Séance N°1

Quelques mots de vocabulaire pour commencer à prendre en main son poste informatique :

Ordinateur, clavier, écran, souris, mémoire (vive/morte), disque dur, processeur
Logiciel, traitement de données, algorithme, instruction, langage de programmation, C, C++, etc
Touches clavier : « Entrée » ou «Return », « Echappe » ou «Escape», majuscule, minuscule, etc

Utilisation de Microsoft Visual Studio.NET

1) Lancer le logiciel **Visual Studio** à partir de la barre d'état de Windows : cliquer sur
« Demarrer » -> « Tous les programmes » -> « Microsoft Visual Studio 2013 »

2) Attendre que la configuration de l'environnement s'achève. Lors de la première connexion sera demandée l'application la plus utilisée. Choisir « Visual C++ ».

3) Une fois la fenêtre de Visual Studio ouverte, cliquer sur l'Option « Fichier » du menu principal, sous-option « Nouveau » -> « Projet ».

4) Une fois la fenêtre « **Nouveau projet** » ouverte, choisir,
« Types de projet » : Projet Visual C++
« Modèle » : Application console Win32
« Nom » : donner un nom à votre choix, ex :TD1
« Emplacement » : C:\temp.
Valider par le bouton « OK ».

5) Dans la fenêtre « **Assistant application Win 32** », cliquer sur « Paramètres de l'application », et sélectionner le bouton radio « Application console », et cocher dans « Options supplémentaires » la case « Projet vide ». Cliquer sur bouton « Terminer ».

6) Sur la droite de l'écran, une fenêtre « **Explorateur de solution** » doit s'ouvrir (si ce n'est pas le cas aller dans l'option « Affichage » du menu principal, et activer la sous-option « Explorateur de solutions »). Clic bouton droit sur le dossier « Fichiers source » du projet. Option « Ajouter » puis sous-option « Ajouter un nouvel élément ».

7) S'ouvre une fenêtre « **Ajouter un nouvel élément** » où dans la Catégorie : Visual C++ on choisit le « Modèle » : **Fichier C++**, en lui donnant un « Nom » (choisi par l'utilisateur, par exemple le même que le nom du projet), puis clic sur le bouton « Ajouter ».

8) S'ouvre alors une fenêtre d' « édition » de fichier, avec un curseur clignotant. Copier le programme exemple ci-dessous :

```
#include <iostream>
using namespace std;
void main()
{
cout<<"ceci est un texte";
}
```

Remarque :
Un programme C++ contient toujours une **unique fonction principale** dont le nom est *« void main() »*. Cette fonction principale est précédée d'instructions « #include » qui précisent quelles bibliothèques seront utilisées. Par exemple, la bibliothèque <iostream> contient les fonctions d'« entrée » (saisie au clavier : *cin*) et « sortie » (affichage à l'écran : *cout*). Le **corps** ou **bloc principal** du programme est délimité par les symboles de accolades: « { » et « } ». Il contient un ensemble d'instructions séparées les unes des autres par le **séparateur** point-virgule « ; » .

9) Pour **compiler le programme** (c'est-à-dire le traduire en langage machine) :
Option « Générer» du Menu principal, sous option « Générer la solution ».

10) Pour **exécuter le programme** :
Option « Déboguer » du Menu principal, sous-option « Exécuter sans débogage ».
Le résultat de l'exécution apparaît dans une nouvelle fenêtre (fenêtre « console »).

11) Pour **sauvegarder un projet**

Deux solutions possibles :

- dans **votre espace de travail** sur le serveur du domaine « Etudiants » : l'avantage est que le projet sera alors accessible de n'importe quel poste de travail mais cela nécessite de passer par le réseau (connexion valide qui peut être lente) ;

- dans **le répertoire « Temp »** (seul accessible en écriture pour tous) sur le disque dur du PC utilisé : l'avantage est que l'on travaille alors en direct sans passer par le réseau, c'est plus rapide mais il faudra penser à sauvegarder ensuite le projet sur votre espace de travail ou une clé USB pour ne pas perdre le programme réalisé, car le contenu du répertoire « Temp » est régulièrement effacé.

Dans la fenêtre « Explorateur de solutions » :
Sauvegarde d'un fichier: « Fichier » -> « Enregistrer » du fichier cpp.
Sauvegarde d'un projet (ou solution) : « Fichier » -> « Enregistrer » de la solution sln.

Fonction d'AFFICHAGE : *cout<<constante ou variable*

Ex 1 - Affichage d'une constante "chaîne de caractères" (d'un texte) :

```
#include <iostream>
using namespace std;
void main()
{
cout<<"ceci est un texte";
}
```

Question : A quoi sert l'instruction #include <iostream> ?
Tester le en plaçant cette instruction en « commentaire » en la faisant précéder de « // » (double slash)

Ex 2 - Affichage d'une constante numérique :

```
#include <iostream>
using namespace std;
void main()
{
cout<<2<<endl;
}
```

Question : A quoi correspond « endl » ?

Fonction de SAISIE : *cin>>variable*

Ex 3 - Saisie d'un entier :

```
// saisie d'un entier
#include <iostream>
using namespace std;
void main()
{
int v;
cout<<"saisir un entier: ";
cin>>v;
cout<<"la valeur saisie est: "<<v<<endl;
}
```

Ex 4 - Faire un calcul simple :

```
// un calcul simple
#include <iostream>
using namespace std;
void main()
{
cout<<2+3<<endl;
}
```

DECLARATION, INITIALISATION, AFFECTATION des VARIABLES

Une **constante** a toujours la même valeur (ex : 3, "adcd"), lorsque le programme est exécuté plusieurs fois. Un programme utilise des **variables** dont la valeur peut changer au cours de l'exécution du programme. Elles occupent des zones de la mémoire centrale volatile (modifiée d'une exécution à l'autre). La taille de cet emplacement mémoire est déterminée par le type de la variable : int pour entier, float ou double pour réel en double précision, char pour un caractère, etc. Consulter les fichiers *climits.h* et *float.h* dans l' « Explorateur de solutions » / « Dépendances Externes » pour voir les capacités de stockage définies pour chaque type.

Une variable doit toujours être **déclarée** avant d'être utilisée, en précisant son type. L'instruction d'affectation permet d'**initialiser** une variable avec une valeur (*variable = valeur),* en respectant son type. Une variable peut être déclarée et **initialisée** dans la même instruction : int i = 0 ; est équivalent à int i; i = 0;

Ex 5 - Testez le programme suivant :

```
//des affectations
#include <iostream>
using namespace std;
void main()
{
int a;
a = 2;
int b;
b = 3;
int c;
c = a+b;
cout<<c<<endl;
}
```

Ex 6 - Puis celui-ci :

```
//avec une fonction
#include <iostream>
#include <cmath> // pour accéder à la fonction sinus
using namespace std;
void main()
{
double a; //double indique que a est un réel double précision
a = 0.7;
double b;
b = 3.14;
double c;
c = sin(a)+sin(b) ;
cout<<c<<endl;
}
```

Question : A sert l'instruction #include <cmath> ?

Remarque: Un nom de variable ne doit pas commencer par un chiffre, ne doit pas comporter d'espace, ni de caractères spéciaux (@,#,-,&,…). Consulter la liste des mots-clés du langage C++.

Blocs et portée des variables

Un **bloc** en C++ est délimité par les symboles de accolades: « { » et « } » (dits **séparateurs de blocs**). Un programme comprend toujours un bloc principal pour l'unique fonction « *void main()* », plus éventuellement des sous-blocs imbriqués. Dans un bloc, on peut définir ou déclarer des **variables locales**: une variable n'est accessible ou utilisable que dans le bloc où elle a été définie et dans les éventuels sous-blocs inclus dans ce bloc. On dit que la **portée** d'une variable est limitée à son bloc de définition ou déclaration. L'alignement des séparateurs de blocs « { » et « } » facilite la compréhension de la structure du programme.

Exemple :

```
void main()
{
int i;
for (i = 0; i<10; i++) // la variable i est connue de toute la fonction main
        {
        int j; // la variable j n'est connue que du sous-bloc
        ...;
        }
}
```

TRAVAIL PERSONNEL

Pour mettre en application ce que j'ai appris, je fais les exercices suivants :

Exercice 1
Réaliser un programme permettant de calculer la surface d'un rectangle.
La longueur et la largeur seront saisies par l'utilisateur et le résultat sera affiché à l'écran.

Exercice 2
1) Réaliser un programme permettant de
- Demander à l'utilisateur de saisir au clavier une valeur de température en degrés Celsius;
- Calculer ses équivalents en degrés Fahrenheit et en Kelvin (rappel: °F =5/9 °C + 32 et 0 K = -273.15°C)
- Afficher les 3 valeurs l'une en dessus de l'autre avec les unités.

2) Compléter le programme pour réaliser les conversions suivantes :
Fahrenheit vers degré Celsius et Kelvin
Kelvin vers degré Fahrenheit et degré Celsius

BILAN PERSONNEL

Ce que j'ai appris aujourd'hui :

- démarrer le logiciel VISUAL Studio.NET
- créer un projet en C++
- écrire un programme
- compiler un programme
- générer une solution
- afficher une phrase à l'écran
- saisir un nombre au clavier
- placer un commentaire dans un programme
- déclarer et initialiser des variables

Vocabulaire informatique :

- constante
- variable
- expression
- déclaration
- affectation
- nombre entier
- nombre réel, nombre décimal
- bloc, séparateur

Savoir répondre aux questions

Un programme peut-il contenir plusieurs fonctions principales « *main* » ?
Quelle est la portée d'une variable ?