TP C no 1

Objectifs:

Comprendre et mettre en œuvre la notion de représentation des données en langage C.

Maîtrise du langage C

- Ecriture et évaluation des expressions
- Nommage des variables en langage C
- Utilisation de constantes **#define** ou **const**
- Structure du code source pour un programme constitué d'un fichier source unique.
- Indentation du code et remise en forme automatique.
- Utilisations des commentaires.
- Mise en place de structures conditionnelles et itératives (**if** - **for** – **while**).

Mise au point d'un programme avec Visual Studio

- Utilisation du débogueur et des points d'arrêts pour tracer le code
- Débogage de la valeur des variables
- Utilisation de la documentation en ligne

Liens :

https://docs.microsoft.com/en-us/cpp/c-language

Modalités :

Le TP no 1 est disponible sous la forme d'un dépôt Github distribué de manière privée et nominative à l'ensemble des étudiants de la classe. Utilisez le lien communiqué sur Team pour activer votre dépôt en prenant soin d'utiliser votre adresse universitaire en junia.com.

Le TP est fourni sous la forme d'une solution Visual Studio 2019.

Nous vous invitons à réaliser un 'commit' après chaque exercice et à pousser l'ensemble du travail sur GitHub dès la fin de séance.

Exercice 1

Rappel :

```
// le type 'char' caractérise un entier positif codé sur 8 bits compris entre -128 et +127,
// le bit de poids fort sert au codage du signe.
// un 'unsigned char' pourra donc représenter un nombre positif compris en 0 et 255,
// soit 256 valeurs.
// La représentation en hexadécimale de 255 est FF (notée 0xFF)
// Le type booléen n'existe pas en C90, une valeur non nulle
// dans une expression logique est comprise comme 'VRAI'.

// La prise en charge du type booléen peut se faire en ajoutant une directive #include <stdbool.h>
```

Dans le projet Exo1. Le fichier **source** à utiliser se nomme **main.c**

```
#define _CRT_SECURE_NO_WARNINGS

// inclusion des fichiers d'entêtes (header) d'extension .h


#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <float.h>


// définition de type

typedef unsigned char byte;

int main() { // accolade ouvrante = début de bloc d'instructions
        // Déclaration de variables réservées au main(), et initialisations facultatives
        char OneLetter = 'A';
        char AnOtherLetter = 66;
        byte toto = 0;
        unsigned char VerySmallCounter = 255;
        short SmallCounter = 32767;
        int Counter = 0;
        long LargeCounter = 2147483647;
        unsigned long FullRangeLargeCounter = 4294967295;
        int ChienDeGarde = 0;
        float Dim1 = 1.414F;
        double Dim2 = .5;
        bool TestResult = 1;
        // Déclaration de constante
        const double Pi = 3.14159265;
        // les instructions du programme principal : main()
        /********** Debut de programme ****************/
        printf("\n");
        printf("exemple d'affichage d'un caractere %c, codage sur %d octets\n", OneLetter, sizeof(OneLetter));
        printf("exemple d'affichage d'un entier court %d, codage sur %d octets\n", Counter, sizeof(Counter));
        /********** Fin de programme ****************/

        return(EXIT_SUCCESS);
} // accolade fermante = fin de bloc d'instructions
```

- Compléter le code par des appels à la fonction d'affichage **printf** pour afficher la valeur de chacune des variables dans un format correct. Attention, le choix du filtre **%**, impacte le résultat car il réalise une conversion de la donnée avant son affichage dans le format souhaité. Donc, ce qui s'affiche est ce qui a été demandé, et pas nécessairement la donnée dans son format d'origine.
- Donner pour chaque type, le nombre d'octets utilisés pour le codage, et indiquer quelle valeur minimum et quelle valeur maximum (dynamique) on peut représenter avec une variable qui aurait ce type. Donner la réponse sous la forme de commentaires dans votre fichier source.

Références :

Noter que le manuel de référence du langage indique dans quel fichier *.h* une fonction est déclarée. Il faut donc ajouter une directive **#include <stdio.h>** en début de code source.
En effet, d'après l'information suivante https://docs.microsoft.com/fr-fr/cpp/c-runtime-library/reference/printf-printf-l-wprintf-wprintf-l , on découvre que la fonction **printf** requière **stdio.h** (section Requirements).

Liens d'aide pour les **formats d'affichage** avec **printf** :

https://docs.microsoft.com/fr-fr/cpp/c-runtime-library/format-specification-syntax-printf-and-wprintf-functions

Quels sont les types qui ne sont pas natifs* du langage C (version 90), à partir de quels autres types sont-ils définis ici ?
*natif : Qui a été créé de façon originale dans le langage.
La taille réellement occupée pour le codage de ces types de données peut varier en fonction du compilateur et de l'architecture de la machine.

Type	Taille en octets	Valeur Minimum	Valeur Maximum
char			
byte			
unsigned char			
short int			
int			
long			
unsigned long			
float			
double			

Rappel :

```
// Une variable est caractérisée par son nom (ou identifiant), par le type de donnée quelle peut
// recevoir et un emplacement en mémoire où elle stocke ses données.
// L'emplacement des données en mémoire s'appelle l'ADRESSE
// En C, on fait référence à l'adresse d'une variable à l'aide du caractère '&'
//
// Par exemple, si on déclare la variable entière Nombre :
// int nombre;
// nombre = 12;
// alors &nombre est l'adresse en mémoire de cette variable qui contient la valeur 12.
// L'adresse est une valeur entière codée sur 64 bits dans un environnement 64 bits.
// Il est probable que deux variables déclarées l'une à la suite de l'autre aient des adresses
// mémoire très proches.
```

- Compléter le code en déclarant en une seule instruction les trois variables A, B et C de type *int*. Les variables A, B et C sont des variables déclarées **localement** au main(), donc à l'intérieur des accolades du main() { }.
- Donner une valeur initiale aux variables A, B et C en les assignant respectivement des valeurs constantes 1, 2 et 3.

- En une seule instruction, afficher les valeurs en décimal et l'adresse mémoire en notation hexadécimale de ces trois variables, utiliser '**\n**' pour afficher les valeurs sur trois lignes différentes. Que penser de ces trois adresses quand on les compare ?
  Remarque: l'option de formatage pour afficher une valeur entière en hexadécimale est %X, %p est également utilisable pour afficher une adresse en hexadécimal.

Rappel :

```
// Les ordinateurs sont des machines conçues pour faire des opérations répétitives, très rapidement.
// La manière dont se déroulent ces opérations est contrôlées par un algorithme.
// Les boucles sont des algorithmes qui exécutent un groupe d'instructions un certain nombre de fois.
// Il se peut que l'on connaisse le nombre d'itérations (boucles) avant de commencer le traitement, ou
// que l'on réalise des itérations jusqu'à ce qu'un événement se produise. Cela dépend du contexte
// et des informations dont on dispose.
// Par exemple, monter les 20 marches de cet escalier et s'arrêter, ou, monter les marches jusqu'à
// atteindre le 1er étage et s'arrêter.
// Le premier cas est le plus simple à traduire, on connait une valeur de début et une valeur de fin.
// De plus, l'incrément, c'est-à-dire la valeur qu'on ajoute pour passer au suivant est ici de 1
// (on monte une marche à la fois).
// Idéalement, on implémente (traduction dans le langage de programmation) une boucle FOR :
// Ce qui donne en pseudo-langage,
// Pour noMarche allant de 1 à 100
// faire
//      MonterMarche
// fait.
//
// et en langage C,
// for (noMarche=1 ; noMarche <= 100 ; noMarche++ )
//      {
//      MonterMarche();
//      }
//
// noMarche = 1 : Donne une valeur initiale à la variable qui indique le numéro de marche
// noMarche <= 100 : Condition à vérifier pour continuer à boucler
// noMarche++ : incrémenter la variable noMarche de 1 après avoir utilisé sa valeur.
// Remarque : ne pas confondre a=1 avec a==1
// dans le 1 er cas ‘a’ prend la valeur 1, dans le second il s'agit d'une comparaison.
```

Exercice 2

a- Donnez le code qui permet de calculer la valeur de Pi en utilisant la relation ArcTang(1) = Pi/4 et le développement limité de la fonction Arctangente :

Pi = 4 * (1 - 1/3 + 1/5 - 1/7 + 1/9 -1/11 + ...)

On arrêtera de calculer les termes de l'addition quand leur valeur sera inférieure à 10E-10.
Afficher le résultat avec 10 décimales.

b- Affichez la table des sinus des angles allant de 0 à 90 degrés en faisant varier l'angle de 10 degrés en 10 degrés.
Quel fichier d'entêtes .h doit-on utiliser ?
Définissez la valeur de PI à l'aide d'une constante en lui donnant la valeur 3.1415926535898
Afficher la valeur de l'angle en degrés et la valeur du sinus sur la même ligne en alignant les valeurs

Exercice 3

Compléter le code de main.c afin d'afficher une table de caractères ASCII étendue
La table est organisée suivant 8 colonnes de 32 lignes.
Chaque colonne présente le code en décimal aligné à gauche sur deux caractères,
Le code caractère en hexadécimal Majuscule sur deux caractères avec un zéro en tête quand il n'y a qu'un seul digit.
Le caractère lui-même qui sera remplacé par un caractère espace lorsqu'un code spécial perturbe l'affichage.