Tutorial n°3

Preliminaries : Create and add to your project two files called **Labs3.h** and **Labs3.cpp**.

1 Dynamic allocation

Declare and implement a function `AllocateArray` which dynamically allocates an array of n integers, n being provided by the user, and which returns the pointer to the first element of the allocated array.

2 Array deletion

Declare and implement a function `DeleteArray` which deletes an array of n integers, the array being given by a pointer to its first element by the user.

3 Array initialisation using pointers

Declare and implement a function `InitArray` to initialize an array of n integers : the user provides the pointer to the first element of the array and n. All the values will be randomly initialized to values in between 0 and 99.

4 On displaying an array using its pointer

Declare and implement a function `DisplayArray` which displays the n values of an array using the pointer to its first element.

5 Reverse

Declare and implement a function `ReverseArray` which reverses the order of the array (the first value becomes the last, the second becomes the before the last one, etc).

6 Swapping values of arrays represented by pointers

Declare then implement a function `SwapArrays(...)` that swaps all the values of two arrays represented by pointers. The arrays are supposed to have the same size.

7 On pointers used as function parameters

Let consider the two following functions:

```
void SpanArray1 (int* p, const int& n) {
for (int i=0; i<n; p++, i++) { }
}

void SpanArray2 (int* &p, const int& n ) {
for (int i=0; i<n; p++, i++) { }
}
```

Try both functions and display the array pointer before and after the calls. What differences do you see, how can you explain such result? What happens if the array has been statically allocated?

8 Allocation and deallocation of monodimensional and bidimensional arrays represented by pointers

1. Declare then implement a function `AllocateMatrix(...)` that returns a pointer to an array of arrays of $n \times m$ integer.
2. Declare then implement a function `InitializeMatrix(...)` that randomly initializes all the values in between 0 and 99.
3. Declare then implement a function `Deletematrix(...)` that takes and deletes this kind of matrix.
4. Declare then implement a function `DisplayMatrix(...)` that displays the address of the matrix in the memory and all its elements.

In the main procedure, try the following: create an array or a matrix then display it immediately. What do you remark?

9 A little bit of geometry

Declare and implement functions to compute the dot product (in $3D$ at least, nD suggested) and the inner product (in 3D).

10 Matrix multiplication in the general case

Declare and implement a function `MatrixProduct(...)` that returns the matricial product of two matrices of arbitraty dimensions. What are the differences with the exercice with static arrays?

11 Pointers arithmetic

You may have used indexes and the offset operator in the previous exercices. Redo them, **WITHOUT** using indexes and offset operator [ ] (when possible).