TP 1 – IMEC4-U2

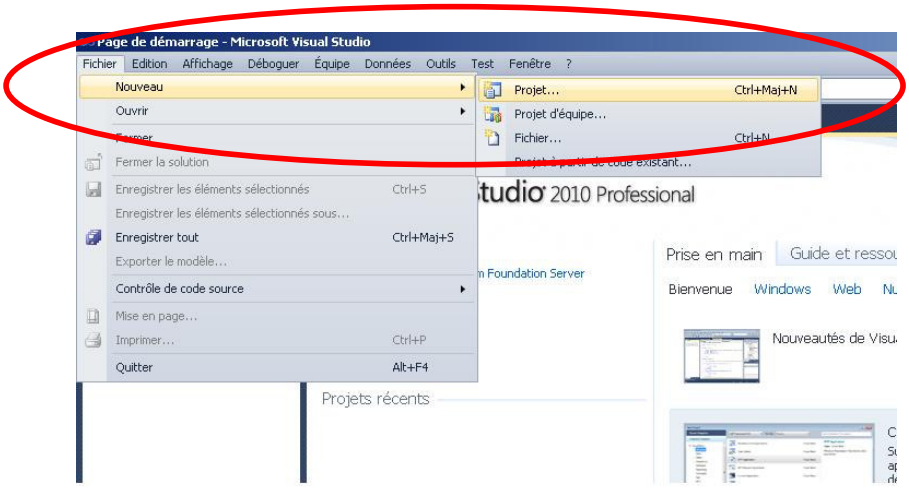
C++ Programming using Microsoft Visual Studio 2010

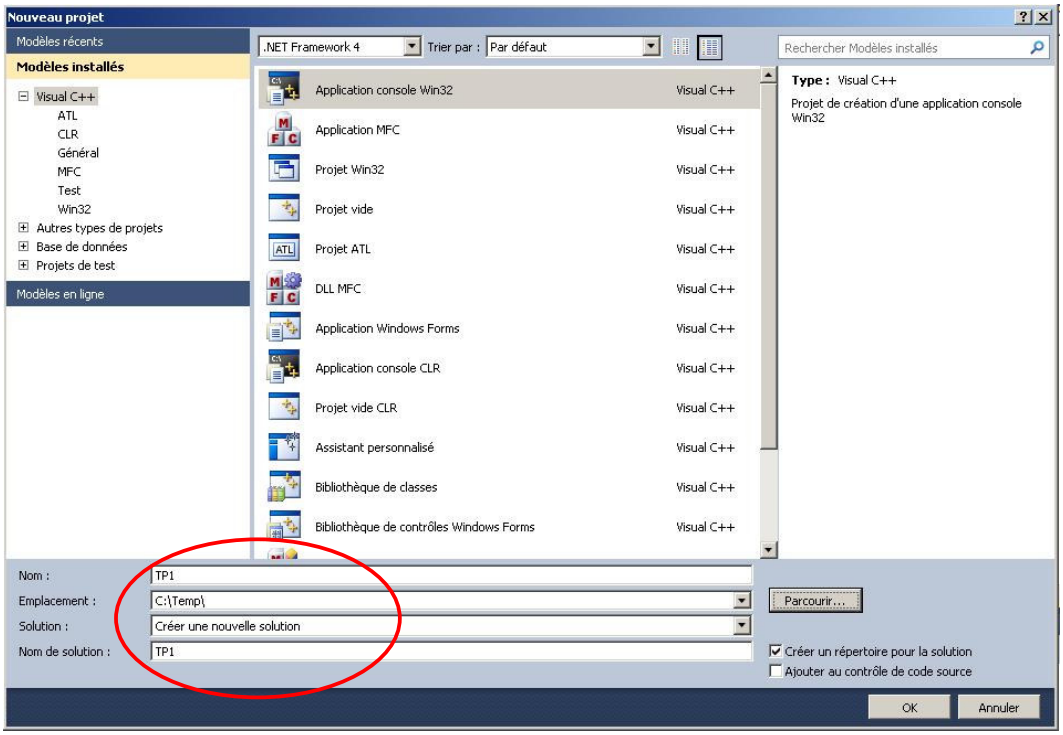
T. Grandpierre (office : 5253)

1. The first project

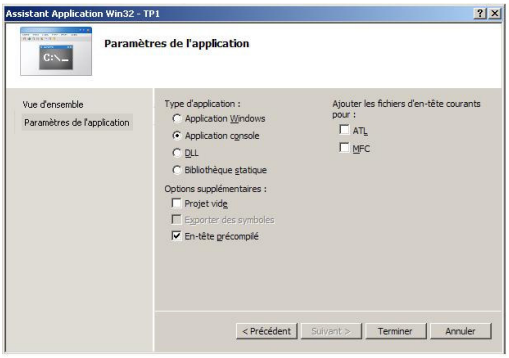
Inn the following you will learn how to create an executable program using Visual Studio 2010. At this point we will build a project in console mode: this program won't use the windows graphical interface (i.e. mouse, button etc.) but only interact within a DOS console windows.

Step 1 : Run Microsoft Visual 2010, then select "New Project" (or "Nouveau Projet" in French), then select "Win32" and "Application console win32", then give a name (Nom) for this project and select the directories (Emplacement) where to create this project. Finally click OK :

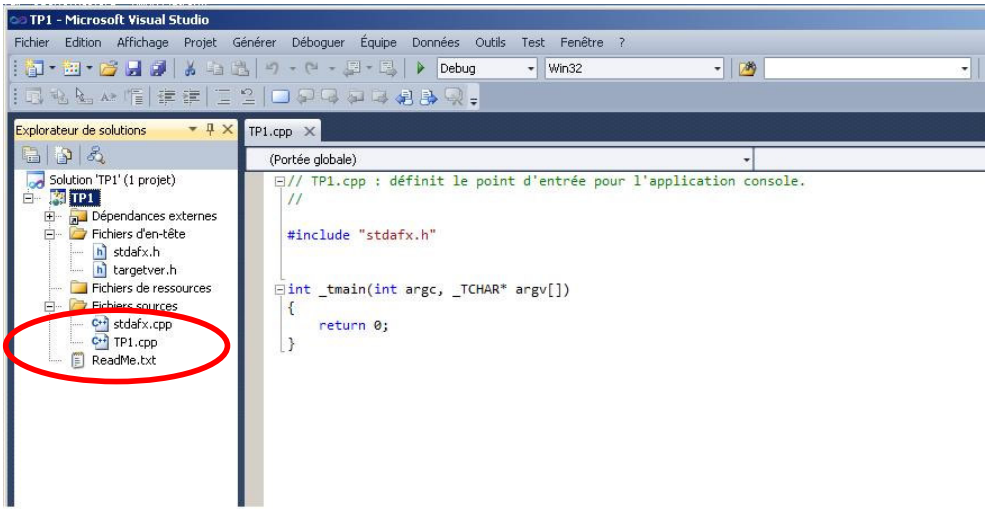




Then let the default parameter proposed by visual 2010 and click on finish (Terminer) :

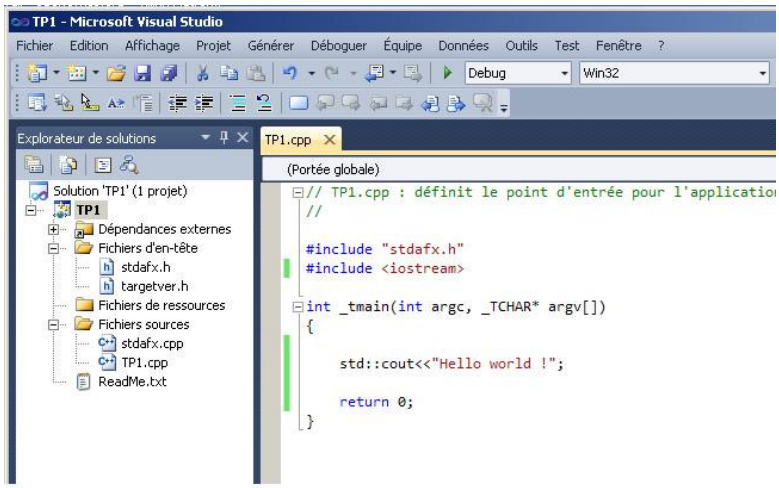


This will build a complete C++ project in console mode. It will also produce an example of program in the file TP1.cpp (left side of visual interface: project explorer) which is an empty main function (you can see the code in the right side): so it does nothing:

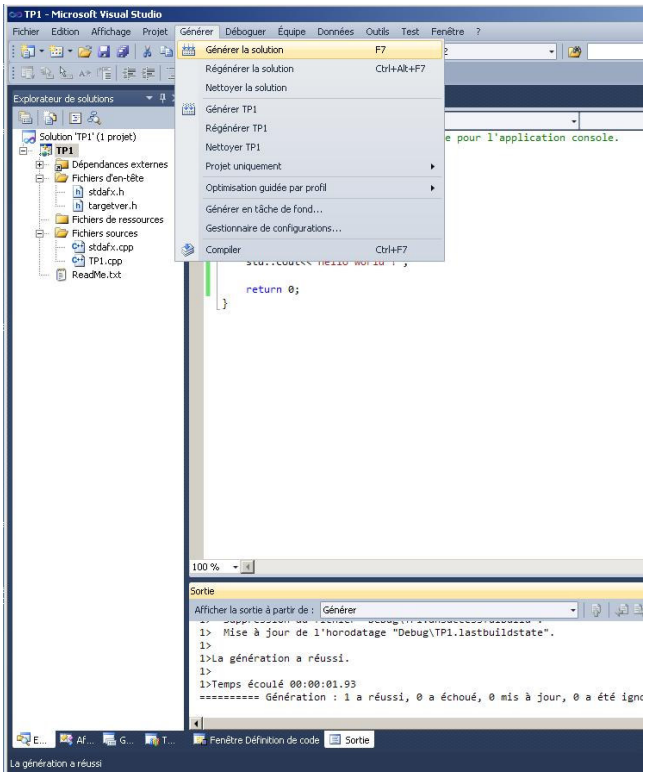


At this point you may compile it and execute it, but we will firstly add 2 lines of code in order to make your program doing something:

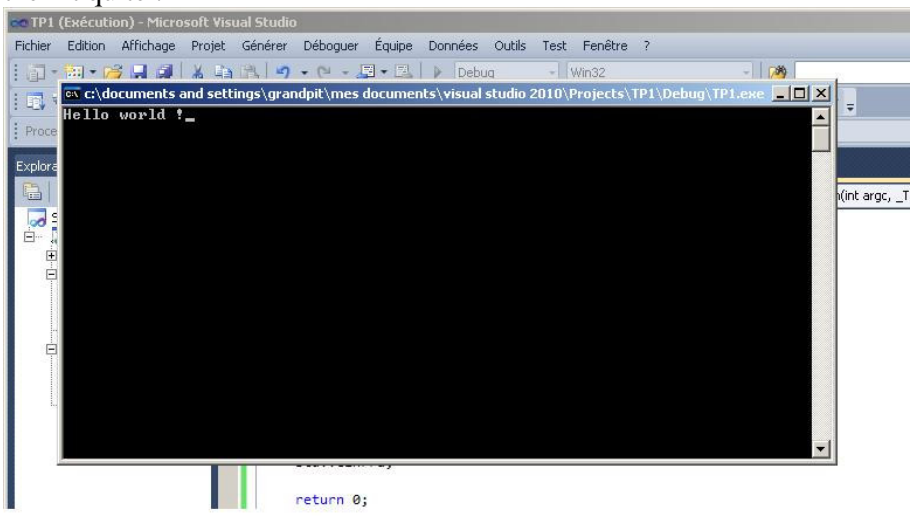
- After the declaration of the main function (which is _tmain in the current case), add the following line:
  std::cout<<"Hello world !"
- You must also add the following line after the #include statement :
  #include <iostream>



You can now compile this project using the menu item “Generer”->”Generer la solution”, and if everything is going right, you should get “Génération à réussi” on the bottom of visual windows. This means that the generation is successful.



You are now able to run this code in debug mode by clicking on the small green triangle or by selecting “Déboguer->démarrer le débogage”. You will obtain a DOS console windows which will quicly disappear because the program just execute what you wrote : it display Hello and then it quite :



In order to get the windows stay longer, I propose to add the following lines before the end of the main function:

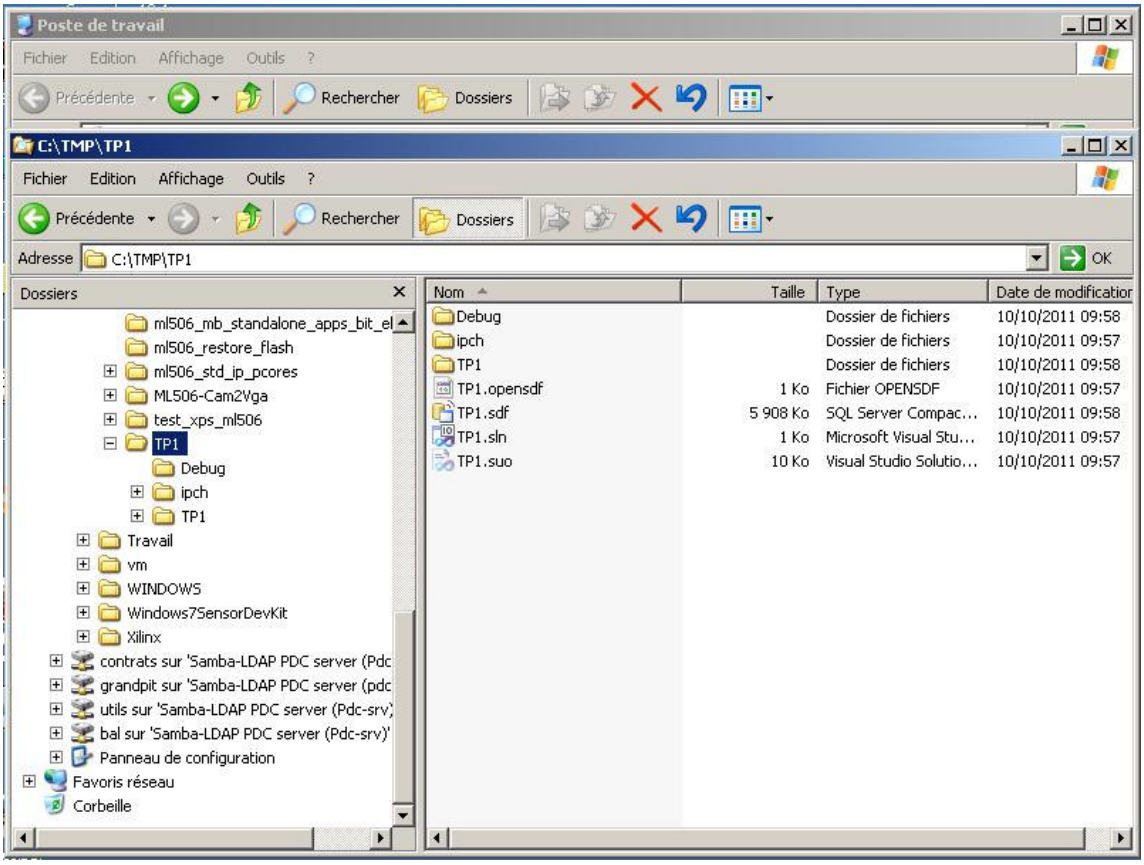
```
int a ;
```

std::cin>>a; //this code wait the user to type an integer in the console, it then stores this integer in ‘a’ variable which has been declared as an integer.

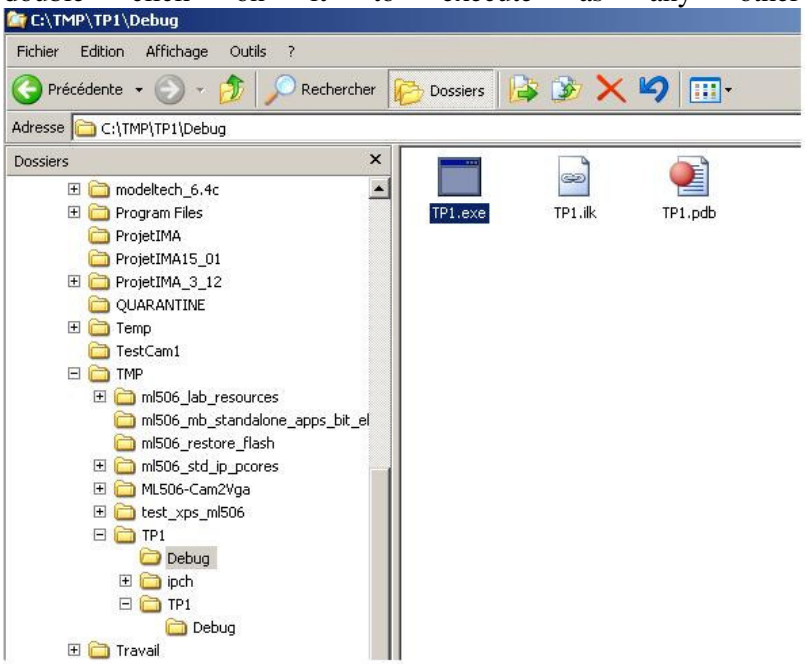
Congratulation, you have created your first C++ program using visual.

2. Running and saving the project

It is possible to run your program without visual, for that purpose you can close visual and go in c:\temp\TP1 where you created the project.



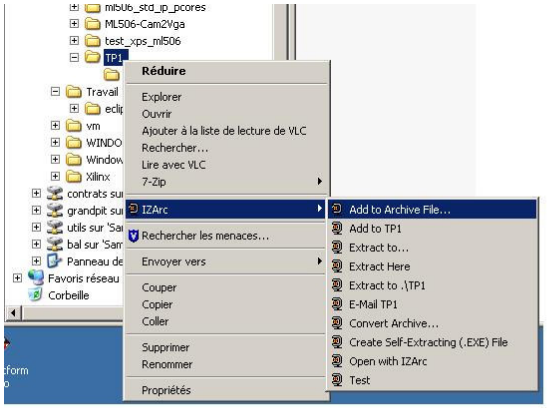
“Debug” subdirectories has been built: it contains you executable namely “TP1.exe”. You can double click on it to execute as any other windows application.

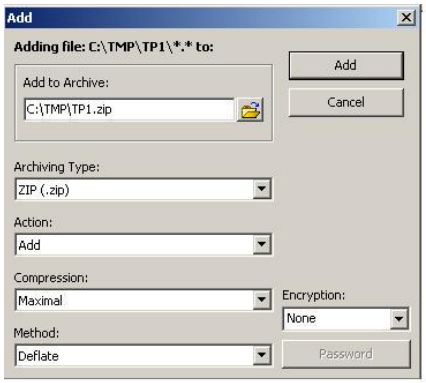


Take a look at the size of the whole visual project by right click on TP1 root directories, selecting “Properties”. It already takes a lot of place! So, if you want to archive this project you can reduce easily its size:

- Delete TP1.sdf (it’s a kind of data base for visual to accelerate the search operation and so on)
- Delete the second “Debug” directories located in TP1\TP1 subdirectories: this second Debug directories contains intermediate object code stored temporarly here.
- Do not delete T1\TP1 contains since it’s the place where your c++ files are stored!
- You can delete TP1\Debug if you don’t need to keep the application but only the visual project code. (you can rebuild it later).

Once this cleaning has been done you can use zip (or IZarc) to archive your project by right clicking on c:\temp\TP1 directories: “IZArc→ Add to Archive File” and select Add. The zip file TP1.zip has been created!





3. Programming with arrays

Let's declare an array of integers. Add this line in the main function:

int myArray[] = {23,12,10,8,2,3,5,15};

Let's get is length and store it in a scalar integer len :

int len = sizeof (myArray) / sizeof (int) ;
(There is no direct function returning the number of element of an array, but sizeof is a function that returns the memory space of an array and the size of any type –in bytes)

Question: how to display the content of myArray ?
(Try the following code if you have no idea:

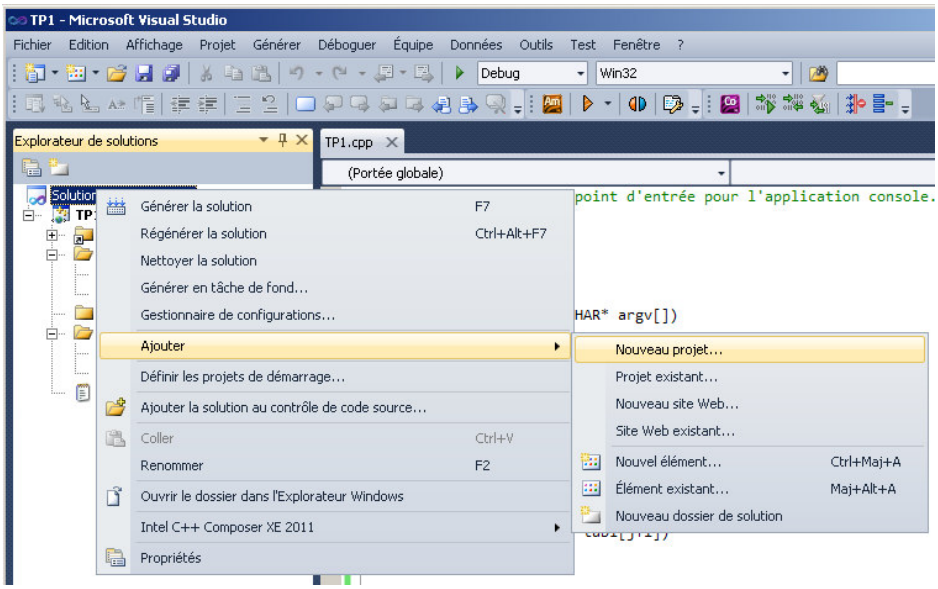
```
int val=22;
cout<<"Value of val is "<<val; )
```

Question: use a loop based on the "for" statement to display these elements
Question: declare and copy myArray in a second array. Print the contains of both arrays.
Question: write the code which will sort myArray
Question: put this code in a function

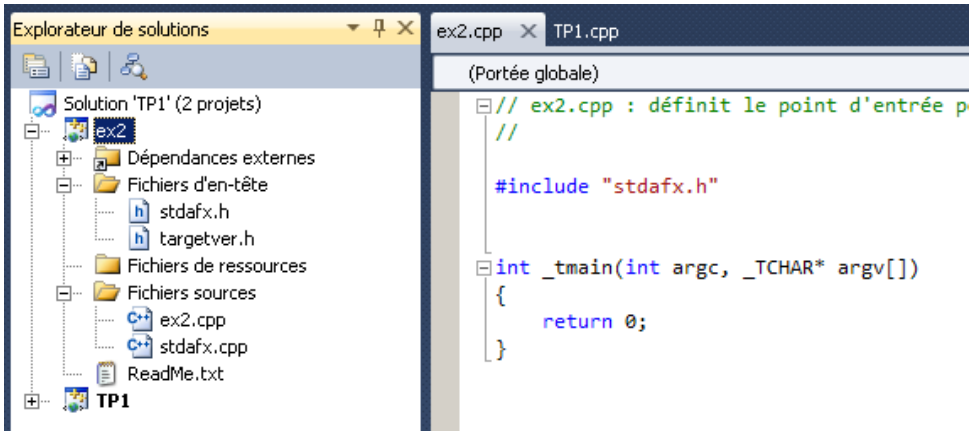
4. *Oriented Object Programming:*

Until now we didn't really use OOP, let's start with a first class: the class Point. It will be use to store x and y coordinate of a point.

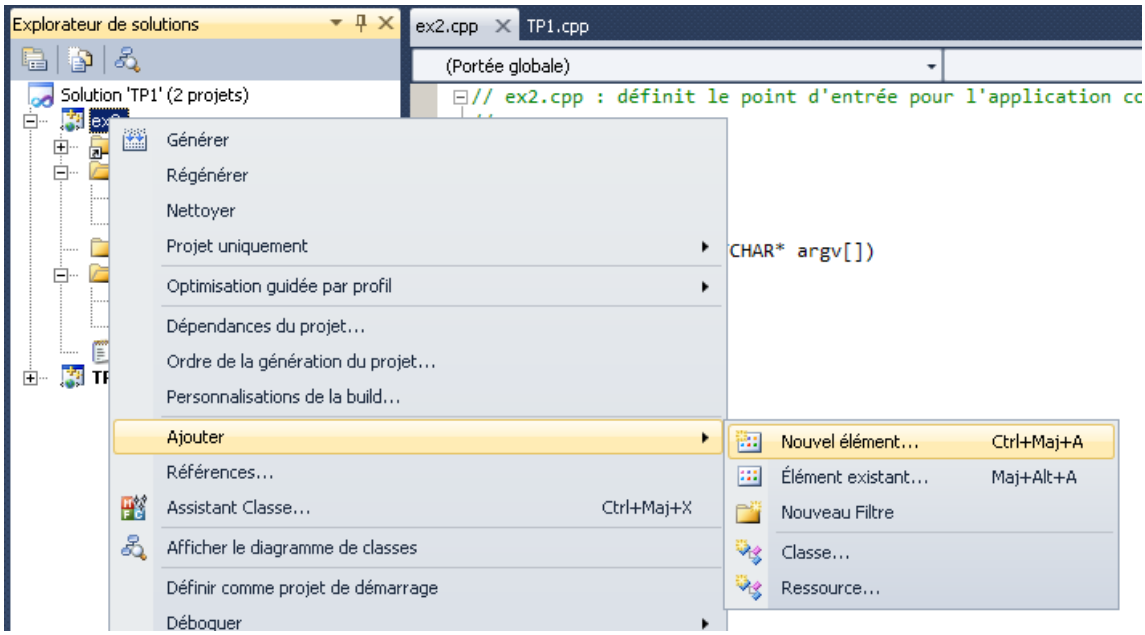
Start by creating a new project in your TP1 solution and name it "ex2"(for exercice2):



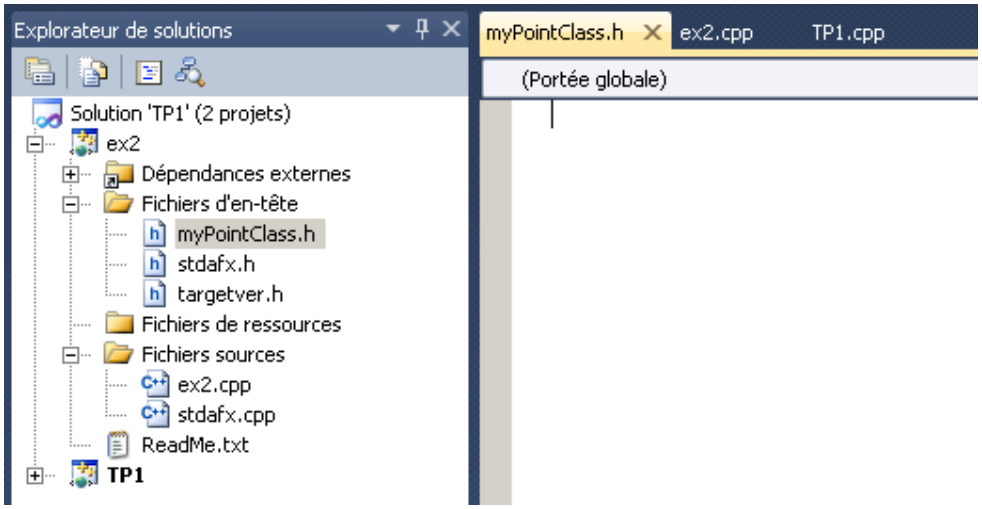
You should get this, where ex2 is the new project and contains the file ex2.cpp in which is declared the main function :



Now you will create your first class in a separate file. To do so, lets create a new C++ file in this project: right click on ex2, then select “Ajouter” (Add), “Nouvel Element” (New element)



Name it “myPointClass” and select “.h”, you get this screen: an empty file in the right side windows.



Declare your Point class in this file:

```
class Point {
	public :
		int x;
		int y;
};
```

In the main function add the following code:

```
Point p1, p2;
std::cout<<"values"<<p1.x<<","<<p1.y;
```

And also add #include "myPointClass.h" after the others #include.

1) You can now compile and execute. What's happening??

X and Y have not been initialized so we can do it in the so-called constructor of the class Point. Add the following code in the class Point declaration:

```
public :
	Point(){
		x=1;
		y=1;
	};
```

2) You can now compile and execute. Is it ok??

Yes, it should be.

3) Next we will modify class Point in order to declare the 2 members x and y as "private", then we will try to execute this code again:

```
	private :
		int x;
		int y;
```

Try to compile. What's happening??

A member declared as private can not be accessed from outside the class. So if we want to access x and y we have to use a public method of this class.

4) So, add the following public method (after the declaration of the constructor) :

```
	void Display(){
		std::cout<<"x="<<x;
		std::cout<<" y="<<y;
	};
```

(don't forget to add #include<iostream> on top of the file since you use cout now!)

We have to change the main function in order to use this method: replace the cout line by this one:

```
	p1.Display();
```

You can now compile and execute.

5) For better clarity when reading C++ code, we use to separate the class declaration and the methods of this class in 2 separates files. myPointClass.h should only contain the class declaration with just the names of the methods, and myPointClass.cpp should contain the code of each method of that class. So, add a new file "myPointClass.cpp" as you did for myPointClass.h. Move the Display() declaration in that file :

In myPointClass.cpp you should have:

```
#include "stdafx.h"
#include "myPointClass.h"

void Point::Display(){
	std::cout<<"x="<<x;
	std::cout<<" y="<<y;
};
```

In myPointClass.h you should have only the prototype of the display method:

```
void Display();
```

You can now compile and execute.

6) Add another constructor taking 2 arguments in order to init x and y
7) Using a class in another class :

Add the following class after the declaration of class Point :

```
class Polygone  {
	private:
		Point* _TabPoint;
		int _iTaille;
	public :
		Polygone (int iTaille) {
			_TabPoint = new Point[iTaille];
			_iTaille = 0;
		};
		void AddPoint(Point& P) {
			_TabPoint[_iTaille] = P;
			_iTaille++;
		};
		void UpdatePoint(int indice ,Point& P) {
			_TabPoint[indice] = P;
		};
};
```

Use this class in the main function:

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

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


	Point p1(0,0), p2(1,0), p3(1,1), p4(0,1);

	p4.Display();

	Polygone carre (4);
	carre.AddPoint(p1);
	carre.AddPoint(p2);
	carre.AddPoint(p3);
	carre.AddPoint(p4);

	int a;
	std::cin>>a;
	return 0;
}
```

Compile and execute.
Add a method to view the polygon's coordinates.