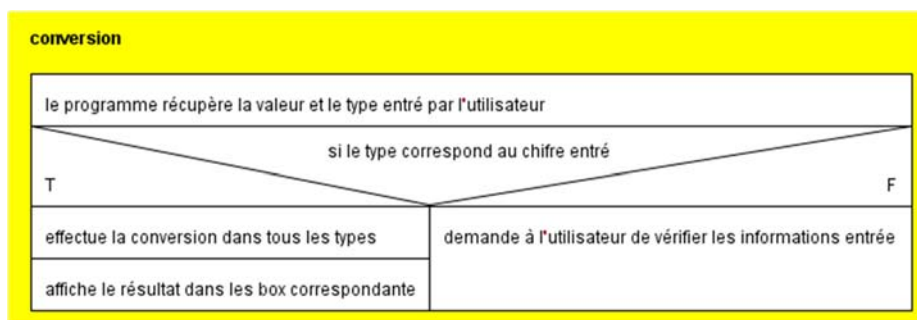


Figure 5 Structogramme conversion

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.

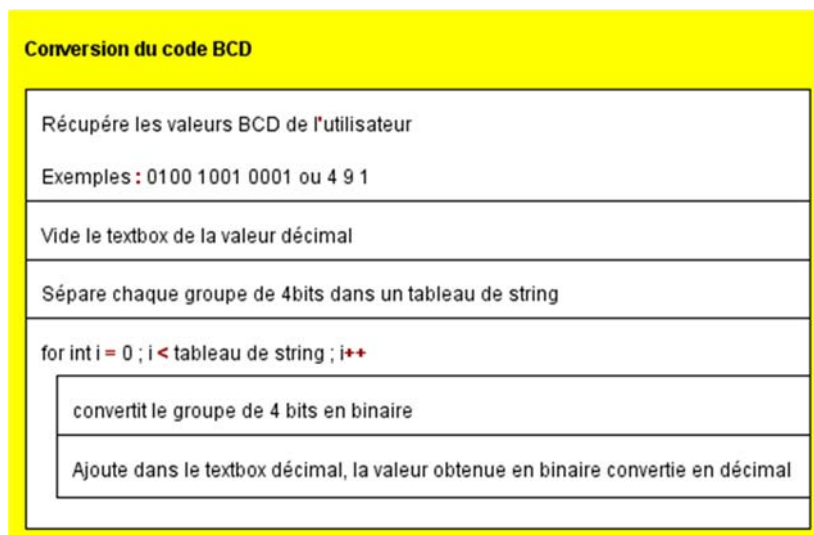


Figure 6 Structogramme de la fonction de conversion du BCD

Ce qui suis 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.

| | | | | | |
|---|---|---|---|---|----------------|
| 1 | 1 | 0 | 1 | 1 | Nombre binaire |
| ↓ | + | + | + | + | |
| 1 | 1 | 1 | 0 | 1 | |
| | | | | | Code Gray |

Ce qui donne le structogramme suivant pour la fonction binaireToGray

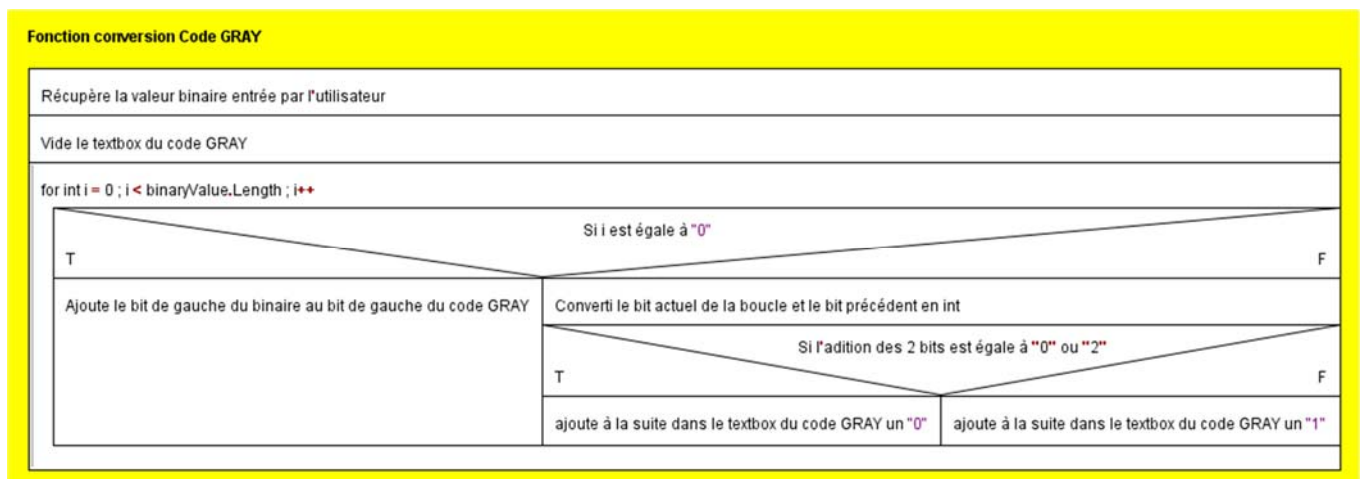


Figure 7 Structogramme de la fonction de conversion du GRAY

3) La méthode Calcul

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

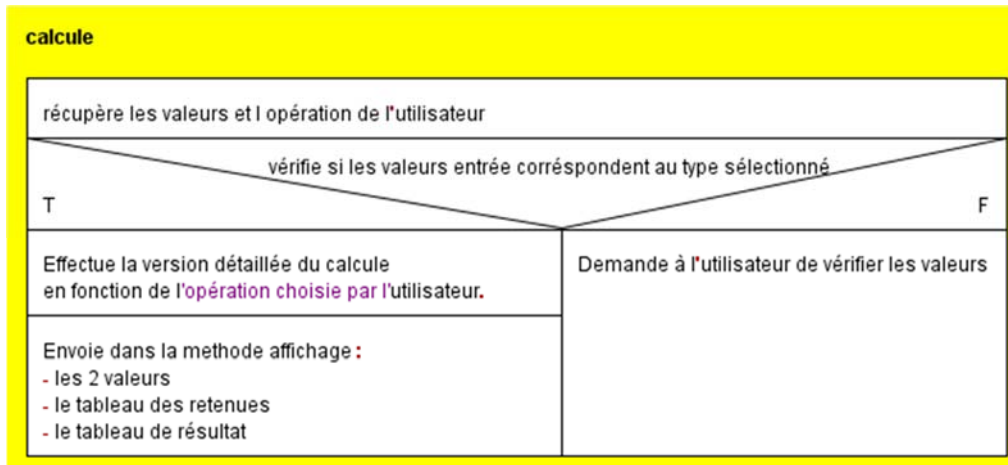


Figure 8 Structogramme de la fonction des opérations

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

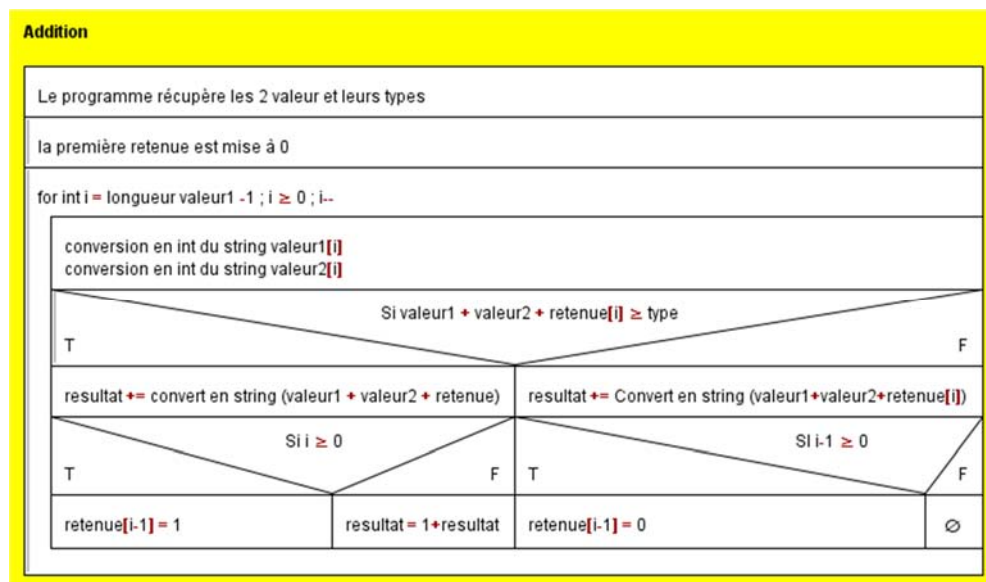


Figure 9 Structogramme de la fonction de l'addition

Toutes les fonctions pour les opérations suivront le même schéma. En effectuant directement la version détaillée du calcul, j'économise du temps et des lignes de codes.

4) La méthode Affichage

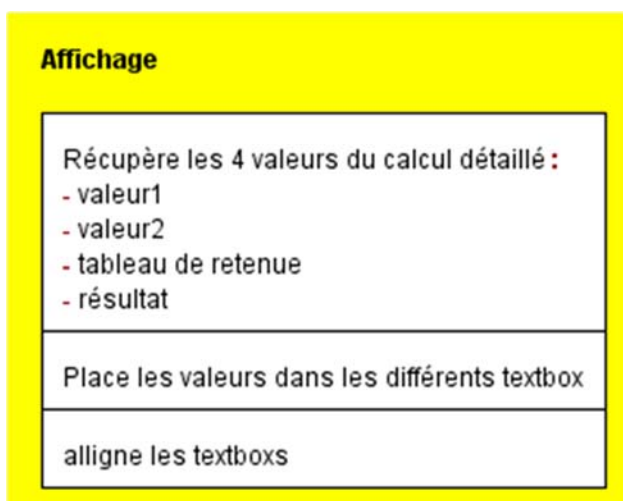


Figure 10 Structogramme de la fonction d'affichage

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 4.1.13.

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 suivront une grille prédéfinie, et le tout sera détaillé dans le point 5 du rapport.

3.4 Planification détaillée

[La planification détaillée du projet se trouve dans les annexes au chapitre 8.4](#)

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. 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 }
```

Figure 11 Fonction convertToBcd

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              //Récupère 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

```

Figure 12 Fonction convertToGray

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;

```

Figure 13 Fonction convertButton_Click

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     }
```

Figure 14 Séparation des types de bases

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 }
```

Figure 15 Lancement du convertToAll

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  }
```

Figure 16 Fonction ConvertToAll

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

```
.ToInt32(userValue, baseType);
```

int Convert.ToInt32(string value, int fromBase) (+ 18 surcharges)
Convertit la représentation sous forme de chaîne d'un nombre dans une base spécifiée en un entier 32 bits signé équivalent.

Figure 17 Description des arguments de ConvertToInt32

L'orsque 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 Calculs

La méthode pour effectuer les opérations, se situe dans la Form Calculaté. 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 calculs.

4.1.10 Fonction addValue

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

```

106      /// <summary>
107      /// Fonction permettant d'effectuer une addition en colonne entre 2 valeurs de meme base entrée par l'utilisateur.
108      /// </summary>
109      /// <param name="value1">Valeur 1 de l'utilisateur convertie dans son format</param>
110      /// <param name="value2">Valeur 2 de l'utilisateur convertie dans son format</param>
111      /// <param name="baseValue">Prends la base de la valeur 1 ou 2 pour afficher le résultat en fonction</param>
112      /// <param name="result">Stock et permet l'affichage du résultat</param>
113      /// <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>
114      private void addValue(string value1, string value2, int baseValue, string result, int[] retenue)
115      {
116
117          retenue[value1.Length-1] = 0;

```

Figure 18 Fonction addValue

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  }

```

Figure 19 Système de retenue

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 grand 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.

```

170  /// <summary>
171  /// Fonction permettant d'effectuer une soustraction en colonne entre 2 valeurs de meme base entrée par l'utilisateur.
172  /// </summary>
173  /// <param name="value1">Valeur 1 de l'utilisateur convertie dans son format</param>
174  /// <param name="value2">Valeur 2 de l'utilisateur convertie dans son format</param>
175  /// <param name="baseValue">Prends la base de la valeur 1 ou 2 pour afficher le résultat en fonction</param>
176  /// <param name="result">Stock et permet l'affichage du résultat</param>
177  /// <param name="restraint">Tableau de int permettant de stocker les retenues lors de l'addition en colonne. Est aussi utilisée lors de l'affichage</param>
178  private void subtractValue(string value1, string value2, int baseValue, string result, int[] restraint)
179  {
180      char[] charValue1 = value1.ToCharArray();
181      restraint[value1.Length - 1] = 0;
182
183      //pour chaque chiffre effectue une soustraction et place les retenues dans le tableau à l'index actuel
184      for (int i = value1.Length - 1; i >= 0; i--)
185      {
186          string stringValue1Unit = Convert.ToString(charValue1[i]);
187          string stringValue2Unit = Convert.ToString(value2[i]);
188
189          int intValue1Unit = Convert.ToInt32(stringValue1Unit);
190          int intValue2Unit = Convert.ToInt32(stringValue2Unit);

```

Figure 20 Initialisation de la fonction *subtractValue*

Une fois les valeurs transformée et envoyée dans la fonction, le programme va soustraire à chaque unité.

```

192      //si la soustraction est supérieur à 0 le programme ne va pas placer de retenues
193      if (intValue1Unit + restraint[i] - intValue2Unit >= 0)
194      {
195          result = Convert.ToString(intValue1Unit + restraint[i] - intValue2Unit) + result;
196      }
197
198      //sinon le programme diminue le chiffre suivant de 1 et place la base en retenue de la valeur actuelle
199      else
200      {
201          string temp = Convert.ToString(value1[i - 1]);
202          int valueDiminued = Convert.ToInt32(temp);
203          temp = Convert.ToString(valueDiminued - 1);
204          charValue1[i-1] = Convert.ToChar(temp);
205          restraint[i] = baseValue;
206          result = Convert.ToString(intValue1Unit + restraint[i] - intValue2Unit) + result;
207      }
208      showResult(value1, value2, result);
209  }
210

```

Figure 21 Système de soustraction et des retenues

Si le résultat est plus petit que 0, la fonction va alors mettre en place la retenue. Pour ce faire elle prend le chiffre suivant et le diminue de 1. La retenue, quant à elle, vaut la valeur de la base (sauf pour le binaire où elle vaut 1). Ensuite, elle va refaire la soustraction avec toutes les nouvelles valeurs, et placer ce résultat dans la variable « result ».

Ensuite elle va envoyer les deux valeurs et le résultat dans la fonction d'affichage.

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.

```
219  /// <summary>
220  /// Fonction permettant d'effectuer une multiplication en colonne entre 2 valeurs de même base entrée par l'utilisateur.
221  /// </summary>
222  /// <param name="value1">Valeur 1 de l'utilisateur convertie dans son format</param>
223  /// <param name="value2">Valeur 2 de l'utilisateur convertie dans son format</param>
224  /// <param name="baseValue">Prends la base de la valeur 1 ou 2 pour afficher le résultat en fonction</param>
225  private void multipliacateValue(string value1, string value2, int baseValue)
226  {
227      //Contrairement aux autres fonctions, vu que ce sont des tableaux à 2 dimensions, ils sont déclarés dans la fonction
228      string[,] result = new string[value1.Length, value1.Length+2];
229      int[,] restraint = new int[value1.Length, value1.Length+2];
230
231      //première retenue toujours à 0
232      restraint[value1.Length - 1, value2.Length - 1] = 0;
233  }
```

Figure 22 Initialisation de la fonction *multiplyValue*

Dans le cas d'une valeur à 2 chiffres, 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 permettent une meilleure lisibilité du code.

Pour le fonctionnement, je reste sur mon principe de double boucle for pour permettre une multiplication d'absolument tous les nombres.

```
235 //Récupère chaque unité des 2 valeurs
236 for (int i = value1.Length - 1; i >= 0; i--)
237 {
238     int tempRestraint = 0;
239     string unitValue1 = Convert.ToString(value1[i]);
240     int factor1 = Convert.ToInt32(unitValue1);
241     int counterRestraint = value1.Length+1;
```

Figure 23 Algorithme pour les retenues

Dans la première boucle, je récupère chaque unité de la valeur 1. Je mets la première retenue à 0, et lance une variable de compteur permettant de placer les 0 quand on change de niveau.

La deuxième boucle, permet de prendre chaque unité de la valeur 2. Une fois ceci récupéré, j'effectue une multiplication entre les deux, en y ajoutant la retenue d'avant stockée dans la variable « tempRestrain ». Si la valeur est en dessous de la base, j'inscris juste la valeur dans le tableau de résultat à la suite du précédent.

Si le résultat est plus grand que la base. Je prends le premier chiffre et le stock dans le tableau des retenues et dans la variable « tempRestrain ». Pour le deuxième chiffre, je le place à la suite dans le tableau de résultat.

Si la deuxième boucle arrive à 0 mais qu'une retenue est encore stockée dans la variable de retenue, elle sera automatiquement ajoutée à la suite dans le tableau de résultat.

4.1.13 Fonction d'affichage

Pour l'affichage, la fonction consiste à placer les valeurs données par les fonctions de calcul dans les bons textboxes.

On peut remarquer que le textbox pour le tableau des retenues a disparu. Ceci vient du fait

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://codertoolbox.net/number/>

Le principal souci du `parseInt`, est que si je veux l'intégrer, je dois revoir toute l'infrastructure de mon programme. Ce qui correspondrait à tout recommencer la partie conversion.

Il me permettrait, de séparer ce qui va avant et après la virgule. Ensuite par une fonction de conversion en bits par bits, je pourrais convertir le contenu d'après virgule. Pour ce qui se trouve avant, j'utiliserais la même méthode qu'actuellement.

4.2.2 Les retenues des opérations

Le principal souci des retenues, se situe au niveau de l'affichage. Le problème se trouve dans le fait de les placer correctement au-dessus des bonnes valeurs. Un moyen facile de contrer ce problème est de placer les valeurs à la fin du textbox et non au début.

Ensuite j'ai rencontré plusieurs soucis avec les retenues des multiplications. En effet, devant effectuer un tableau à x dimensions, je me retrouve avec un dilemme lors de l'affichage. Créer plusieurs textboxes ou le dimensionner pour que toutes les valeurs entre dedans.

La meilleure solution serait de placer un textbox par ligne de retenue. Permettant une meilleure gestion du placement de ces derniers.

4.2.3 Division détaillée

Pour la division détaillée, je rejoins le problème de la conversion. Pour que tout fonctionne parfaitement, je devrais passer par un `parseInt` et entièrement remanier le système d'opérations. Pour le moment il s'agit d'une division sommaire qui retourne un résultat entier (arrondi au-dessus).

4.2.4 Améliorations possible

- 1) Intégrer les nombres à virgule

- 2) Afficher les retenues à la fin des opérations
- 3) Effectuer du multi threading

Ceci permettra d'effectuer les opérations avec des valeurs de types différentes en même temps et d'afficher les deux résultats possibles. En contrepartie, ceci prends du temps à mettre en place et nécessite une bonne optimisation pour ne gagner que quelques microsecondes.

- 4) Intégrer un système de RegEx pour faciliter le contrôle des valeurs.
- 5) Modifier la fonction division pour qu'elle la fasse de manière détaillée.

5 TESTS

5.1 Grille de tests

| Fonction à tester | Résultat attendu | Date de test | OK / KO | Résultat obtenu | Remarques | Solutions |
|---|--|--------------|---------|---|---|--|
| Page de conversion | | | | | | |
| Convertir un nombre entier de format décimal en tous les autres formats | Le nombre doit apparaitre dans chaque checkbox correspondant au format | 01.06.2017 | OK | Tout les champs des autres formats se remplissent avec les bonnes valeurs | | |
| Convertir un nombre entier de format binaire en tous les autres formats | Le nombre doit apparaitre dans chaque checkbox correspondant au format | 01.06.2017 | OK | Tout les champs des autres formats se remplissent avec les bonnes valeurs | | |
| Convertir un nombre entier de format octal en tous les autres formats | Le nombre doit apparaitre dans chaque checkbox correspondant au format | 01.06.2017 | OK | Tout les champs des autres formats se remplissent avec les bonnes valeurs | | |
| Convertir un nombre entier de format hexadécimal en tous les autres formats | Le nombre doit apparaitre dans chaque checkbox correspondant au format | 01.06.2017 | OK | Tout les champs des autres formats se remplissent avec les bonnes valeurs | | |
| Convertir un nombre entier de format BCD en tous les autres formats | Le nombre doit apparaitre dans chaque checkbox correspondant au format | 01.06.2017 | OK | Tout les champs des autres formats se remplissent avec les bonnes valeurs | Penser à bien espacer les groupes de 4 bits | |
| Convertir un nombre entier de format Gray en tous les autres formats | Le nombre doit apparaitre dans chaque checkbox correspondant au format | 01.06.2017 | OK | Tout les champs des autres formats se remplissent avec les bonnes valeurs | | |
| Convertir un nombre en ne choisissant pas son format | Une erreur doit apparaitre pour prévenir l'utilisateur | 01.06.2017 | OK | Un messageBox apparait pour prévenir l'utilisateur qu'il faut une valeur et une base pour convertir | | |
| Essayer de convertir une valeur vide | Une erreur doit apparaitre pour prévenir l'utilisateur | 01.06.2017 | OK | Un messageBox apparait pour prévenir l'utilisateur qu'il faut une valeur et une base pour convertir | | |
| Essayer de convertir des lettre (en dehors du format hexadécimal) | Une erreur doit apparaitre pour prévenir l'utilisateur que les lettres sont autorisées uniquement en hexadécimal | 01.06.2017 | KO | Le programme passe outre la vérification et plante lors de la conversion | | Intégrer des RegEX pour vérifier les caractères de chaque valeur |
| Afficher le panneau d'aide | Un messageBox doit apparaitre pour donner les informations ainsi que les remarques sur le programme | 01.06.2017 | OK | Le messageBox s'affiche bien et permet de montrer les informations de bases | | |

| Page des opérations | | | | | | |
|---|--|------------|----|--|--|---|
| Addition de deux nombre de meme base | Dans le panneau résultat une addition en colone s'affiche avec le résultat | 01.06.2017 | OK | Pour les valeurs 23 et 15 on obtient bien le résultat 38 | | |
| Addition de deux nombre avec des bases différentes | Dans le panneau résultat une addition en colone s'affiche avec le résultat | 01.06.2017 | OK | Le résultat de l'opération est donné dans la base de la valeur 1 | | |
| Soustraction de deux nombre de meme base | Dans le panneau résultat une Soustraction en colone s'affiche avec le résultat | 01.06.2017 | OK | Pour les valeurs 23 et 15 on obtient bien le résultat 08 | La soustraction ne peut pas aller dans les valeurs négatives | |
| Soustraction de deux nombre avec des bases différentes | Dans le panneau résultat une Soustraction en colone s'affiche avec le résultat | 01.06.2017 | OK | Le résultat de l'opération est donné dans la base de la valeur 1 | La soustraction ne peut pas aller dans les valeurs négatives | |
| Multiplication de deux nombre de meme base | Dans le panneau résultat une Multiplication en colone s'affiche avec le résultat | 01.06.2017 | KO | Le programme plante avant d'afficher les résultats | Suite à la mise à jour de la fonction de multiplication, le 31.05.2017, il se peut qu'un petit bug se soit glissé dans le code | Corriger le bug dans la fonction multiplicateur |
| Multiplication de deux nombre de meme base | Dans le panneau résultat une Multiplication en colone s'affiche avec le résultat | 02.06.2017 | OK | Le programme affiche le résultat | La fonction fonctionne mais il reste un petit soucis. Quand je multiplie une valeur 2 plus grande que la valeur 1 le code plante | |
| Multiplication de deux nombre avec des bases différentes | Dans le panneau résultat une Multiplication en colone s'affiche avec le résultat | 01.06.2017 | KO | Le programme plante avant d'afficher les résultats | Suite à la mise à jour de la fonction de multiplication, le 31.05.2017, il se peut qu'un petit bug se soit glissé dans le code | |
| Division de deux nombre de meme base | Dans le panneau résultat une Division en colone s'affiche avec le résultat | 01.06.2017 | OK | Le résultat s'affiche bien dans le panneau de résultat | Penser que le nombre affiché est un nombre sans virgule | |
| Division de deux nombre avec des bases différentes | Dans le panneau résultat une Division en colone s'affiche avec le résultat | 01.06.2017 | OK | Le résultat s'affiche bien dans le panneau de résultat | Penser que le nombre affiché est un nombre sans virgule | |
| Effectuer une opération en n'entrant aucune valeur | Dans le panneau de résultat, un texte viens prévenir l'utilisateur de son erreur | 01.06.2017 | KO | Le programme plante | | Corriger le bug en intégrant une fonction de vérification des valeurs |
| Effectuer une opération en n'entrant qu'une valeur sur les deux | Dans le panneau de résultat, un texte viens prévenir l'utilisateur de son erreur | 01.06.2017 | KO | Le programme plante | | Corriger le bug en intégrant une fonction de vérification des valeurs |
| Effectuer une opération avec deux valeurs mais aucun format | Dans le panneau de résultat, un texte viens prévenir l'utilisateur de son erreur | 01.06.2017 | KO | Le programme plante | | Corriger le bug en intégrant une fonction de vérification des valeurs |

5.2 Dossier des tests

5.2.1 Bilan des tests

5.2.2 Découverte des bugs

- Convertir un nombre avec une lettre dedans.
 - o Date de découverte : 01.06.2017
 - o Solution : Intégrer un système de REGEX dans le programme
 - o Estimation du temps pour résoudre : 2h
- Lors d'une multiplication, le programme plante avant d'afficher le résultat
 - o Date de découverte : 01.06.2017
 - o Solution : corriger la fonction pour trouver le bug
 - o Temps passé à résoudre : 1h30
 - o Date d'intégration : 02.06.2017
- Sur la form « Calculator », le programme ne vérifie pas si les textbox sont bien remplis
 - o Date de découverte : 01.06.2017

- o Solution : intégrer un système de vérification comme dans la form « Converter »
 - o Temps passé à résoudre : 1h
 - o Date d'intégration de la solution : 01.06.2017
- La conversion de nombre à virgule
 - o Date de découverte : 18.05.2017
 - o Solution : Une solution est disponible au point 4.2.1 du rapport.
 - o Estimation du temps pour résoudre : 10h
- La fonction « multiplyValue »
 - o Date de découverte : 02.06.2017
 - o Problème : Lors d'une multiplication, les opérations fonctionnent normalement. Cependant il reste un souci, si on met une valeur 2 plus longue (en nombre de caractère) que la valeur 1, le programme va planter.
 - o Solution : Modifier les tailles des tableaux de résultat et de retenue, pour qu'il puisse être possible de calculer toutes les valeurs.
 - o Estimation du temps pour résoudre : 3h
- Opérations avec un nombre à virgule ou un nombre négatif
 - o Date de découverte : 18.05.2017
 - o Problème : Tout comme les conversions, les nombres à virgule ou les nombres négatifs ne passent pas.
 - o Solution : Remanier le programme avec la méthode ParseInt pour faire fonctionner le tout.
 - o Estimation du temps pour résoudre : 4h

6 CONCLUSION

6.1 Bilan des fonctionnalités demandées

Pour les fonctionnalités demandées, je vais répartir de la manière suivante :

1) Fonctionnalités terminées

- Conversion d'un nombre entier d'un des six types en tous les autres types
- Opérations de nombre entier de n'importe quel type

2) Fonctionnalités inachevées

- Conversion de nombre à virgule
- Addition et soustraction de nombre à virgule ou négatif.

Toutes les informations relatives à ces deux points se trouvent dans les chapitres 4.2 et 5.2. Pour expliquer pourquoi je mets une différence de temps entre la conversion et les opérations à virgule, c'est que malgré la documentation lue, il me faudrait un temps d'adaptation pour tout mettre en place.

6.2 Bilan de la planification

La planification se trouve dans les annexes au point [8.4](#). Mais pour expliquer les différences de temps je vais le reprendre semaine par semaine.

Semaine 1 :

Pour la première semaine il n'y a pas grand-chose à dire. L'analyse m'a pris moins de temps que prévu, et m'a permis de bien tout intégrer dans la documentation.

Semaine 2 :

A partir de là, les choses ont pris une drôle de tournure. J'ai dû refaire mes maquettes, donc cela m'a pris du temps sur l'analyse mais je compense le temps perdu par une mise en place plus rapide des deux interfaces du programme.

Ensuite vient le moment de commencer la méthode de conversion. Malgré le fait d'avoir analysé ce que je voulais faire, je prends vite du retard (je prends deux fois plus de temps que prévu) à cause des méthodes de conversion du code BCD et du code GRAY. C'est aussi à ce moment que je remarque que les nombres à virgule ne fonctionnent pas. Je cherche une solution mais y passant trop de temps je documente le problème et passe à la suite.

Semaine 3 :

A partir de ce moment, je décide de faire mon projet en suivant au mieux ma planification. Mais tous les petits soucis rencontrés auparavant me font prendre du retard. Pour être le plus économe en temps possible, je travaille le plus possible sur la fonction de calcul. Au bout de la semaine, je dépasse un peu le temps prévu, mais c'est dû au fait que j'ai décidé à ce moment de fusionner les fonctions de calcul et d'affichage.

Semaine 4 :

A ce moment, je décide de d'intégrer dans la doc tout ce que je n'y avais pas encore mis. Après je cumule du temps sur la fonction de conversion en essayant d'intégrer les nombres à virgule.

Semaine 5 :

Je commence par régler quelques petits bugs repérer à la fin de la semaine précédant. Après un rapide test, je remarque que la fonction de multiplication ne fonctionne pas du tout. A ce moment je décide de mettre de côté les nombres à virgule et les négatifs, pour pouvoir finir ceci. Ce qui explique le dépassement de temps sur cette tâche.

Une fois ceci réglé je rattrape mon retard en finissant ma documentation.

Semaine 6 :

En tenant compte du lundi de congé, cette semaine a principalement servi à corriger deux ou trois bugs de mise en page sur mon rapport et à terminer les impressions.

6.3 Bilan personnel

En conclusion de ce bilan, je constate que malgré le fait que j'ai quand même suivi un maximum ma planification dans ma réalisation, le fait que je prenne du retard sur les deux fonctions principales du programme m'a empêché d'ajouter certaine fonction, qui aurait rendu mon application meilleure.

Si le projet était à refaire, je prendrais un peu plus de temps sur mon analyse. Ceci m'aurait permis d'éviter quelques petits soucis comme la fonction de multiplication ou les nombres à virgule. Sinon je pense que je ne changerais pas énormément ma façon de travailler car j'estime avoir un rythme assez soutenu.

Pour les points positifs du projet, je noterai un chef de projet et des experts toujours prêt à me donner un coup de main ou à répondre à mes questions.

Ce projet m'a aussi appris à mieux gérer mon temps. Avant je passais beaucoup de temps sur la recherche de solutions pour que tout fonctionne. Maintenant je document pourquoi ça ne marche pas, et j'y reviens quand j'ai du temps ou une idée. Il m'a aussi appris à mieux gérer mon code, et à mieux optimiser ma façon de coder.

Si je devais donner suite à ce projet, je reviendrais sur les améliorations possible données au point 4.2.4. Et j'essaierais de continuer à optimiser mon code pour le rendre plus performant.

Je tiens à remercier les personnes suivantes :

- Daniel Jorge et M. Melly : Pour leur idée du multi-threading
- Sacha Grenier : pour avoir corrigé un bout de mon rapport

-
- Le chef de projet et les experts : pour le retour régulier et leurs idées d'améliorations

Stéphane Pittier

ETML 07.06.2017

7 DIVERS

7.1 Journal de travail

[Le journal de travail se trouve dans les annexes au chapitre 8.5](#)

7.2 Webographie

- Analyse concurrentielle
 - www.aly-abbara.com/utilitaires/convertisseur/convertisseur_chiffres.html
 - www.sebastienquillon.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>
- C# Variable en lecture seule
 - <https://openclassrooms.com/forum/sujet/c-erreur-indexeur-ne-peut-pas-etre-assigne-41392>
- C# méthode parseInt
 - [https://msdn.microsoft.com/en-us/library/1kc6b02f\(v=vs.100\).aspx](https://msdn.microsoft.com/en-us/library/1kc6b02f(v=vs.100).aspx)
- C# RegEx
 - [https://msdn.microsoft.com/fr-fr/library/hs600312\(v=vs.110\).aspx](https://msdn.microsoft.com/fr-fr/library/hs600312(v=vs.110).aspx)
- Lien du Git
 - <https://github.com/StephanePittier/TPI>

8 ANNEXES

8.1 Cahier des Charges

1 INFORMATIONS GENERALES

| | | |
|---|--|---|
| Candidat : | Nom : PITTIER | Prénom : STEPHANE |
| | ✉ : malto:pittierst@etnl.educa.net2.ch | ☎ : |
| Lieu de travail : | ETML, Sébeillon 12 1004 Lausanne | |
| Chef de projet : | Nom : Gruaz | Prénom : Gilbert |
| | ✉ : malto:gilbert.gruaz@vd.educa.net2.ch | ☎ : 079 338 7808 |
| Expert 1 : | Nom : Carrel | Prénom : Xavier |
| | ✉ : malto:xavier.carrel@cpnv.ch | ☎ : 079 212 96 21 |
| Expert 2 : | Nom : Zahn | Prénom : Jean |
| | ✉ : malto:jean.zahn@vd.ch | ☎ : |
| Dates de réalisation : | Du lundi 1 mai au mercredi 7 juin à 11H25 | |
| Horaire de travail : (Basé sur l'horaire officiel) | Lundi | 08h00-11h25 - Pentecôte: 3 juin 2017 |
| | Mardi | - - |
| | Mercredi | 08h00-12H15 13h10-16h35 Exa CG 31 mai 2017 après-midi |
| | Jeudi | 08h00-11H25 12h20-16h35 Pont de l'Asc, 25 mai 2017 |
| | Vendredi | 08h00-12H15 13h10-16h35 Pont de l'Asc, 26 mai 2017 |
| Présentation : | Entre le mercredi 14 et jeudi 15 juin 2017 | |
| Nombre d'heures : | Environ 110 heures | |
| Planning (en H ou %) | Analyse : 10%, Implémentation 60%, Tests 10%, Doc. 20% | |

2 PROCÉDURE

Le candidat réalise un travail personnel sur la base d'un cahier des charges reçu le 1er jour.

Le cahier des charges est approuvé par la i-CQ VD. Il est en outre présenté, commenté et discuté avec le candidat. Par sa signature, le candidat accepte le travail proposé.

Le candidat a connaissance de la feuille d'appréciation avant de débiter le travail.

Le candidat est entièrement responsable de la sécurité de ses données.

En cas de problèmes graves, le candidat avertit au plus vite les deux experts et son chef de projet.

Le candidat a la possibilité d'obtenir de l'aide, mais doit le mentionner dans son dossier de projet.

A la fin du délai imparti pour la réalisation du TPI, le candidat doit transmettre par courrier électronique le dossier de projet aux deux experts et au chef de projet. En parallèle, une copie papier du rapport doit être fournie sans délai en trois exemplaires. Cette dernière doit être en tout point identique à la version électronique.

3 TITRE

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

4 SUJET

Il s'agit d'implémenter une application qui va permettre aux utilisateurs (des élèves par exemple, mais aussi des enseignants), de contrôler les résultats des exercices de conversions de données numériques en binaire, octal, décimal, hexadécimal, BCD, nombres réels à virgule flottante, ainsi que des opérations élémentaires comme des compléments à deux, additions et soustractions

5 MATÉRIEL ET LOGICIEL À DISPOSITION

1 ordinateur standard ~~etml~~, avec la structure habituelle

6 PRÉREQUIS

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

7 DESCRIPTIF DU PROJET

Les élèves sont parfois soumis à résoudre des exercices d'entraînements pour vérifier ce qu'ils ont compris des concepts et théories vues en classe. Par exemple, comment convertir un nombre réel à virgule fixe de décimal en hexadécimal (ou $134.45_{10} \rightarrow ?_{16}$).

Actuellement, à l'ETML, en informatique, un module nommé ELEOC-NUM pour électronique numérique, traite de ces sujets. Les supports présentent de la théorie et des exercices. Toutefois, il conviendrait de pouvoir laisser les élèves résoudre des mêmes exercices, mais avec des données différentes, et de pouvoir vérifier leurs réponses de manière automatique.

L'idée est d'avoir une application qui permette aux élèves de s'entraîner pour les fonctionnalités suivantes, avec des mots de 16 bits :

- Conversions de nombres décimaux en BCD et Gray, et réciproquement
- Conversions de nombres décimaux, binaires, octaux et hexadécimaux (entiers et/ou réels à virgules fixes) en nombres dans les 3 autres bases que la base courante.
- Additions et soustractions de 2 nombres binaires, octaux et hexadécimaux (entiers et/ou réels à virgules fixes, positifs et/ou négatifs)
- Multiplications et divisions de 2 nombres binaires, octaux et hexadécimaux (entiers)
- Conversion d'un nombre réel décimal, positif ou négatif, en nombre binaire à virgule flottante et réciproquement.

8 POINTS ÉVALUÉS DURANT LE PROJET

- Le comportement et l'engagement durant toute la période pour le travail
- La bonne tenue à jour, systématique, des documents (JNLTRAV, rapport, etc.)
- Les interactions avec le chef de projet

9 LIVRABLES

Le candidat est responsable de mettre à disposition, pour son chef de projet et les deux experts, un dépôt GIT, sur un cloud, avec :

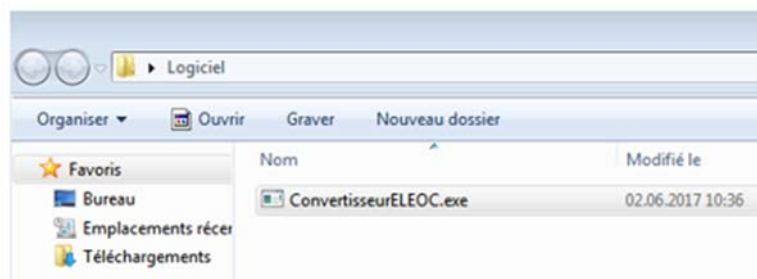
- La planification initiale
- Le rapport de projet
- Le journal de travail
- Le code qu'il produit

8.2 Manuel Utilisateur

Manuel Utilisateur Convertisseur numérique

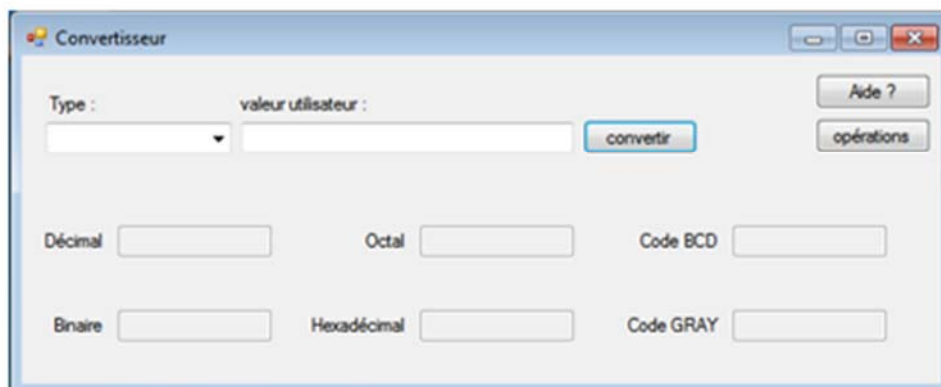
1 LANCEMENT DE L'APPLICATION

Pour lancer l'application, vous avez juste à cliquer sur l'exécutable dans le dossier logiciel.



2 PARTIE CONVERSION

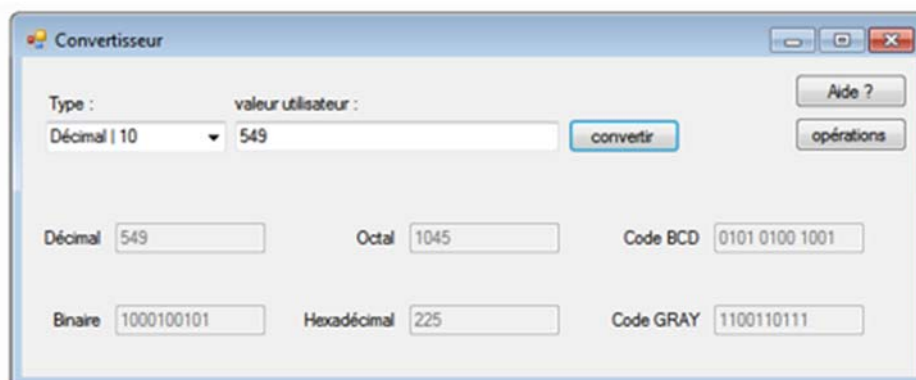
En ouvrant le programme vous allez arriver sur la partie de conversion de l'application.



A partir de ce moment plusieurs options s'offre à vous :

- 1) Convertir un nombre
- 2) Ouvrir le panneau des opérations
- 3) Afficher l'aide
- 4) Quitter le programme

Pour la partie conversion, il vous suffit d'entrer une valeur et de choisir son type. Ensuite en cliquant sur le bouton convertir, vous devriez vous retrouver avec un résultat comme ceci :

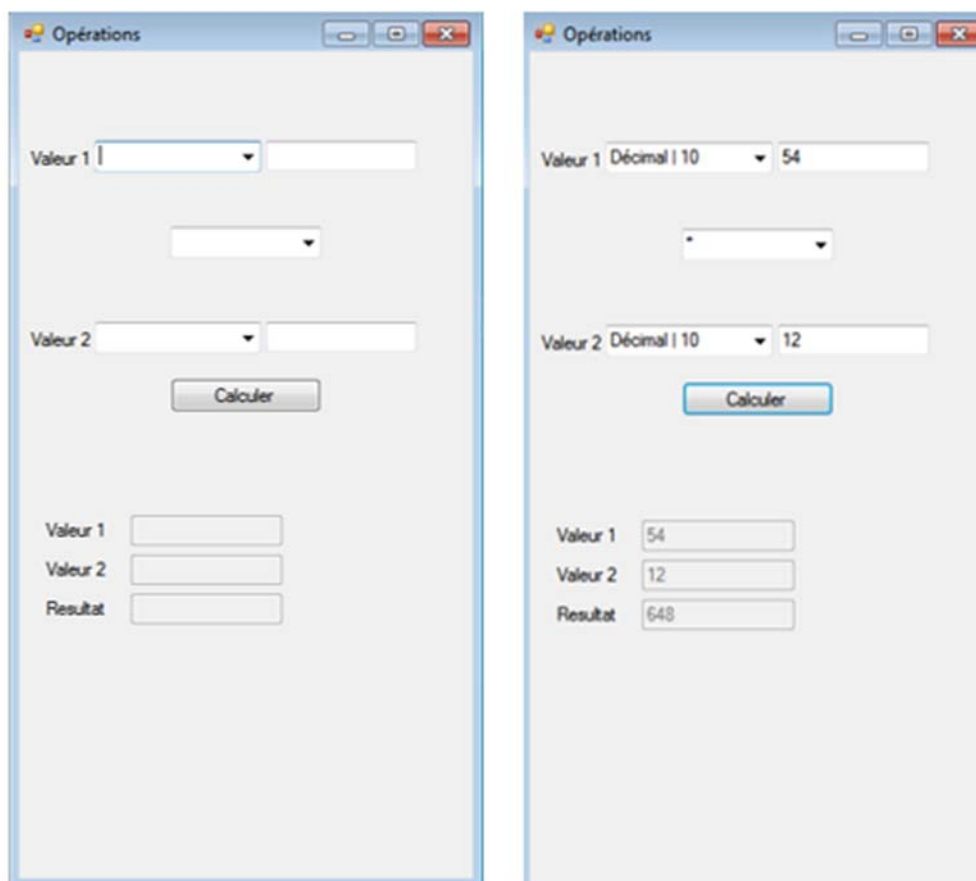


Remarques : Les nombres à virgule ne fonctionnent pas

Si vous cliquez sur le bouton aide, les informations complémentaires apparaîtront, ainsi que la remarques.

3 PARITE OPÉRATIONS

Pour la partie opérations, le système reste le même que pour la partie opérations.



The image displays two side-by-side screenshots of a software window titled "Opérations".

Left Screenshot: The window contains two input sections. The first section has a label "Valeur 1" followed by a dropdown menu and a text input field. Below this is another dropdown menu. The second section has a label "Valeur 2" followed by a dropdown menu and a text input field. A "Calculer" button is located below the second section. At the bottom, there are three labels: "Valeur 1", "Valeur 2", and "Resultat", each followed by a text input field.

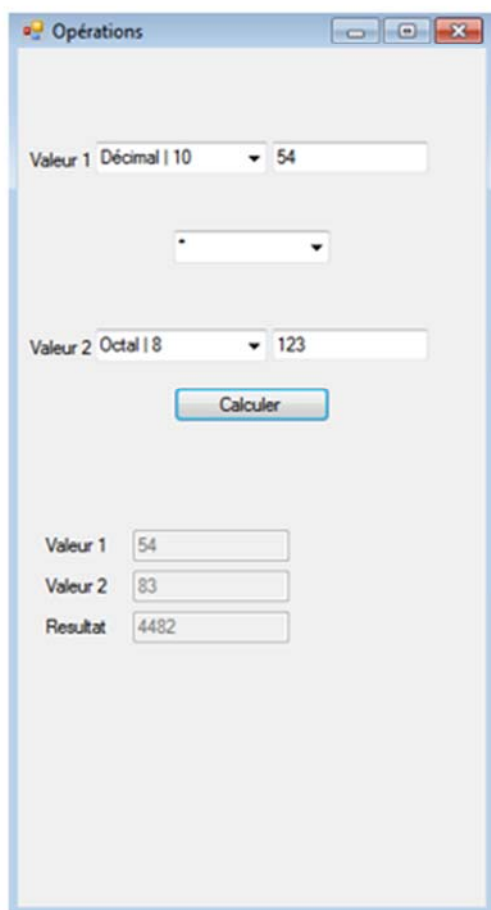
Right Screenshot: The same window is shown but with data entered. In the first section, the dropdown menu is set to "Décimal | 10" and the text field contains "54". The second dropdown menu now contains a multiplication symbol (*). In the second section, the dropdown menu is also set to "Décimal | 10" and the text field contains "12". The "Calculer" button is highlighted. At the bottom, the "Valeur 1" field contains "54", the "Valeur 2" field contains "12", and the "Resultat" field contains "648".

Vous pouvez choisir le type d'opérations que vous voulez entre :

- Addition
- Soustraction
- Multiplication
- Division

Attention, pour que l'opération s'effectue, il vous faut impérativement remplir tous les champs.

Si vous voulez effectuer une opération entre deux valeurs de types différents, le résultat de l'opération sera donné dans le type de la valeur 1.



Ce qui nous donne ceci pour un exemple de multiplication avec un nombre en base 10 et un autre en base 8.

Remarques : Pour les multiplications, si vous entrez une valeur 2 avec plus de caractère que la valeur 1 il y a des chances que le programme n'arrive pas à faire l'opérations.

8.3 Code source

Le code source se trouve dans le git à ce lien :

<https://github.com/StephanePittier/TPI/tree/master/Programme/ConvertisseurNumerique/ConvertisseurNumerique>

8.4 Planification détaillée

| Semaine 1 | | |
|--------------------|----------------|--|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Analyse | 12 | Entretien avec le chef de projet et l'expert et début de la planification initiale |
| Analyse | 15 | Fin de la planification initiale |
| Analyse | 54 | Analyse des besoins du programme |
| Documentation | 9 | Implémentation de l'analyse dans la documentation |
| Journal de travail | 3 | Remplissage du journal de travail |
| Total semaine | 93 | Max. 93 |

| Semaine 2 | | |
|-------------------------------|----------------|---|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Documentation | 16 | Intégrations de l'analyse dans le rapport |
| Création : Interface | 27 | Création de l'interface utilisateur en Windows form c# |
| Création : Méthode conversion | 25 | Création de la méthode de conversion pour les 6 formats de données |
| Création : débogage | 6 | petits test et correction des bugs présent à ce moment |
| Documentation | 16 | Intégration de de la création de l'interface et de la partie conversion dans le rapport |
| Journal de travail | 3 | Remplissage du journal de travail |
| Total semaine | 93 | Max. 93 |

| Semaine 3 | | |
|------------------------------|----------------|---|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Création : Méthode calcul | 25 | Création de la méthode pour effectuer les opérations (+, - et x) sur des chiffres de même format |
| Documentation | 16 | Intégration de la création de la méthode dans la documentation |
| Création : Méthode Affichage | 24 | Création de la méthode pour afficher une version détaillée du calcul réalisé dans la méthode précédente |
| Documentation | 16 | Intégration de la méthode de l'affichage dans la documentation |
| Création : débogage | 9 | vérification du fonctionnement du logiciel |
| Journal de travail | 3 | Remplissage du journal de travail |
| Total semaine | 93 | Max. 93 |

| Semaine 4 | | |
|---------------------|----------------|--|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Création : débogage | 12 | durant la phase de test, corriger les eventuelles erreurs détectée |
| Documentation | 9 | Suite de la documentation sur la partie pratique du projet |
| Tests | 15 | Début de la phase de test du programme selon la grille de tests |
| Journal de travail | 3 | Remplissage du journal de travail |
| Absence - Imprévu | 54 | Pont de L'ascension 25 et 26 mai 2017 |
| Total semaine | 93 | Max. 93 |

| Semaine 5 | | |
|---------------------|----------------|--|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Création : débogage | 24 | Correction des problèmes découverts pendant les tests |
| Tests | 18 | tests finaux du programme |
| Documentation | 36 | Documentation sur la phase de test et finition de la documentation |
| Journal de travail | 3 | Remplissage du journal de travail |
| Absence - Imprévu | 12 | Examen ECG du 31 mai 2017 |
| Total semaine | 93 | Max. 93 |

| Semaine 6 | | |
|--------------------|----------------|--|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Documentation | 9 | Finalisation de la documentation |
| Journal de travail | 3 | Remplissage du journal de travail |
| Absence - Imprévu | 12 | Pentecôte 5.06.2017 |
| Total semaine | 24 | Max. 93 |

8.5 Journal de travail

| Semaine 1 | | Date: |
|---------------|----------------|---|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Analyse | 6 | Lundi : Lecture du cahier des charges et rendez-vous avec l'expert |
| Analyse | 6 | Lundi : Début de la planification initial |
| Analyse | 6 | Mercredi : Rendez-vous avec monsieur Gruaz pour des explications supplémentaire et fin de la planification initiale |
| Analyse | 6 | Mercredi : Réflexion sur la façon de créer le programme. Imagination de l'interface et des diverses options du programme |
| Analyse | 5 | Mercredi : Création de la maquette de l'application sous Visio |
| Documentation | 4 | Mercredi : mise à jour de la documentations avec les informations de bases du TPI |
| Analyse | 6 | Mercredi : Analyse de la construction du programme. Quel seront les methode ou fonction à créer. Mise en service du Git et envoi du mail aux experts |
| Analyse | 6 | Jeudi : > Deconstruction du programme en 3 methode principale > Revision des cours ELEOC sur les format bianire, décimal, hexadécimal et octal |
| Analyse | 3 | Jeudi : > Debut de la création des structogrammes |
| Documentation | 3 | Jeudi : Mise à jour de la partie analyse de la documentation avec les nouveaux élément de l'application |
| Analyse | 15 | Jeudi : > Apprentissage du code BCD et GRAY > Recherche sur la meilleur fonction pour convertir les données > Création sur le papier d'un algorithme pour le code BCD > Création du structogrammes du BCD et de l'affichage |
| Analyse | 2 | Vendredi : Finition et amélioration des structogrammes |
| Documentation | 9 | Vendredi : Implémentation de l'analyse dans la Documentation |
| Documentation | 4 | Vendredi : > Implémentation des structogrammes dans la documentation |
| Documentation | 12 | Vendredi : > création de la planification détaillée |
| Total semaine | 93 | Max. 93 |

| Semaine 2 | | Date: |
|-------------------------------|----------------|--|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Analyse | 2 | Lundi : Convenu avec GGZ > Le journal de travail ne correspond pas aux attentes > Ne pas oublier les liens > Bien décrire mais pas trop de détails inutile |
| Journal de travail | 2 | Lundi : > Mise à jour du journal de travail pour y mettre les liens manquants à la semaine 1 |
| Documentation | 8 | Lundi : Mise à jour de la partie analyse et envoi du mail de la planification détaillée aux experts |
| Documentation | 2 | Mardi : > Ajouts des structogrammes manquants au rapport |
| Journal de travail | 1 | Mardi : Mise à jour des liens de la semaine 1 et 2 |
| Analyse | 3 | Mardi : Convenu avec GGZ > La maquette est une structure graphique. A modifier ! > Utiliser des liste déroulante à la place des checkbox > Mettre des liens fonctionnel dans le journal de travail > Faire une analyse concurrentielle d'un autre convertisseur pour montrer la plus value de mon projet |
| Analyse | 3 | Mardi : > Réalisation de l'analyse concurrentielle sur la base de 2 sites (les liens des sites ce trouvent dans la Webographie du rapport) |
| Analyse | 3 | Mardi : Création de la maquette du site avec visual studio |
| Documentation | 3 | Mardi : Intégration de la maquette dans le rapport |
| Documentation | 3 | Mardi : Modification de la maquette pour y mettre des valeurs réalistes en exemples |
| Création : Interface | 9 | Mardi : Création de l'interface utilisateur selon maquette et nommage selon normes ETML |
| Création : Interface | 1 | Jeudi : Intégration des valeurs par défaut dans les listes déroulantes du logiciel |
| Création : Méthode conversion | 5 | Jeudi : Création de la fonction de vérification des valeurs entrée par l'utilisateur |
| Création : Méthode conversion | 6 | Jeudi : Création du split pour récupérer la base de la conversion, et commencement de la méthode Conversion |
| Création : Méthode conversion | 6 | Jeudi : > Conversion de la valeur utilisateur dans les 4 types de bases (décimal, octal, binaire et hexa) > Modifications des valeurs possibles pour l'utilisateur dans la fonction de vérification |
| Création : Méthode conversion | 9 | Jeudi : > Création de la méthode pour convertir la valeur en code BCD depuis la valeur décimale |
| Création : Méthode conversion | 3 | Vendredi : > Finalisation de la méthode de conversion en code BCD |
| Création : Méthode conversion | 9 | Vendredi : > Créations des méthodes de conversion binaire -> GRAY et GARY -> binaire |
| Création : Méthode conversion | 3 | Vendredi : > Création de la méthode BCD -> Décimal (permettant tous les autres formats) > mise en forme du programme avec une methode convertToAll permettant une meilleur lisibilité du programme |
| Création : Méthode conversion | 3 | Vendredi : > Essai sur la fonction de conversion des nombres à virgule flottante. Celle imaginé durant l'analyse ne fonctionne pas |
| Documentation | 9 | Vendredi : > Correction de quelques erreurs et ajout de fonction dans la partie de réalisation |
| Total semaine | 93 | Max. 93 |

| Semaine 3 | | Date: |
|------------------------------|----------------|---|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Création : Méthode calcul | 6 | Lundi : > Création de la méthode CalculateValues qui permet les opérations arithmétiques |
| Analyse | 1 | Lundi : Convenu avec GGZ > GGZ va m'envoyer un fichier Excel qui contient quelque idée pour le convertisseur > Les chiffres à virgule flottante posent un soucis. Il sera réglé après la méthode de calcul > Le projet avance bien |
| Création : Méthode Affichage | 2 | Création d'un fichier Excel, reprenant une version papier, pour montrer comment l'affichage fonctionne |
| Création : Méthode calcul | 2 | Implémentation du tableau vu sur le Excel dans le programme |
| Journal de travail | 1 | Lundi : Remplissage du journal de travail |
| Création : Méthode calcul | 6 | Mercredi : Création de la fonction addValue permettant l'addition de 2 valeurs |
| Création : Méthode Affichage | 6 | Mercredi : Création de la méthode permettant d'afficher une version en colonne du calcul |
| Création : Méthode Affichage | 3 | Mercredi : Début de la méthode InitialisePannel qui permet de placer les label pour le détail des opérations. |
| Création : Méthode calcul | 3 | Mercredi : > Après réflexion, les opérations seront divisé en 4 fonctions (addition, soustraction, multiplication et division. > Création de la fonction addition > Création de la fonction de soustraction |
| Documentation | 8 | Mercredi : > Mise à jour de la documentation avec pour le calcul : - addition - soustraction > Mise à jour de la documentation avec la partie conversion: - convert to all |
| Journal de travail | 1 | Mercredi : Mise à jour du journal de travail avec les liens |
| Création : Méthode calcul | 6 | Jeudi : Mise à jour de la fonction addition et soustraction avec les nouveau tableau de retenue et de resultat. Pour faciliter les fonction multiplication et division |
| Création : Méthode calcul | 6 | Jeudi : Création de la fonction multiplication. 2 valeurs en base différente. Retenue et resultat |
| Analyse | 2 | Jeudi : Petite réflexion sur les multiplications et les divisions. Comment les expliquer étapes par étapes |
| Création : Méthode calcul | 13 | Jeudi : Suite de la fonction de multiplication. Création des boucle for pour le multiplication étape par étape. |
| Tests | 2 | Vendredi : Création de la grille de test sur Excel |
| Création : Méthode calcul | 2 | Vendredi : Finition de la fonction de multiplication |
| Création : Méthode calcul | 2 | Vendredi : Création de la fonction de division. |
| Création : Méthode calcul | 3 | Vendredi : Suite de la méthode de division. > Ma fonction imaginée ne fonctionne qu'avec les nombres décimaux. > Je vais réfléchir sur une nouvelle façon de convertir les données |
| Documentation | 5 | > Mise à jour de la fonction de division dans le rapport, et analyse d'une meilleure méthode de division |
| Journal de travail | 1 | Vendredi : Mise à jour du journal de travail |
| Documentation | 2 | Vendredi : Mise à jour des structogrammes de la méthode calcul, addition, BCD |
| Analyse | 2 | Vendredi : Rendez-vous avec M.Zahn > discussion sur l'avancement et l'état actuel du projet > Discussion sur le futur du projet |
| Documentation | 2 | Vendredi : Mise à jour du structogramme affichage avec la nouvelle méthode, et mise à jour du chapitre du rapport |
| Documentation | 4 | Vendredi : Mise à jour de la partie réalisation avec les paragraphe sur les nouvelles fonctions et sur les nombres à virgule flottante |
| Journal de travail | 2 | Vendredi : Mise à jour du journal de travail avec les liens |
| Total semaine | 93 | Max. 93 |

| Semaine 4 | | Date: |
|-------------------------------|----------------|---|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Journal de travail | 1 | Lundi : Mise à jour du planing et du JDT avec les dates pour les semaines |
| Documentation | 5 | Lundi : Mise à jour de la documentation dans la partie réalisation |
| Création : Méthode conversion | 4 | Lundi : recherche sur une possibilité de convertir les nombres à virgule flottante. |
| Documentation | 2 | Lundi : mise à jour de la documentation sur le point 4.2 modification |
| Création : débogage | 6 | Mercredi : Mise à jour des nom des variables dans la form calculator et mise à jour des commentaires du code |
| Création : Méthode conversion | 6 | Mercredi : Pour la conversion des nombres à virgule, lecture de la documentation du parseInt32 et exemples avec un site |
| Documentation | 3 | Mercredi : Mise à jour de la partie réalisation (4.1) avec les screenshot pour les différentes fonctions |
| Création : Méthode Affichage | 2 | Mercredi : Création de la fonction pour placer les valeurs, résultat et retenue dans les différenttextbox |
| Documentation | 4 | Mercredi : Mise à jour de la partie réalisation (4.1) avec les screenshot pour les fonctions convertToAll, Addvalue |
| Création : Méthode conversion | 6 | Mercredi : Analyse de la fonction IntParse et analyse du site pour pouvoir convertir des nombres à virgule |
| Absence - Imprévis | 54 | Week-end de l'ascension |
| Total semaine | 93 | Max. 93 |

| Semaine 5 | | Date: |
|-------------------------------|----------------|--|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Création : débogage | 2 | Lundi : Mise à jour du code et des commentaires |
| Création : Méthode Affichage | 4 | Lundi : Mise en place de la fonction showResult, permettant d'afficher les elements dans le pannel |
| Documentation | 6 | Lundi : Mise à jour de la documentation, sur le point réalisation. |
| Création : Méthode conversion | 6 | Mercredi : Amélioration de la méthode de multiplication pour réduire le nombre de ligne de code. |
| Création : Méthode Affichage | 3 | Mercredi : finition de l'affichage des résultats des opérations |
| Documentation | 2 | Mercredi : mise à jour de la partie sur l'affichage et sur les multiplications |
| Journal de travail | 1 | Mercredi : mise à jour du journal de travail et envoi du mail aux experts |
| Absence - Imprévis | 12 | Mercredi : Examen ECG 31.05.2017 |
| Création : Méthode calcul | 2 | Jeudi : Correction de la fonction sbstractValue, en y intégrant un tableau de char pour les retenues |
| Tests | 5 | Jeudi : Demarrage des tests, plusieurs soucis détecter à corriger |
| Création : débogage | 4 | Jeudi : Mise à jour des divers bugs repérer dans la phase de tests |
| Absence - Imprévis | 1 | Jeudi : Séance de classe avec le prof. Infos concernant l'après TPI |
| Documentation | 3 | Jeudi : Intégration de la partie tests dans la documentation |
| Création : Méthode calcul | 2 | Jeudi : Intégrations pour la solutions du bug de contrôle des valeurs dans la form calculator |
| Documentation | 6 | Jeudi : Mise à jour de la partie réalisations avec les nouveautés intégrées durant le debug |
| Tests | 1 | Jeudi : petits tests des divers elements corrigés le matin |
| Documentation | 3 | Jeudi : Correction de la documentations entre les elements actuelle et ceux rajouté durant le debug. |
| Documentation | 3 | Jeudi : Mise à jour de la doc avec les élément du jour (screenshot des fonctions et descriptions) |
| Création : Méthode calcul | 9 | Vendredi : Finition de la méthode de conversion, et correction de petit bugs découvert plus tot |
| Documentation | 6 | Vendredi : Finition de la doc sur le point 5 et ajout des annexes |
| Documentation | 6 | Vendredi : Création du manuel utilisateur |
| Documentation | 6 | Vendredi : Rapport : Création des bilans |
| Total semaine | 93 | Max. 93 |

| Semaine 6 | | Date: |
|--------------------|----------------|---|
| Tâche | Durée [1/4 h.] | Explications: qu'est-ce qui se fait et comment ? |
| Absence - Imprévis | 12 | Pentecôte 5.06.2017 |
| Documentation | 3 | Finition du rapport, (ajout des annexes) |
| Documentation | 9 | Impression des rapport, contrôle qualité, mise sous plis, envoi du dernier mail |
| Total semaine | 24 | Max. 93 |