1 Sous-programmes

1.1 Géométrie

Ecrire un programme demandant à l'utilisateur de saisir une valeur numérique positive n et affichant un carré, une croix et un losange, tous de côté n. Par exemple, si n = 10, l'exécution donne

Saisissez la taille des figures 10

Vous définirez des sous-programmes de quelques lignes et au plus deux niveaux d'imbrication. Vous ferez attention à ne jamais écrire deux fois les mêmes instructions. Pour ce faire, complétez le code source suivant :

```
using System;
namespace tests
{
    class MainClass
    {
        /* Affiche le caractère c */
        public static void afficheCaractere(char c)
        {

        }
        /* ************************************ */
        /*
        Affiche n fois le caractère c, ne revient pas à la ligne après le dernier caractère.
        */
        public static void ligneSansReturn(int n, char c)
        {
        }
        /* ************************************ */
        /*
        Affiche n fois le caractère c, revient a la ligne après le dernier caractère.
        */
        public static void ligneAvecReturn(int n, char c)
        {
        }
        /* ************************************ */
        /*         Affiche n espaces .          */
        public static void espaces(int n)
        {
        }
        /* ************************************ */
        /*         Affiche le caractère c a la colonne i, ne revient pas à la ligne après. */
        public static void unCaractereSansReturn(int i, char c)
        {
        }
        /* ************************************ */
        /*         Affiche le caractère c à la colonne i, revient a la ligne après.         */
        public static void unCaractereAvecReturn(int i, char c)
        {
        }
        /* ************************************ */
        /*  Affiche le caractère c aux colonnes i et j, revient a la ligne après.
      */
        public static void deuxCaracteres(int i, int j, char c)
        {
        }
        /* ************************************ */
```

```
        /*
        Affiche un carre de cote n .
        */
        public static void carre(int n)
        {
        }
        /* ************************************ */
        /*         Affiche un chapeau dont la pointe - non affichee - est sur la colonne centre , avec
les caractères c.         */
        public static void chapeau(int centre, char c)
        {
        }
        /* ************************************ */
        /*
        Affiche un chapeau a l ' envers avec des caracteres c,
        la pointe - non affichee - est à la colonne c entre
        */
        public static void chapeauInverse(int centre, char c)
        {
        }
        /* ************************************ */
        /*
        Affiche un losange de cote n .
        */
        public static void losange(int n)
        {
        }
        /* ************************************ */
        /*
        Affiche une croix de cote n
        */
        public static void croix(int n)
        {
        }
        /* ************************************ */
        public static void Main(string[] args)
        {
            int taille;
            Console.WriteLine(" Saisissez la taille des figures : ");
            taille = int.Parse(Console.ReadLine());
            carre(taille);
            losange(taille);
            croix(taille);
        }
    }

}
```

1.2 Arithmétique

Exercice 1 - chiffres et nombres

- Ecrire la fonction public static int unites(int n) retournant le chiffre des unités du nombre n.
- Ecrire la fonction public static int dizaines(int n) retournant le chiffre des dizaines du nombre n.
- Ecrire la fonction public static int extrait(int n, int p) retournant le p-ième chiffre de représentation décimale de n en partant des unités.
- Ecrire la fonction public static int nbChiffres(int n) retournant le nombre de chiffres que comporte la représentation décimale de n.
- Ecrire la fonction public static int sommeChiffres(int n) retournant la somme des chiffres de n.

Exercice 2 - Nombres amis

Soient a et b deux entiers strictement positifs. a est un diviseur strict de b si a divise b et a != b. Par exemple, 3 est un diviseur strict de 6. Mais 6 n'est pas un diviseur strict 6. a et b sont des nombres amis si la somme des diviseurs stricts de a est b et si la somme des diviseurs de b est a. Le plus petit couple de nombres amis connu est 220 et 284.

- Ecrire une fonction public static int sommeDiviseursStricts(int n), elle doit renvoyer la somme des diviseurs stricts de n.
- Ecrire une fonction public static bool sontAmis(int a, int b), elle doit renvoyer 1 si a et b sont amis, 0 sinon.

Exercice 3 - Nombres parfaits

Un nombre parfait est un nombre égal à la somme de ses diviseurs stricts. Par exemple, 6 a pour diviseurs stricts 1, 2 et 3, comme 1 + 2 + 3 = 6, alors 6 est parfait.

1. Est-ce que 18 est parfait ?

2. Est-ce que 28 est parfait ?

3. Que dire d'un nombre ami avec lui-même ?

4. Ecrire la fonction public static bool estParfait(int n), elle doit retourner true ssi n est un nombre parfait.

1.3 Passage de tableaux en paramètre

Exercice 4 - Somme

- Ecrire une fonction public static int somme(int[] T) retournant la somme des éléments de T.

Exercice 5 - Minimum

- Ecrire une fonction public static int min(int[] T) retournant la valeur du plus petit élément de T.

Exercice 6 - Recherche

- Ecrire une fonction public static bool existe(int[] T, int k) retournant true si et seulement si k est un des éléments de T.

Exercice 7 - Somme des éléments pairs

- Ecrivez le corps de la fonction public static int sommePairs(int[] T), sommePairs(T) retourne la somme des éléments pairs de T. N'oubliez pas que a%b est le reste de la division entière de a par b.

Exercice 8 - Vérification

- Ecrivez le corps de la fonction public static bool estTrie(int[] T), estTrie(T) retourne vrai si et seulement si les éléments de T sont triés dans l'ordre croissant.

Exercice 9 - Permutation circulaire

- Ecrire une fonction public static void permutation(int[] T) effectuant une permutation circulaire vers la droite des éléments de T .

Exercice 10 - Miroir

- Ecrire une fonction public static void miroir(int[] T inversant l'ordre des éléments de T.

1.4 Décomposition en facteurs premiers

On rappelle qu'un nombre est premier s'il n'est divisible que par 1 et par lui-même. Par convention, 1 n'est pas premier.

Exercice 11

- Ecrivez une fonction public static bool estPremier(int x, int[] premiers, int k) retournant vrai si et seulement si x est premier. Vous vérifierez la primarité de x en examinant les restes des divisions de x par les k premiers éléments de premiers. On suppose que k est toujours supérieur ou égal à 1.

Exercice 12

- Modifiez la fonction précédente en tenant compte du fait que si aucun diviseur premier de x inférieur à x n'a été trouvé, alors x est premier

Exercice 13

- Ecrivez une fonction public static int [] trouvePremiers(int n) retournant un tableau conte- nant les n premiers nombres premiers.