	Développement logiciel ***Structures de boucles***	**BTS SN IR** ***TP5***

Configuration du projet :

- Nom du projet: ***struct_boucle***
- Nom du fichier source: ***En fonction de l'exercice***

Reprise de l'exercice n°3 du cours associé avec N=7.

Les corps de programme seront séparés par des commentaires.

1. *Mise en œuvre de l'instruction while():* *boucle.cpp*

```
PROGRAMME pairs
VAR       N, CPT : entier

DEBUT
          Afficher « veuillez saisir un chiffre. »
          N <- le nombre saisi
          CPT <- 0
          TantQue CPT<=N
                    faire
                    Afficher CPT
                    CPT<-CPT + 2
          FinTantQue
FIN
```

```
#include "stdafx.h"
#include <iostream>

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    int N, CPT;

    cout << "Veuillez saisir un chiffre n=   ";
    cin >> N;

    // Boucle while
    CPT = 0;
    while (CPT <= N)
    {
        cout << "valeur paire......   " << CPT << endl;
        CPT = CPT + 2;
    }

    // Boucle do...while

    // Boucle for...

    system("PAUSE");

    return 0;
```

- Donner l'organigramme répondant au problème et le traduire sous VISIO4.

Le code relatif à l'exercice 3 est donné ci-contre :

- Ecrire de code, et tester ce programme.

Comme dans les structures de choix, les conditions de boucle se comportent comme des booléens.

- Exécuter le programme en pas à pas principal, et compléter le tableau ci-dessous en précisant comment le compilateur interprète les instructions proposées.

Instructions	*Itération n°*	*Itération n°*	*Itération n°*	*Itération n°*	*Itération n°*
`while (CPT <= N)`					
`cout << "valeur paire.....`					
`CPT = CPT + 2;`					

2. ***Mise en œuvre de l'instruction do… while() :***

- Proposer un organigramme intégrant une instruction *do…while().*
- A la suite du programme précédent *(après le commentaire // Boucle do while)*, coder et tester ce nouveau bout de programme.
- Exécuter le programme en pas à pas principal et constater le bon fonctionnement.

3. ***Mise en œuvre de l'instruction for(…) :***

- Précisez les 3 expressions constituant l'instruction *for(….).*
    - *l'initialisation de l'itération.*
    - *la condition de contrôle.*
    - *L'instruction d'évolution.*

- A la suite du programme précédent *(après le commentaire // Boucle for)*, coder et tester ce nouveau bout de programme.
- Exécuter le programme en pas à pas principal et constater le bon fonctionnement.

4. ***Utilisation de tests combinés comme condition de contrôle de boucle:***

On souhaite maintenant que le programme affiche tous les nombres pairs inferieurs quelque soit le nombre entré N, *sauf lorsque N=5*. Dans ce cas aucun affichage ne doit se produire, et la boucle ne doit pas être exécutée.
On considère cette problématique uniquement pour la structure de programmation while().

- En utilisant des tests combinés (utilisation des opérateurs &&, ||..), adapter le 1er programme de façon à prendre en compte cette nouvelle contrainte.
- Programmer et tester.

5. ***Exercice n°4 du cours:*** ***saisie.cpp***

On souhaite écrire un programme qui demande à l'utilisateur un nombre compris entre 1 et 3 jusqu'à ce que la réponse convienne. L'algorithme est celui vu en cours et correspond à l'exercice n°4.

- Programmer, commenter et tester.

6. ***Conversion binaire-décimal :*** ***convb_d.cpp***

On souhaite réaliser un programme qui effectue la conversion binaire/décimale d'une constante de type char non signé. La valeur de cette constante sera définie en base héxadécimale. Pour la résolution de l'exercice, celle-ci devra néanmoins être considérée à partir de sa valeur binaire.

Les variables à utiliser sont les suivantes :

- N : Nombre binaire à convertir en décimal.
- Pds : Permet « d'isoler » le poids décimal du bit à traiter dans la variable N.
    - Au départ Pds=0x80 (poids fort)

- Res : Résultat final
- Mem : A utiliser pour des calculs intermédiaires si-besoin.

L'organigramme « général » du programme à coder est donné ci-après :

- Coder la solution et tester.