TD PRG1011 – Codage en Binaire – Représentation des entiers – Opérations en binaire et opérations bit à bit.

I. Exercice de la présentation PRG1011-Codage v1.0

1. Combien faut-il de bits pour représenter les valeurs entières suivantes ?

Valeur décimale	Nombre minimal de bits pour la représentation binaire	Type à choisir de préférence, en C
8		
13		
15		
125		
32 000		
66 700		
100 000 000 000		

2. Donner la puissance de 2 immédiatement inférieure ou égale aux valeurs suivantes :

Valeur décimale	puissance de 2 immédiatement inférieure ou égale
32 921	
153	
25	
9	
1	

En déduire la représentation binaire sur 16 bits de la valeur 32921 :

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

32921 =

3. Répéter la division euclidienne par 2 de la valeur 32921 jusqu'à ce que ça ne soit plus possible :

En déduire la représentation binaire sur 16 bits de la valeur 32921 :

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

32921 =

4. Donner la représentation binaire en "complément à un" sur 8 bits des valeurs suivantes :

Valeur	Valeur binaire sur 8 bits en complément à un
10_{10}	
0120_{8}	
10100000_{2}	
10_{16}	
FF_{16}	

5. Donner la représentation binaire en "complément à deux" sur 8 bits des valeurs suivantes :

Valeur	Valeur binaire sur 8 bits en complément à deux
-10_{10}	
-128_{10}	
-1_{10}	
-63_{10}	
-127_{10}	

6. **A quelles opérations arithmétiques correspondent les expressions en C suivantes :**

Valeur	Opération réalisée	Valeur binaire sur 8 bits
10_{10} >> 1		
128_{10} >> 2		
1_{10} << 1		
63_{10} << 2		
128_{10}<< 1		

7. **Réaliser les opérations suivantes en binaire sur 8 bits (poser l'opération en binaire)**

Opération	Valeur binaire du résultat sur 8 bits
50 / 2	
25 * 4	
80 - 80	
-128 + 127	
-30 + 5	
127 - 128	

II. Calcul binaire et langage C

Cet exercice peut avoir été déjà proposé sous la forme d'un TDPRG1010 calculatrice binaire.

Dans une solution TDPRG1011 Créer un projet nommé Starter, ajoutez-y le code du fichier **TDPRG1011-Code a etudier.c** que vous devrez comprendre et compléter

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <math.h>

// Un peu de code C pour vous donner des pistes sur la manière de manipuler et de
// comprendre la nature binaire des données en informatique

int main() {
	// gestion des caractères accentués dans la console.
	setlocale(LC_ALL, "fr-FR");

	// voici une possibilité pour mettre la valeur décimale 65 dans la variable valeur1
	// noter le préfixe 0b qui indique qu'il s'agit d'une valeur directement
	// représentée en binaire dans votre code source.

	unsigned char valeur1 = 0b01000001;

	// Ce qui équivaut à écrire :
	// unsigned char valeur1 = 65; Vérifiez cela à la calculatrice et sur le papier !
	// Le compilateur n'a pas d'autre choix que de faire une traduction en binaire de cette valeur 65.

	// Notez qu'en C, le type ‘char’ correspond à une valeur entière codée sur 8 bits.
	// ce qui est aussi parfaitement adapté pour stocker le code d'un caractère ASCII, d'où le type char ..

	// En ASCII, le code du caractère A est 65.

	// pour obtenir de l'aide sur une fonction, sélectionnez le nom de cette fonction et
	// appuyez sur la touche F1 en étant connecté à Internet.

	printf(" la variable valeur1 contient la valeur %d et représente le caractère %c \n", valeur1, valeur1);


	// Ecrire le code qui affiche la valeur binaire de valeur1

	unsigned char tmp = valeur1; // recopie de la valeur de la variable valeur1 dans la variable tmp



	// étudiez ce code et expliquez comment il permet de faire un affichage de valeurs en binaire,
	// ce que ne permet pas la commande printf
	// A quoi sert l'opérateur / ?
	// A quoi sert l'opérateur % ?
	// Vérifiez que tmp = tmp % 128 peut aussi s'écrire tmp %= 128

	printf("\n Ce qui donne en binaire (version 1 :\n");

	printf("%u", tmp / 128); tmp %= 128;
	printf("%u", tmp / 64); tmp %= 64;
	printf("%u", tmp / 32); tmp %= 32;
	printf("%u", tmp / 16); tmp %= 16;
	printf("%u", tmp / 8); tmp %= 8;
	printf("%u", tmp / 4); tmp %= 4;
	printf("%u", tmp / 2); tmp %= 2;
	printf("%u", tmp % 2);
	printf("\n");
```

```
55.
56.       // autre méthode qui utilise l'opérateur de décalage de bits
57.       tmp = valeur1; // on replace la valeur 65 d'origine dans tmp
58.
59.       printf("\n Ce qui donne en binaire (version 2) :\n");
60.
61.       // expliquez le principe de fonctionnement de l'expression : variable = condition ? valeur1 : valeur2
62.       // A quoi sert l'opérateur bit à bit '&' de dans ce contexte ?
63.       // A quoi sert l'opérateur << ?
64.
65.
66.       printf("%u", (tmp & 128) == 0 ? 0 : 1);
67.       tmp <<= 1;
68.       printf("%u", (tmp & 128) == 0 ? 0 : 1);
69.       tmp <<= 1;
70.
71.       // Ecrire la suite afin de traiter l'ensemble des bits
72.
73.
74.   // Qu'est-ce que le bit de poids fort, MSB ?
75.
76.           // calcul du poids représentée par le bit de poids fort (MSB) dans un octet :
77.
78.   // Quelle valeur est fournie par sizeof pour le type ou la variable entre parenthèses ?
79.
80.       int nbOctets = sizeof(unsigned char);
81.       int nbBits = nbOctets * 8;
82.
83.       int valeurMSB = (int)pow(2.0, (double) (nbBits) - 1);
84.       printf("\n Le poids du MSB avec pow dans un octet est %d", valeurMSB);
85.
86.       // calcul de la valeur du bit de poids fort - version 2
87.
88.       valeurMSB = 1 << (nbBits - 1); // Que se passe t-il ici ?
89.
90.       printf("\n La valeur du MSB avec décalage à gauche de 1 , %d fois, est %d\n", nbBits - 1, valeurMSB);
91.
92.       // mettez un point d'arrêt et
93.       // déroulez ce programme pas-à-pas en observant la valeur des variables locales pour mieux comprendre
94.       printf("\n Ce qui donne en binaire (version 3) :\n");
95.       tmp = valeur1;
96.       for (int i = 0; i < nbBits; i++) {
97.           printf("%u", (tmp & valeurMSB) == 0 ? 0 : 1);
98.           tmp <<= 1;
99.       }
100.      // OK, ceci est une boucle POUR ! :-)
101.      system("pause");
102.
103.      return(EXIT_SUCCESS);
104.  }
```

III. **Challenge** : Développer une calculatrice qui fait du calcul binaire

Cet exercice peut avoir été déjà proposé sous la forme d'un TDPRG1010 calculatrice binaire.

Description : Il s'agit dans un premier temps, d'étudier le comportement de la calculatrice fournie avec Windows dans le mode développeur et de reproduire ce comportement dans votre propre programme dans la console Windows en mode texte.

Méthode : Chacun met en place son propre projet, il n'est pas envisageable de récupérer le code d'une autre personne. Cependant, il est possible de travailler en s'échangeant des informations et de solliciter les compétences d'un autre étudiant. Il s'agit donc de manipuler la calculatrice en essayant de prédire le résultat et l'affichage obtenu pour différentes opérations. Il faut décrire l'utilité de chaque bouton de la calculatrice. Pour vérifier la bonne compréhension du principe de calcul et d'affichage des nombres en binaire, vous devrez faire des petits essais en écrivant des morceaux de code en langage C. Avant de se lancer dans le codage de la calculatrice, il faudra avoir mis à plat sur le papier les principes des différentes opérations en binaire et décrit par un algorithme les différents blocs fonctionnels de la calculatrice. Ne cherchez pas à avoir une calculatrice complète dès le départ, réfléchissez déjà à réaliser l'addition deux nombres entiers positifs en base 10 et à afficher le résultat sous une forme binaire pour différentes capacités de codage (8 – 16 – 32 – 64 bits).

Ressources :

Fonctions de positionnement et d'affichage d'un caractère sur la console et fonction de lecture d'un caractère entré au clavier fournies dans ConsoleTools.zip. Ce fichier est disponible dans le dépôt de fichier Teams ou sur Github

Manuel de C, ressources Internet et les étudiants ayant acquis des compétences sur le sujet du langage C ou du binaire, ainsi que les éléments de cours.

Contrainte :

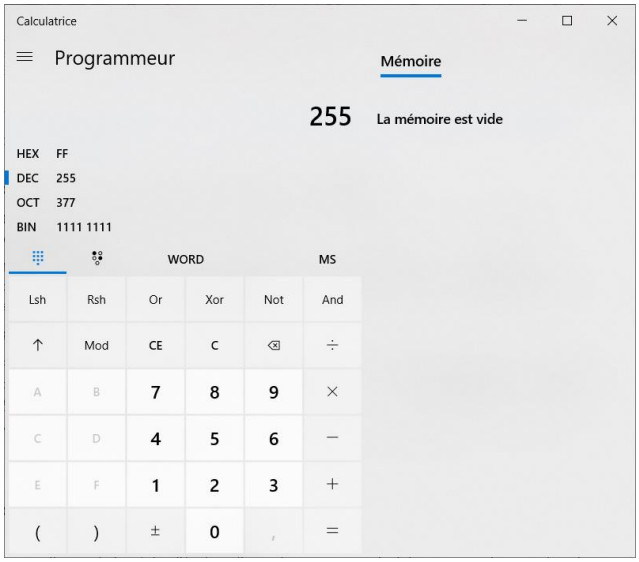
- Le programme est écrit en langage C
- Il respecte les règles d'écriture d'un code « propre », indentation, commentaires pertinents, noms de variables judicieux.
- La mise au point se fait en utilisant le débogueur de Visual Studio et en corrigeant toutes les alertes et erreurs de compilation.

Outils :

- Visual Studio
- Code partagé sur un dépôt privé GitHub.

Temps alloué :

- Séances de TD + travail **off**
- Amélioration du programme au fil de l'année en fonction de l'acquisition des nouvelles connaissances algorithmiques et de langage.


Calculatrice
Programmeur
Mémoire
255
La mémoire est vide
HEX FF
DEC 255
OCT 377
BIN 1111 1111
WORD
MS
Lsh Rsh Or Xor Not And
Mod CE C
A B 7 8 9
C D 4 5 6
E F 1 2 3
( ) ± 0 , =