Compter et énumérer

Les TPs de tout le semestre seront réalisés en `C++`. Le but n'est pas d'utiliser toute la puissance du `C++`, mais simplement quelques outils qui nous faciliterons la tâche, en particulier la fonction `cout` à la place de `printf`, et la structure de donnnées `vector` pour faire des tableaux dynamiques. Les fichiers à télécharger sont disponibles à http://pagesperso.g-scop.fr/~pastorl/.

NB : Les questions marquées d'une ou plusieurs étoiles sont plus compliquées. Elles ne sont à traiter que lorsque le reste est fait.

Exercice 1 : Mise en route

- Téléchargez le fichier `HelloWorld.cpp`. Compiler à l'aide de la commande `g++ HelloWorld.cpp -o hello` puis lancez-le avec la commande `./hello`. Observez bien comment est utilisée la commande `cout`.
- Pour tester nos futures fonctions efficacement, il sera pratique de passer un paramètre directement dans la console lors de l'execution. Téléchargez le ficher `HelloWorldAvecParam.cpp`, compilez-le et testez-le à l'aide de la commande `./HelloWorldAvecParam 5`. Prédisez le comportement des tests suivants, puis testez-les :
  - `./HelloWorldAvecParam 3`
  - `./HelloWorldAvecParam 4 Bla Toto`
  - `./HelloWorldAvecParam Toto 5`
  - `./HelloWorldAvecParam`

  Il est conseillé d'aller voir la documentation en ligne de la fonction `atoi`.

Exercice 2 : Vector - Prise en main

Pour manipuler des tableaux facilement, nous allons utiliser la classe `vector` de la librairie standard. Télécharger le fichier `TestVector.cpp`, lisez-le et testez-le. On retiendra en particulier que :

- Il faut ajouter ____ dans l'en-tête.
- Pour déclarer un vector d'entiers `toto`, on utilise : ____.
- Pour accéder au `i`-ème élément du vector `toto`, on utilise : ____.
- Pour obtenir la taille du vector `toto`, on utilise : ____.
- Les cases sont numérotées de ____ à ____.
- Pour ____ un élément à la fin du vector `toto`, on utilise ____.
- Pour ____ un élément à la fin du vector `toto`, on utilise ____.

Nous découvrirons plus de méthodes de cette classe dans les prochains TP, en particulier nous ferons des `vector` de `vector<int>` pour représenter des matrices d'adjacence de graphes.

Exercice 3 : Calcul de la factorielle et des coefficients binomiaux

1. Ecrire une fonction `int factorielle(int n)` qui renvoie la valeur $n!$. Donner une version itérative et récursive de cette fonction. Tester votre fonction.
2. Ecrire une fonction `int coefficientbinomiaux(int n, int k)` qui calcule la valeur $\binom{n}{k}$ (vous pouvez utiliser la fonction précédénte, qui n'est pas la manière optimale de procéder).
3. Afficher le triangle de Pascal sur les 6 premières lignes (là encore, vous pouvez utiliser la fonction précédente, même si ce n'est pas optimal). Le résultat obtenu pourra ressembler à :

```
1  1
1  2  1
1  3  3  1
1  4  6  4  1
1  5  10  10  5  1
1  6  15  20  15  6  1
```

Exercice 4 : Fibonacci

Le suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ est définie par récurrence de la manière suivante :

- $F_0 = 0$
- $F_1 = 1$
- Pour $n > 0$, $F_{n+1} = F_n + F_{n-1}$

Écrire une fonction qui calcule le n-ième terme de la suite de Fibonacci. Donner une version récursive et itérative. Comparer les deux fonctions : laquelle est la plus rapide (en terme de nombre d'opérations) ?

Exercice 5 : Somme d'entiers

Dans chacun des cas suivants, écrire une fonction qui prend un entier n en argument et qui calcule la somme de :
- tous les entiers de 1 à n,
- tous les entiers impairs de 1 à n,
- tous les carrés des entiers de 1 à n.

Dans chacun des cas, tester votre fonction et essayer de deviner une formule qui donne la valeur de la somme en fonction de n. (Certaines formules seront vues et démontrées en TD).

Exercice 6 : Sous-ensemble à k elements de $\{1, ..., n\}$

1. Ecrire à la main tous les sous-ensembles (non ordonnés) à deux éléments de $\{1, ..., 5\}$.
2. Ecrire une fonction qui prend un entier n en argument et qui affiche la liste de tous les sous-ensembles à deux éléments de $\{1, ..., n\}$. Modifier votre fonction pour qu'elle renvoie le nombre de sous-ensembles affichés. Deviner une formule pour ce nombre.
3. Faire de même avec les sous-ensembles à trois éléments de $\{1, ..., n\}$.
4. Faire de même (question 3) pour les sous-ensembles ordonnés à trois éléments de $\{1, ..., n\}$.
5. (**) Ecrire un programme qui énumère les sous-ensembles à k éléments de $\{1, ..., n\}$.

Exercice 7 : Mots de longueur n sur un alphabet à k lettres

Rappel : Pour écrire la i-ème lettre de l'alphabet on pourra utiliser la commande `putchar('a'+i-1);`.

1. Ecrire à la main tous les mots de quatre lettres sur l'alphabet $\{a, b\}$.
2. Ecrire une fonction qui prend un entier k en argument et qui affiche la liste de tous les mots de quatre lettres utilisant les k premières lettres de l'alphabet. Modifier votre fonction pour qu'elle renvoie le nombre de mots affichés. Deviner une formule pour ce nombre.
3. (*) Modifier le programme précédent pour qu'il n'affiche que les mots contenant au plus deux fois la lettre `'a'`. Compter le nombre de tels mots (avec le programme) et essayer de deviner une formule pour ce nombre.
4. (**) Faire une fonction qui prend en arguments deux entiers n et k et qui affiche tous les mots de longueur n utilisant les k premières lettres de l'alphabet.

HelloWorld.cpp

```cpp
#include <stdlib.h>
#include <iostream>

using namespace std;


/* Hello World. */
int main(int argc, char* argv[])
{

    cout << "Hello World !" << endl;
    return EXIT_SUCCESS;
}
```

HelloWorldAvecParam.cpp

```cpp
#include <stdlib.h>
#include <iostream>

using namespace std;


/* Hello World avec 1 paramètre décidant
 * le nombre de répétitions.
 */
int main(int argc, char* argv[])
{
    /* atoi transforme une chaine de
     * caractères en entier.
     */
    int n=atoi(argv[1]);
    int i;

    for (i=1; i<=n; i++)
    {
        cout << "(Ligne n°:" << i
             << ") Hello World !" << endl;
    }
    cout << "argv est un tableau contenant "
         << argc << " entrées: "<< endl;
    cout << "La première est: "
         << argv[0] << endl;
    cout << "Les suivantes sont: ";
    for (i=1; i<argc; i++)
    {
        cout<< argv[i] << " " ;
    }
    cout << endl;

    return EXIT_SUCCESS;
}
```

TestVector.cpp

```cpp
#include <stdlib.h>
#include <iostream>
#include <vector>

using namespace std;


/* Affiche dans la console le contenu
 * du vector<int> tab.
 */
void affiche_vect(vector<int> &tab)
{
    int taille=tab.size();
    int i;

    for (i=0; i<taille; i++)
    {
        cout << tab[i];
    }

    cout << endl;
}









/* Tests divers sur la classe vector. */
int main(int argc, char* argv[])
{
    int i, n;
    if (argc>1)
    {
        n=atoi(argv[1]);
    }
    else
    {
        n=5;
    }
    cout<< endl;
    vector<int> monVector(n);
    cout<< "monVector contient:" << endl;
    affiche_vect(monVector);

    cout<< "La méthode size donne la taille "
           "du vector: "
        << monVector.size() << endl;

    for (i=0; i<n; i++)
    {
        monVector[i]=monVector[i]+i;
    }

    cout << "Après modification, "
            "monVector contient: ";
    affiche_vect(monVector);

    vector<int> unAutreVect;
    unAutreVect.assign(6, 1);
    cout<< "unAutreVect contient: " ;
    affiche_vect(unAutreVect);
    unAutreVect.push_back(3);
    cout << "Après modification, unAutreVect "
            "contient: ";
    affiche_vect(unAutreVect);
    unAutreVect.pop_back();
    cout << "Après modification, unAutreVect "
            "contient: ";
    affiche_vect(unAutreVect);
    cout<< endl;


    return EXIT_SUCCESS;

}
```